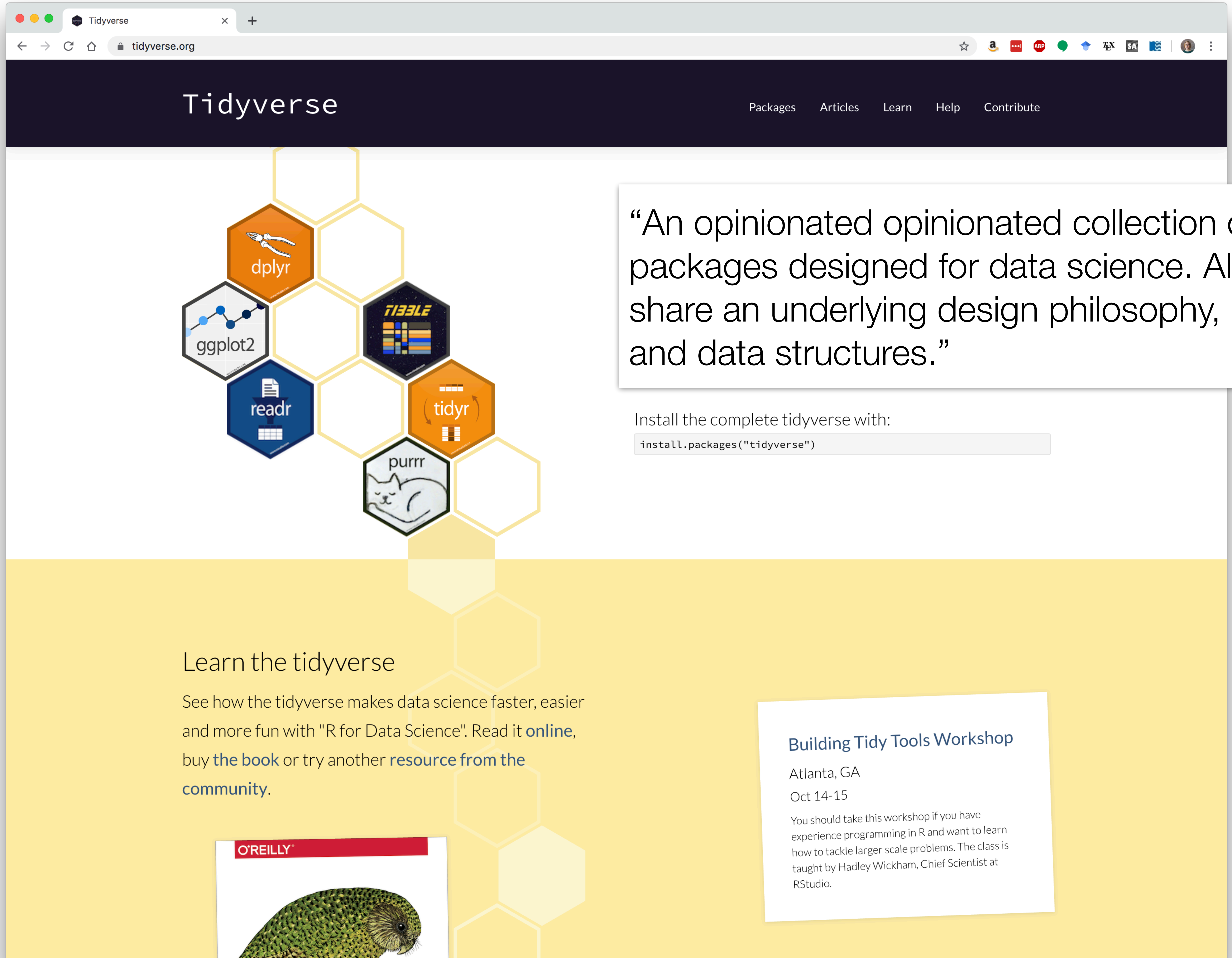
Why the tidyverse?

Amelia McNamara @AmeliaMN
www.amelia.mn

What is the tidyverse?



Tidyverse

Install the complete tidyverse with:

```
install.packages("tidyverse")
```

Learn the tidyverse

See how the tidyverse makes data science faster, easier and more fun with "R for Data Science". Read it online, buy the book or try another resource from the community.

Building Tidy Tools Workshop

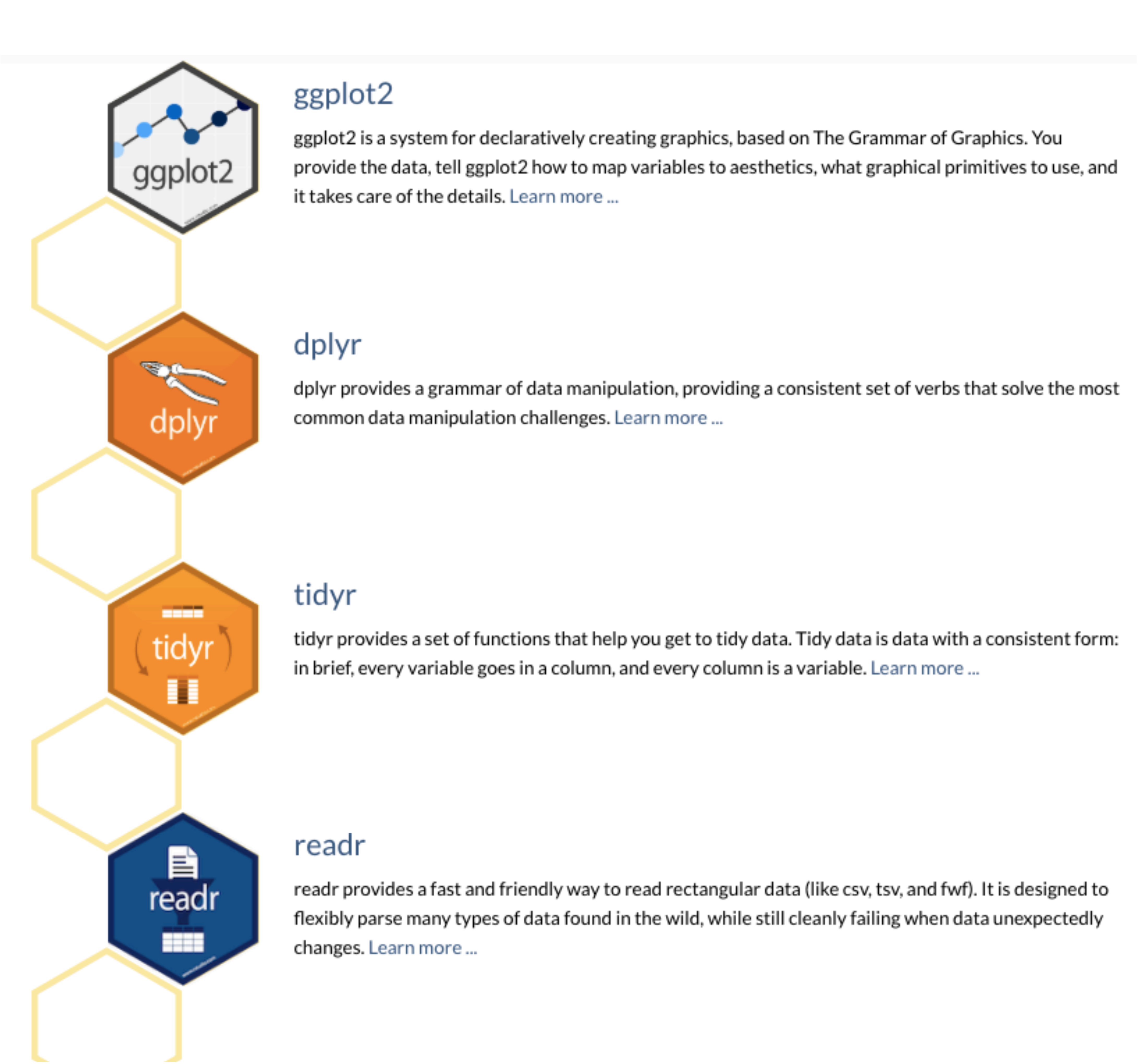
Atlanta, GA

Oct 14-15

You should take this workshop if you have experience programming in R and want to learn how to tackle larger scale problems. The class is taught by Hadley Wickham, Chief Scientist at RStudio.

> "An opinionated opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures."

Core tidyverse



ggplot2

ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphics. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details. Learn more ...

dplyr

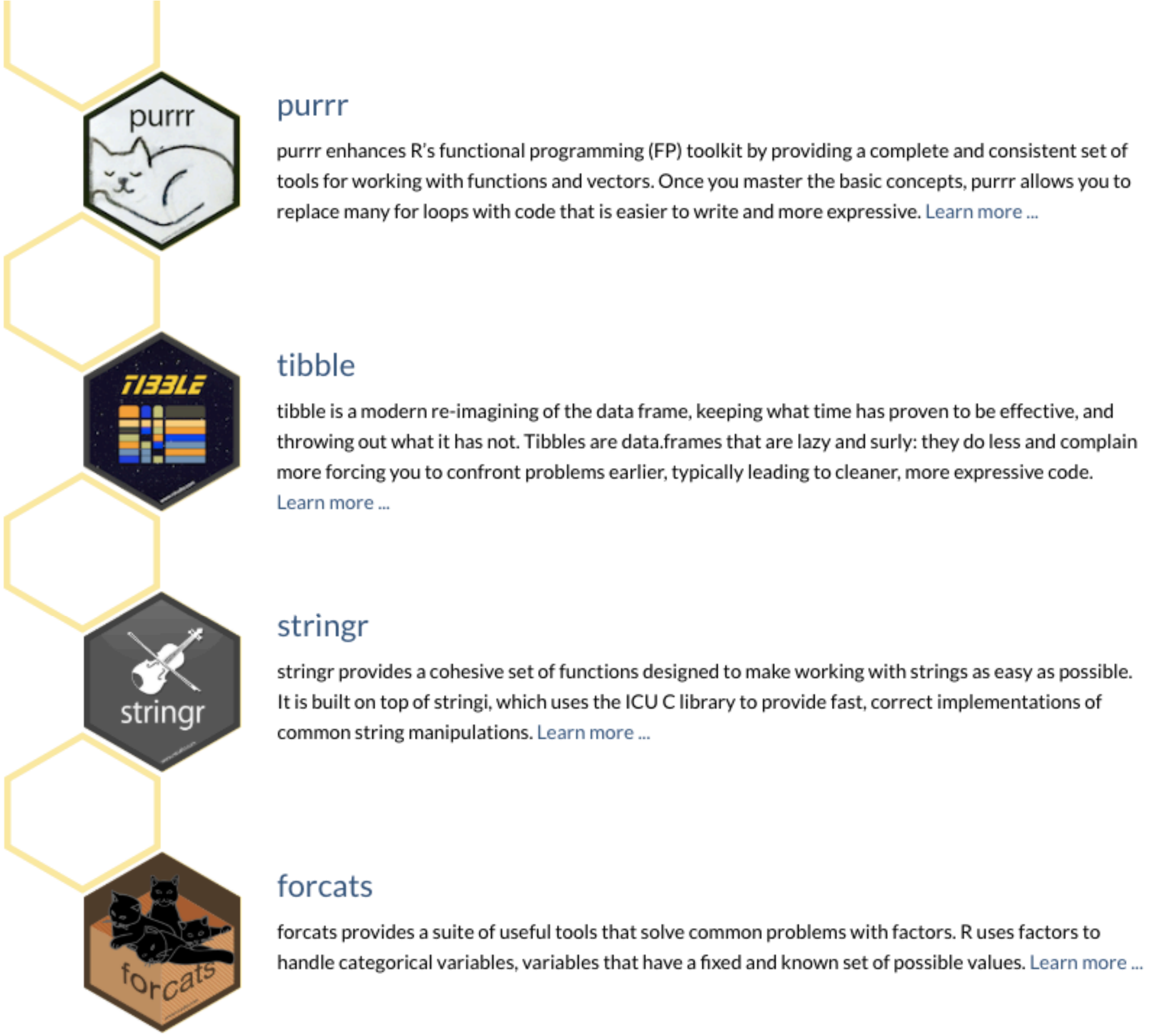
dplyr provides a grammar of data manipulation, providing a consistent set of verbs that solve the most common data manipulation challenges. Learn more ...

tidyr

tidyr provides a set of functions that help you get to tidy data. Tidy data is data with a consistent form: in brief, every variable goes in a column, and every column is a variable. Learn more ...

readr

readr provides a fast and friendly way to read rectangular data (like csv, tsv, and fwf). It is designed to flexibly parse many types of data found in the wild, while still cleanly failing when data unexpectedly changes. Learn more ...



purrr

purrr enhances R's functional programming (FP) toolkit by providing a complete and consistent set of tools for working with functions and vectors. Once you master the basic concepts, purrr allows you to replace many for loops with code that is easier to write and more expressive. Learn more ...

tibble

tibble is a modern re-imagining of the data frame, keeping what time has proven to be effective, and throwing out what it has not. Tibbles are data.frames that are lazy and surly: they do less and complain more forcing you to confront problems earlier, typically leading to cleaner, more expressive code. Learn more ...

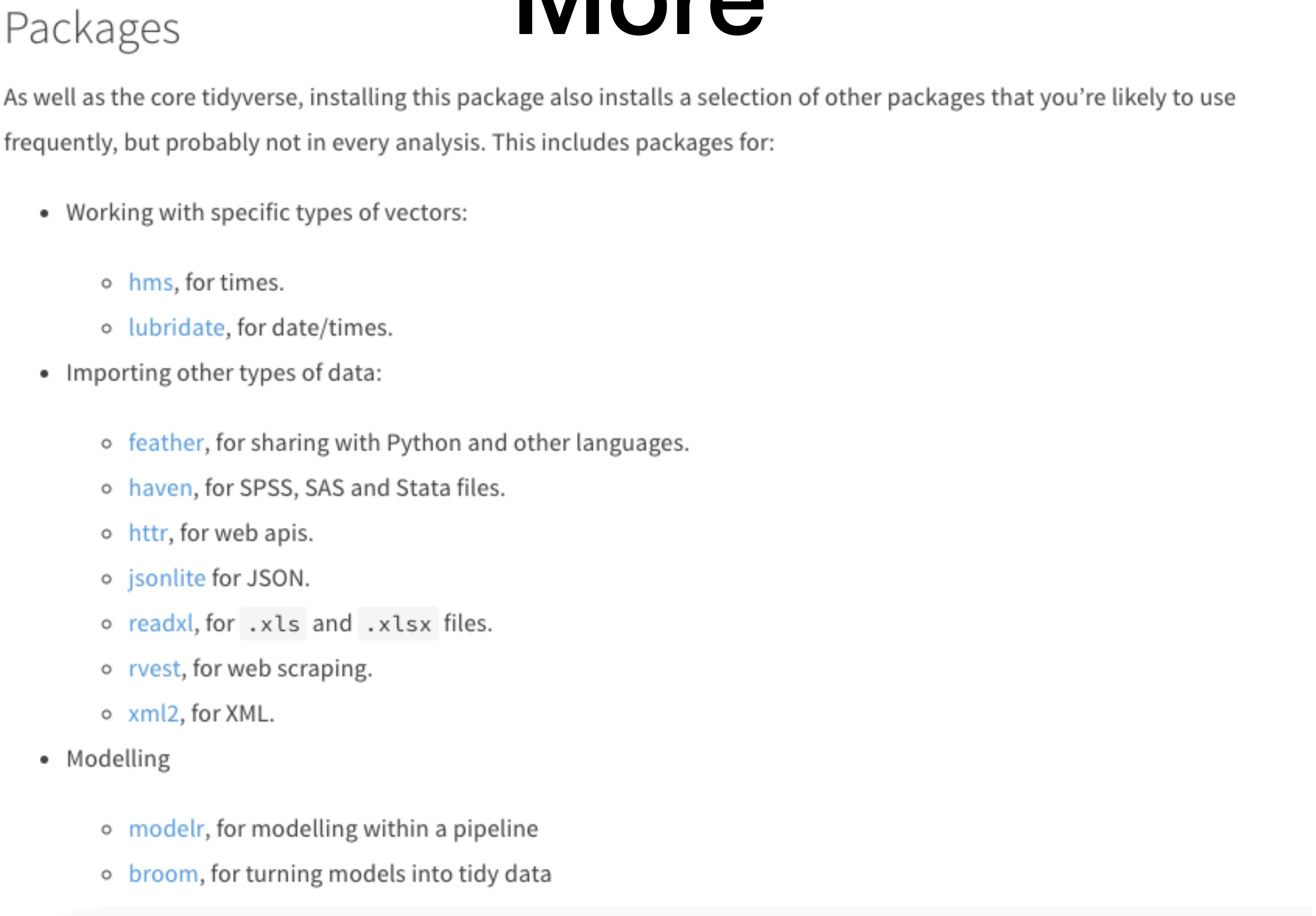
stringr

stringr provides a cohesive set of functions designed to make working with strings as easy as possible. It is built on top of stringi, which uses the ICU C library to provide fast, correct implementations of common string manipulations. Learn more ...

forcats

forcats provides a suite of useful tools that solve common problems with factors. R uses factors to handle categorical variables, variables that have a fixed and known set of possible values. Learn more ...

More



Packages

As well as the core tidyverse, installing this package also installs a selection of other packages that you're likely to use frequently, but probably not in every analysis. This includes packages for:

- Working with specific types of vectors:
  - hms, for times.
  - lubridate, for date/times.
- Importing other types of data:
  - feather, for sharing with Python and other languages.
  - haven, for SPSS, SAS and Stata files.
  - httr, for web apis.
  - jsonlite for JSON.
  - readxl, for `.xls` and `.xlsx` files.
  - rvest, for web scraping.
  - xml2, for XML.
- Modelling
  - modelr, for modelling within a pipeline
  - broom, for turning models into tidy data

Even more

tidytext (Julia Silge and David Robinson)

skimr (Elin Waring, Michael Quinn, Amelia McNamara, Eduardo Ariño de la Rubia, Hao Zhu, Shannon Ellis)

(fun rOpensci connection)

Tidyverse philosophy/manifesto

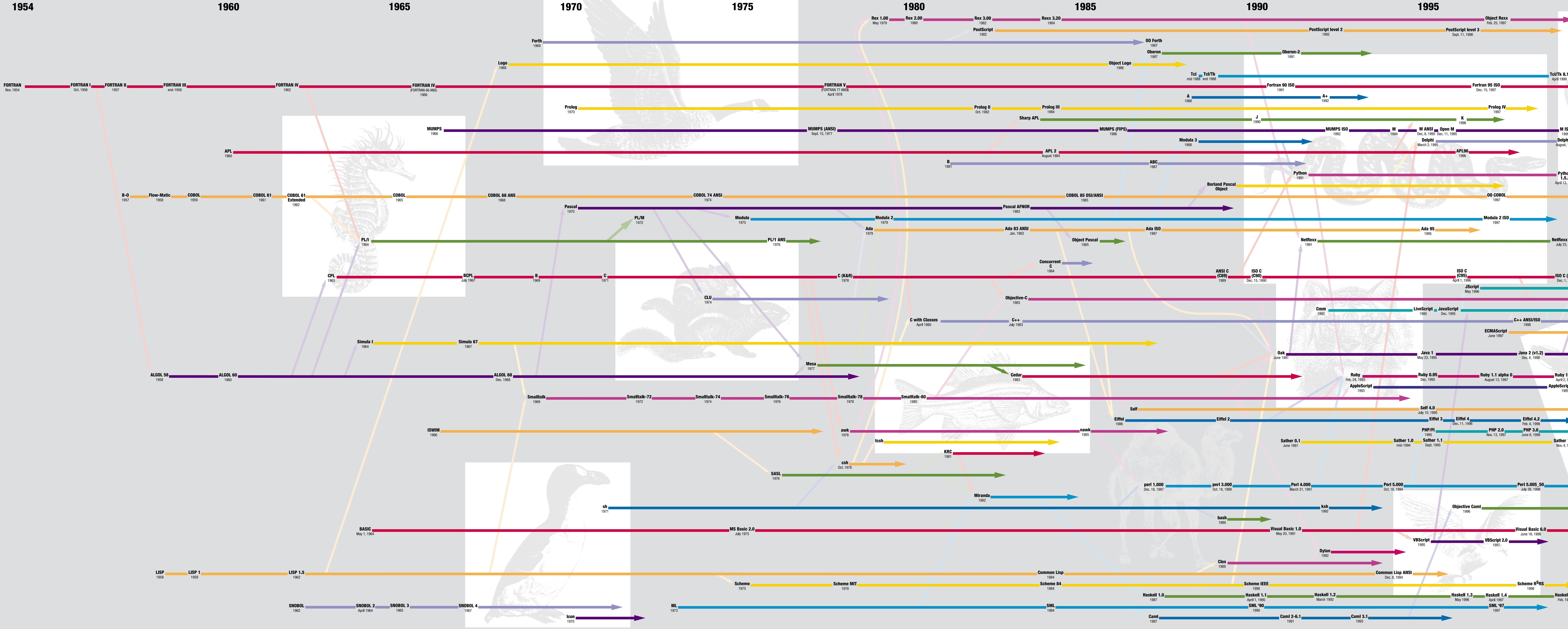
"There are four basic principles to a tidy API:

- Reuse existing data structures.
- Compose simple functions with the pipe.
- Embrace functional programming.
- Design for humans."

Pure,
predictable,
pipeable

https://tidyverse.tidyverse.org/articles/manifesto.html

History of Programming Languages



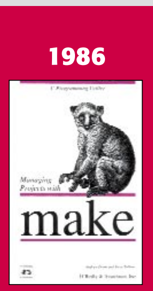
www.oreilly.com

For more than half of the fifty years computer programmers have been writing code, O'Reilly has provided developers with comprehensive, in-depth technical information. We've kept pace with rapidly changing technologies as new languages have emerged, developed, and matured. Whether you want to learn something new or need answers to tough technical questions, you'll find what you need in O'Reilly books and on the O'Reilly Network.

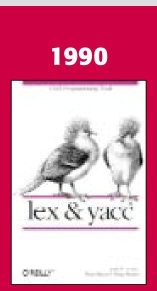
This timeline includes fifty of the more than 2500 documented programming languages. It is based on an original diagram created by Éric Lévénez (*www.levenez.com*), augmented with suggestions from O'Reilly authors, friends, and conference attendees.

For information and discussion on this poster, go to *www.oreilly.com/go/languageposter*.

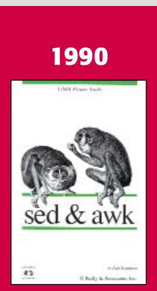
1986



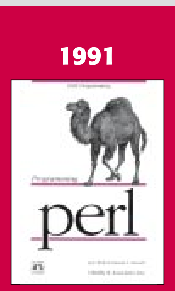
1990



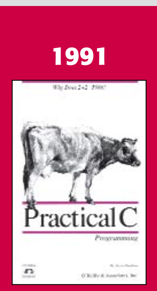
1990



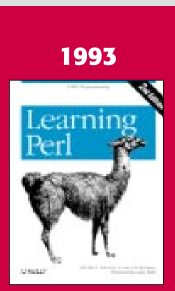
1991



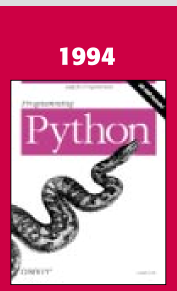
1991



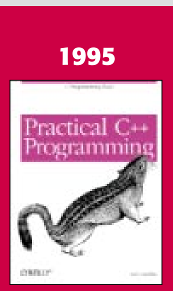
1993



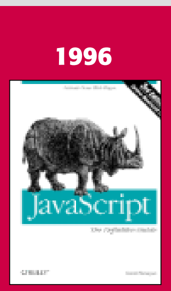
1994



1995

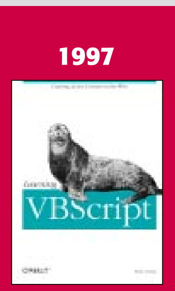


1996

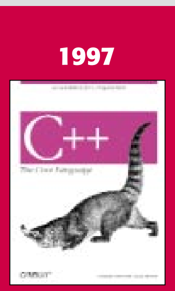


1996

1997



1997



Languages

1970
1975
1980
1985
1990
1995
2000
2001
2002
2003
2004

Rex 1.00 May 1979
Rex 2.00 1980
Rex 3.00 1982
Rexx 3.20 1984
Object Rexx Feb. 25, 1997

PostScript 1982
PostScript level 2 1992
PostScript level 3 Sept. 11, 1996

Forth 1969
OO Forth 1987

Oberon 1987
Oberon-2 1991

Logo 1968
Object Logo 1986

Tcl mid 1988
Tcl/Tk end 1988
Tcl/Tk 8.1 April 1999
Tcl/Tk 8.2.3 Dec. 16, 1999
Tcl/Tk 8.3 Oct. 22, 2001
Tcl/Tk 8.4 Sept. 10, 2002
Tcl/Tk 8.4.1 Oct. 22, 2002
Tcl/Tk 8.4.2 March 3, 2003
Tcl/Tk 8.4.3 May 20, 2003

TRAN IV
RAN 66 ANS)
1966
FORTRAN V (FORTRAN 77 ANSI) April 1978
Fortran 90 ISO 1991
Fortran 95 ISO Dec. 15, 1997
Fortran 2000 (draft) Sept. 30, 2002

A 1988
A+ 1992

Prolog 1970
Prolog II Oct. 1982
Prolog III 1984
Prolog IV 1997

Sharp APL
J 1990
K 1996

MUMPS 1966
MUMPS (ANSI) Sept. 15, 1977
MUMPS (FIPS) 1986
MUMPS ISO 1992
M 1994
M ANSI Dec. 8, 1995
Open M Dec. 11, 1995
M ISO 1999

Modula 3 1988

Delphi March 2, 1995
Delphi 5 August, 1999
Delphi 6 May 1, 2001
Delphi 7 August 6, 2002

APL 2 August 1984
APL96 1996

B 1981
ABC 1987

Python 1991
Python 1.5.2 April 13, 1999
Python 1.6 Sept. 5, 2000
Python 2.0 Oct. 16, 2000
Python 2.1 April 17, 2001
Python 2.2 Dec. 21, 2001
Python 2.2.1 April 10, 2002
Python 2.2.2 Oct. 14, 2002
Python 2.3a2 Feb. 19, 2003
Python 2.2.3 May 30, 2003
Python 2.3 July 29, 2003
Python 2.3.1 Sept. 23, 2003

Borland Pascal Object

COBOL 68 ANS 1968
COBOL 74 ANSI 1974
COBOL 85 OSI/ANSI 1985
OO COBOL 1997
COBOL 2002 (draft) 2001

Pascal 1970
Pascal AFNOR 1983

PL/M 1972

Modula 1975
Modula 2 1979
Modula 2 ISO 1997

Ada 1979
Ada 83 ANSI Jan. 1983
Ada ISO 1987
Ada 95 1995

PL/1 ANS 1976

Object Pascal 1985

NetRexx 1991
NetRexx 1.150 July 23, 1999

C# June 26, 2000
C# (ECMA) Dec. 13, 2001
C# (ISO) March 28, 2003

Concurrent C 1984

BCPL July 1967
B 1969
C 1971
C (K&R) 1978
ANSI C (C89) 1989
ISO C (C90) Dec. 15, 1990
ISO C (C95) April 1, 1996
ISO C (C99) Dec. 1, 1999

JScript May 1996
JScript

CLU 1974

Objective-C 1983

Cmm 1992
LiveScript 1995
JavaScript Dec. 1995
JavaScript 1.5

C with Classes April 1980
C++ July 1983
C++ ANSI/ISO 1998

ECMAScript June 1997
ECMAScript ed3 Dec., 1999

Simula 67 1967

ActionScript July, 2000
ActionScript 2.0 Sept., 2003

Oak June 1991
Java 1 May 23, 1995
Java 2 (v1.2) Dec. 8, 1998
Java 2 (v1.3) May 8, 2000
Java 2 (v1.4) early access Feb. 6, 2002
Java 2 (v1.4.0_01) June 4, 2002
Java 2 (v1.4.1) Sep. 2002
Java 2 (v1.4.1_02) Feb, 27, 2003
Java 2 (v1.4.1_03) June 11, 2003
Java 2 (v1.5) early release Dec. 19, 2003

Mesa 1977

ALGOL 68 Dec. 1968

Cedar 1983

Ruby Feb. 24, 1993
Ruby 0.95 Dec. 1995
Ruby 1.1 alpha 0 August 13, 1997
Ruby 1.3.2 April 2, 1999
Ruby 1.6.1 Sept. 27, 2000
Ruby 1.6.5 Sept. 19, 2001
Ruby 1.6.7 March 1, 2002
Ruby 1.6.8 Dec. 24, 2002
Ruby 1.8 August 4, 2003

AppleScript 1993
AppleScript 1.3.7 1999

Smalltalk 1969
Smalltalk-72 1972
Smalltalk-74 1974
Smalltalk-76 1976
Smalltalk-78 1978
Smalltalk-80 1980

Self
Self 4.0 July 10, 1995
Self 4.1 August 7, 2001
Self 4.1.6 Sept. 2002

Eiffel 1986
Eiffel 2
Eiffel 3
Eiffel 4 Dec. 11, 1996
Eiffel 4.2 Feb. 6, 1998

ISWIM 1966

awk 1978
nawk 1985

PHP/FI 1995
PHP 2.0 Nov. 13, 1997
PHP 3.0 June 6, 1998
PHP 4.0 May 22, 2000
PHP 4.1.0 Dec. 8, 2001
PHP 4.2.0 April 22, 2002
PHP 4.2.2 July 22, 2002
PHP 4.2.3 Sept. 6, 2002
PHP 4.3.0 Dec. 27, 2002
PHP 4.3.1 Feb. 17, 2003
PHP 4.3.2 May 29, 2003
PHP 4.3.3 August 25, 2003

tcsh

Sather 0.1 June 1991
Sather 1.0 mid-1994
Sather 1.1 Sept. 1995
Sather 1.2.1 Nov. 4, 1999

KRC 1981

csh Oct. 1978

SASL 1976

perl 1.000 Dec. 18, 1987
perl 3.000 Oct. 18, 1989
Perl 4.000 March 21, 1991
Perl 5.000 Oct. 18, 1994
Perl 5.005_50 July 26, 1998
Perl 5.6.0 March 28, 2000
Perl 5.7,0 Sept. 2, 2000
Perl 5.8.0 July 18, 2002
Perl 5.8.2 Oct., 2003

Miranda 1982

sh 1971
ksh 1992

Objective Caml 1996

bash 1989

MS Basic 2.0 July 1975
Visual Basic 1.0 May 20, 1991
Visual Basic 6.0 June 16, 1998
VB.NET July, 2000

VBScript 1995
VBScript 2.0 1997

Dylan 1992

Clos 1989

Common Lisp 1984
Common Lisp ANSI Dec. 8, 1994

Scheme 1975
Scheme MIT 1978
Scheme 84 1984
Scheme IEEE 1990
Scheme R^{5}RS 1998

Haskell 1.0 1987
Haskell 1.1 April 1, 1990
Haskell 1.2 March 1992
Haskell 1.3 May 1996
Haskell 1.4 April 1997
Haskell 98 Feb. 1999

SNOBOL 4 1967

ML 1973
SML 1984
SML '90 1990
SML '97 1997

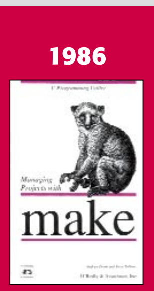
Icon 1970

Caml 1987
Caml 2-6.1 1991
Caml 3.1 1993

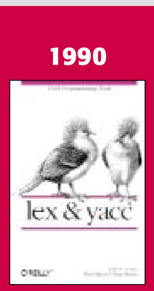
is timeline includes fifty of the more than 2500 documented
ogramming languages. It is based on an original diagram created
Éric Lévénez (*www.levenez.com*), augmented with suggestions
om O'Reilly authors, friends, and conference attendees.

r information and discussion on this poster,
to www.oreilly.com/go/languageposter.

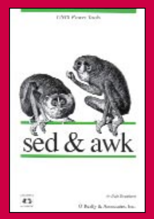
1986



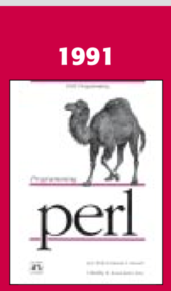
1990



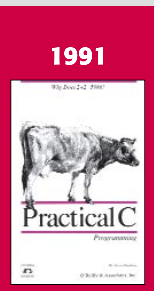
1990



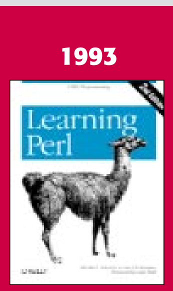
1991



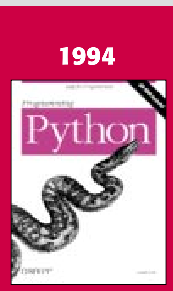
1991



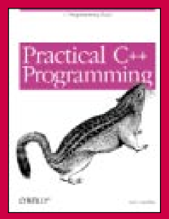
1993



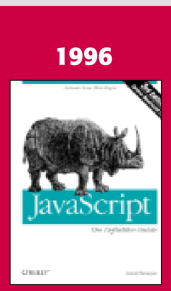
1994



1995

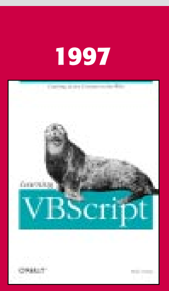


1996

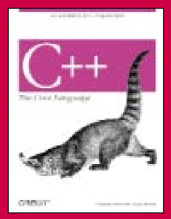


1996

1997

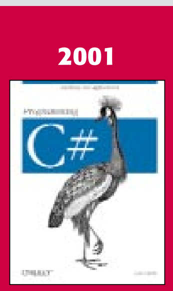


1997

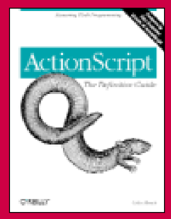


2000

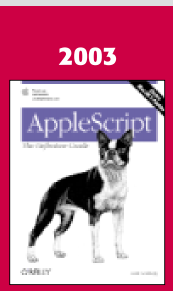
2001



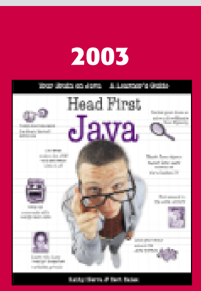
2001



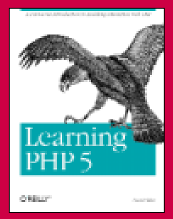
2003



2003

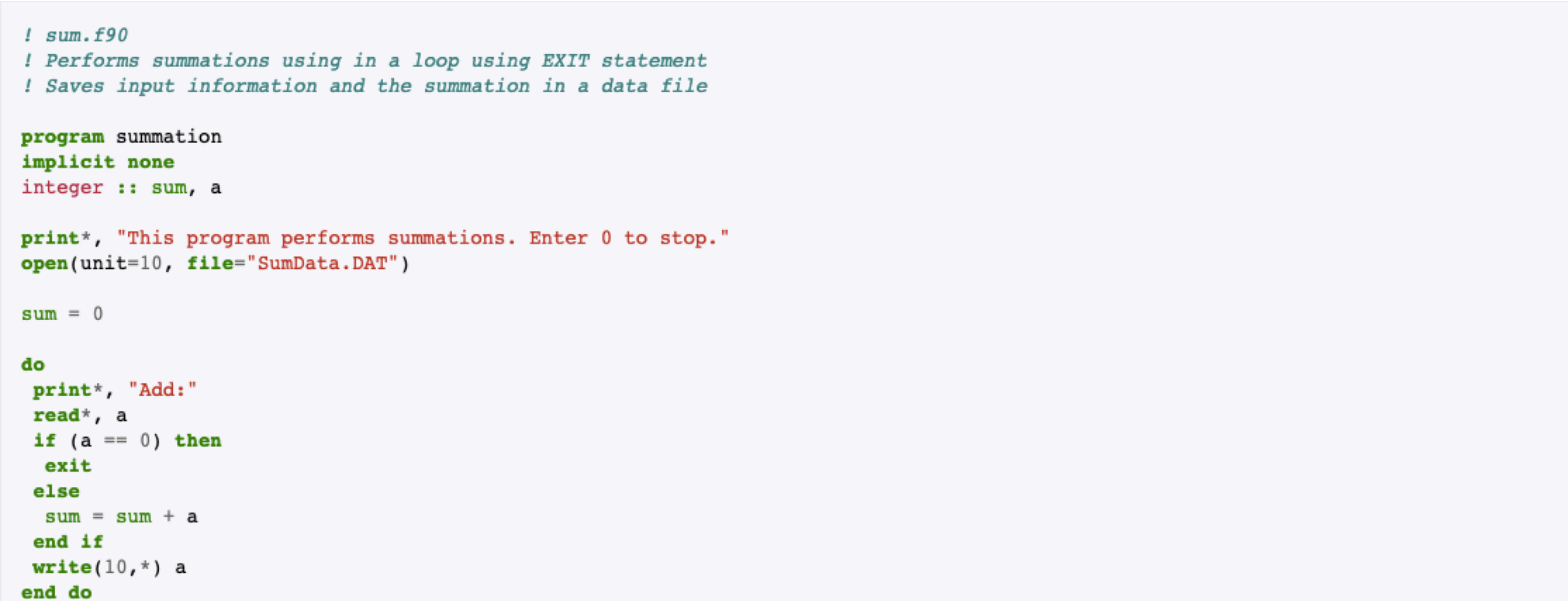


2004



FORTRAN

Developed in the 1950s, still exists today



```
! sum.f90
! Performs summations using in a loop using EXIT statement
! Saves input information and the summation in a data file

program summation
implicit none
integer :: sum, a

print*, "This program performs summations. Enter 0 to stop."
open(unit=10, file="SumData.DAT")

sum = 0

do
 print*, "Add:"
 read*, a
 if (a == 0) then
  exit
 else
  sum = sum + a
 end if
 write(10,*) a
end do
```

Wikipedia

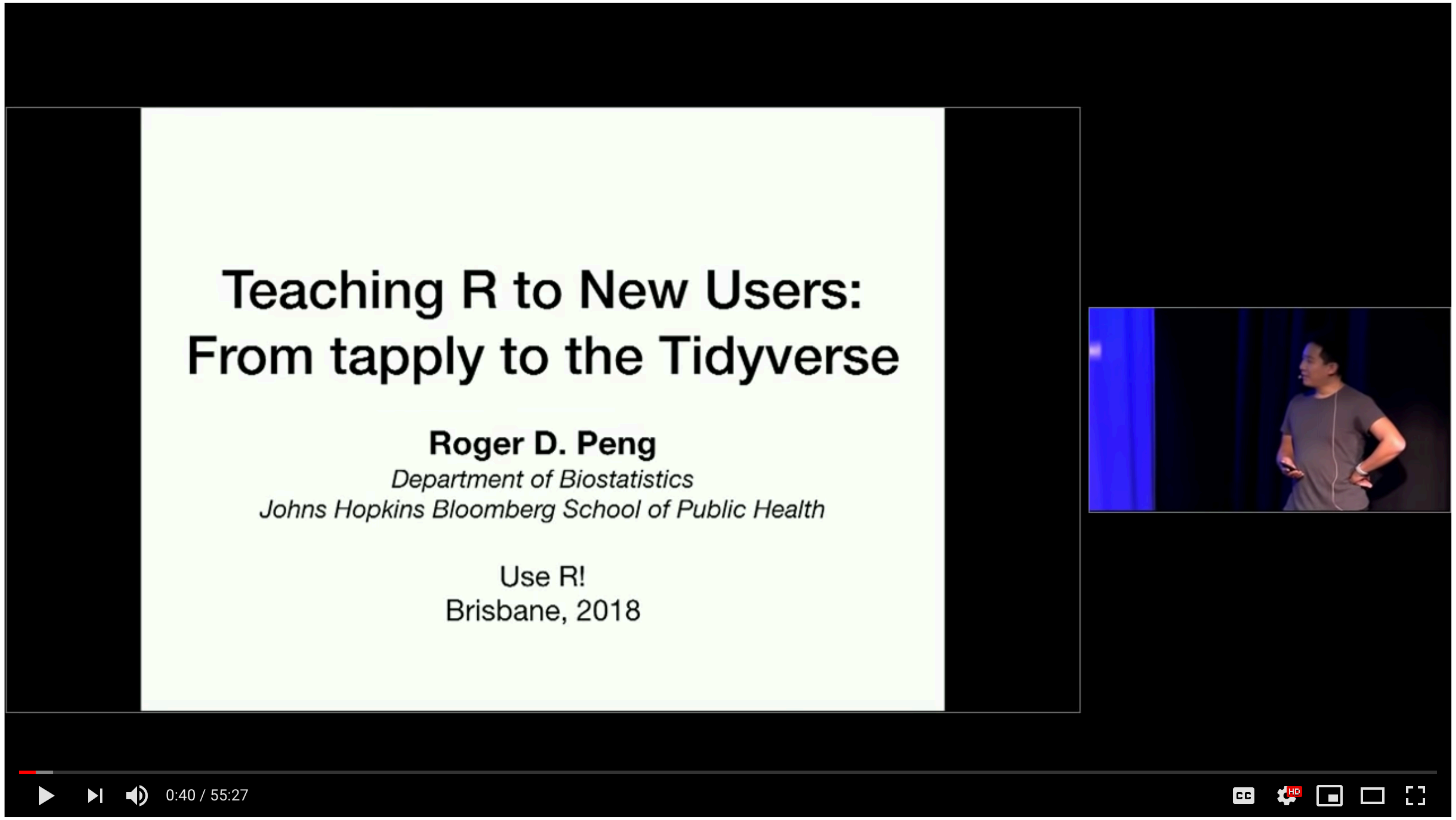
S

Written by John Chambers

Originally developed in 1975 as an alternative to FORTRAN for statistical computing

“Books”

@rabaath



Teaching R to New Users:
From tapply to the Tidyverse

Roger D. Peng

Department of Biostatistics
Johns Hopkins Bloomberg School of Public Health

Use R!
Brisbane, 2018

Roger Peng's useR 2018 keynote,
Teaching R to New Users: From tapply to Tidyverse (video, blogpost)

"The ambiguity [of the S language] is real and goes to a key objective: 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."
Quoted in Roger Peng's keynote

McNamara, Amelia Ahlers. (2015).

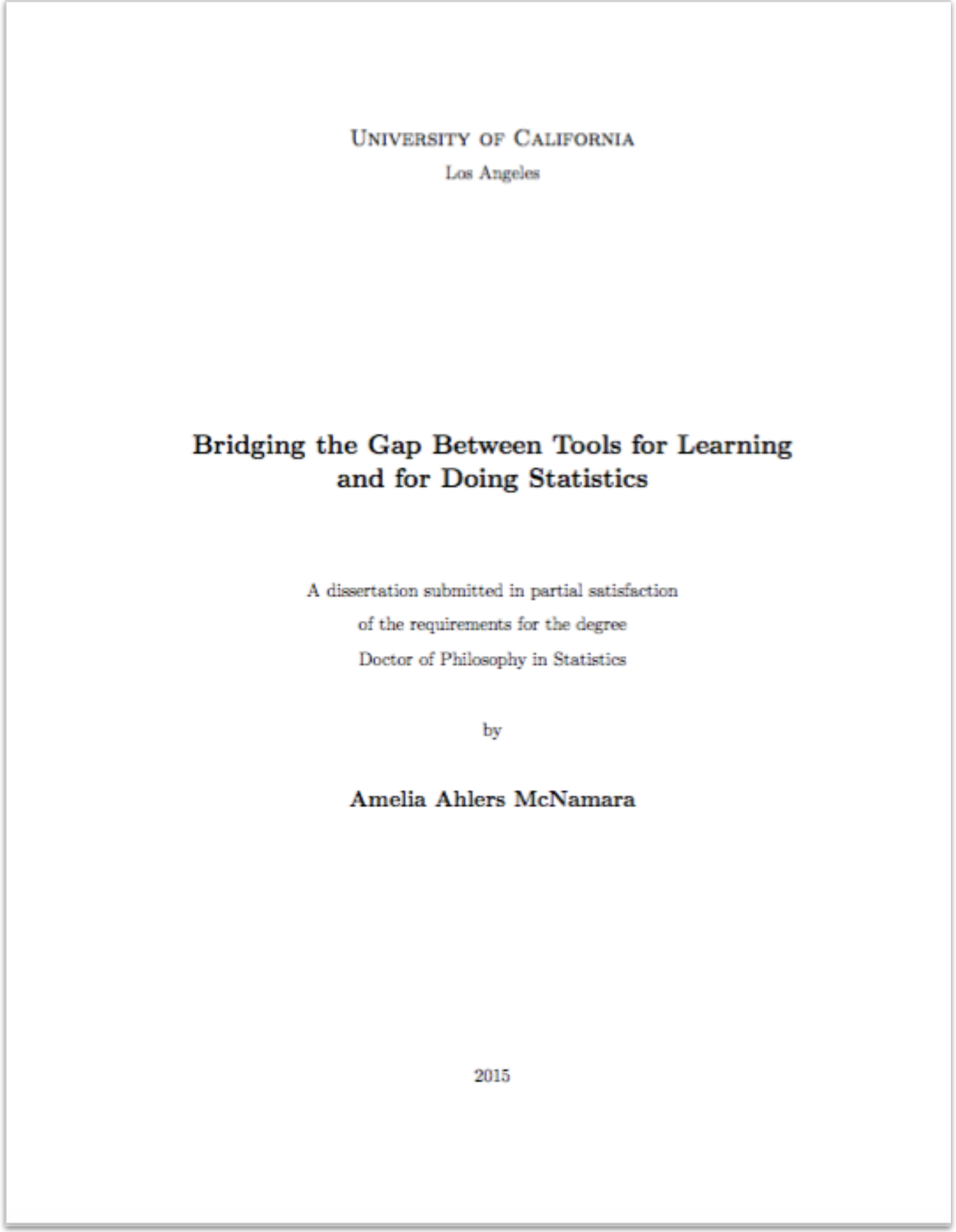
Bridging the Gap Between Tools for Learning and for Doing Statistics.

http://bit.ly/BridgingTheToolGap

or, two arXiv pre-prints:

http://bit.ly/ModernStatComputing

http://bit.ly/StateOfStatCompEd



UNIVERSITY OF CALIFORNIA
Los Angeles

Bridging the Gap Between Tools for Learning and for Doing Statistics

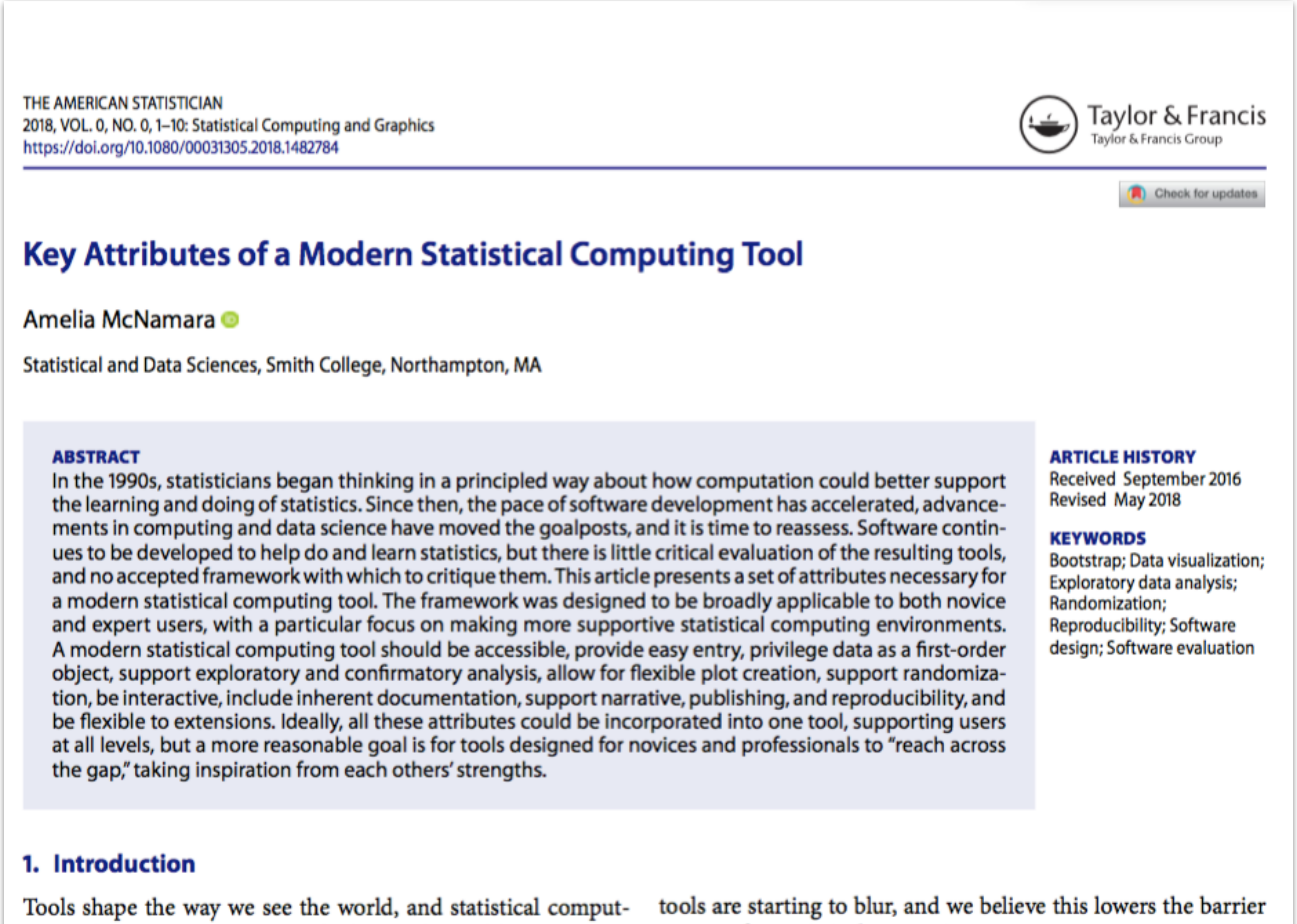
A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Statistics

by

Amelia Ahlers McNamara

2015

http://bit.ly/KeyAttributesPreprint



THE AMERICAN STATISTICIAN
2018, VOL. 0, NO. 0, 1–10: Statistical Computing and Graphics
https://doi.org/10.1080/00031305.2018.1482784

Key Attributes of a Modern Statistical Computing Tool

Amelia McNamara

Statistical and Data Sciences, Smith College, Northampton, MA

ABSTRACT
In the 1990s, statisticians began thinking in a principled way about how computation could better support the learning and doing of statistics. Since then, the pace of software development has accelerated, advancements in computing and data science have moved the goalposts, and it is time to reassess. Software continues to be developed to help do and learn statistics, but there is little critical evaluation of the resulting tools, and no accepted framework with which to critique them. This article presents a set of attributes necessary for a modern statistical computing tool. The framework was designed to be broadly applicable to both novice and expert users, with a particular focus on making more supportive statistical computing environments. A modern statistical computing tool should be accessible, provide easy entry, privilege data as a first-order object, support exploratory and confirmatory analysis, allow for flexible plot creation, support randomization, be interactive, include inherent documentation, support narrative, publishing, and reproducibility, and be flexible to extensions. Ideally, all these attributes could be incorporated into one tool, supporting users at all levels, but a more reasonable goal is for tools designed for novices and professionals to "reach across the gap," taking inspiration from each others' strengths.

ARTICLE HISTORY
Received September 2016
Revised May 2018

KEYWORDS
Bootstrap; Data visualization; Exploratory data analysis; Randomization; Reproducibility; Software design; Software evaluation

1. Introduction

Tools shape the way we see the world, and statistical comput-

tools are starting to blur, and we believe this lowers the barrier http://bit.ly/KeyAttributesPreprint

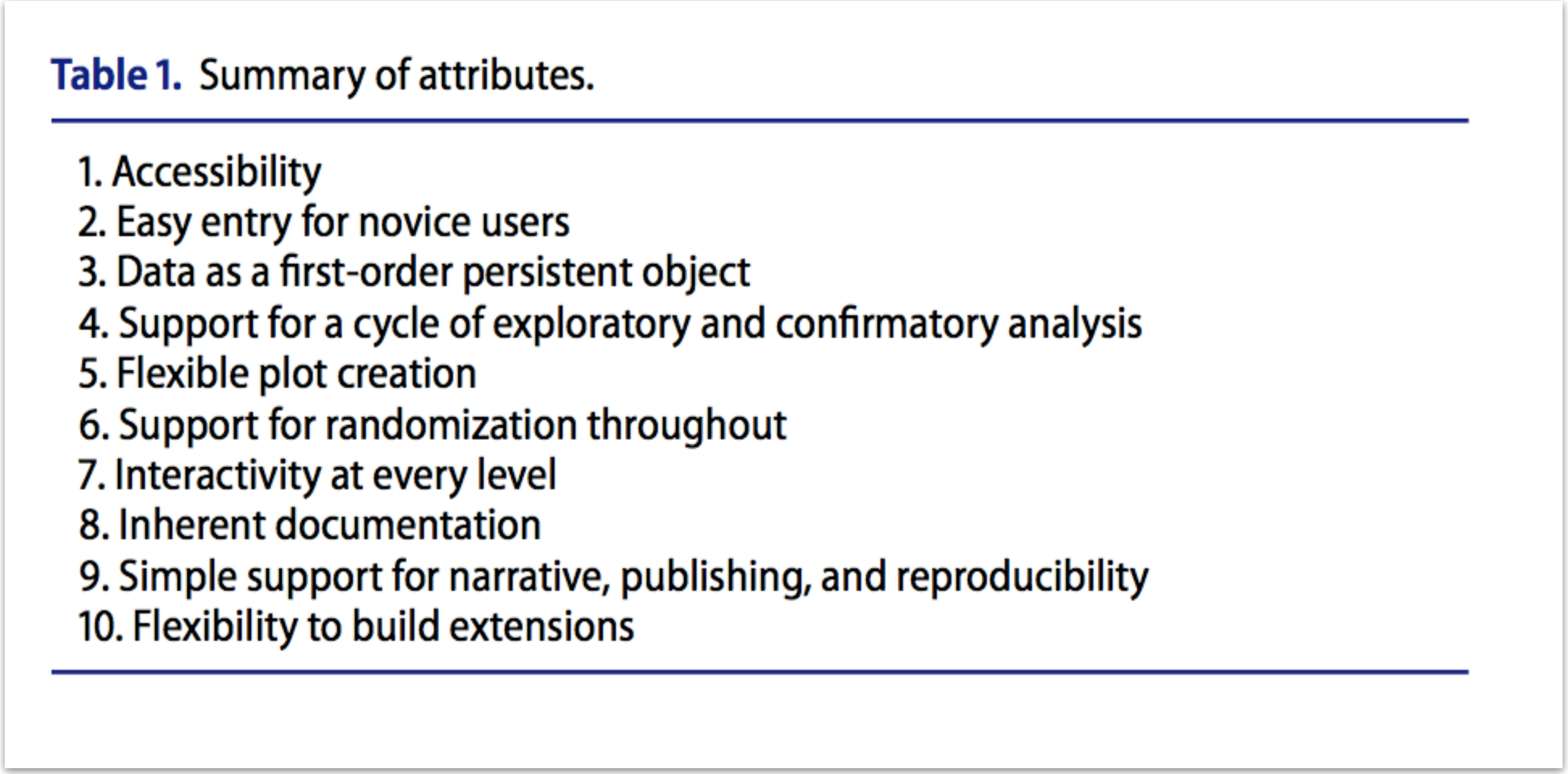
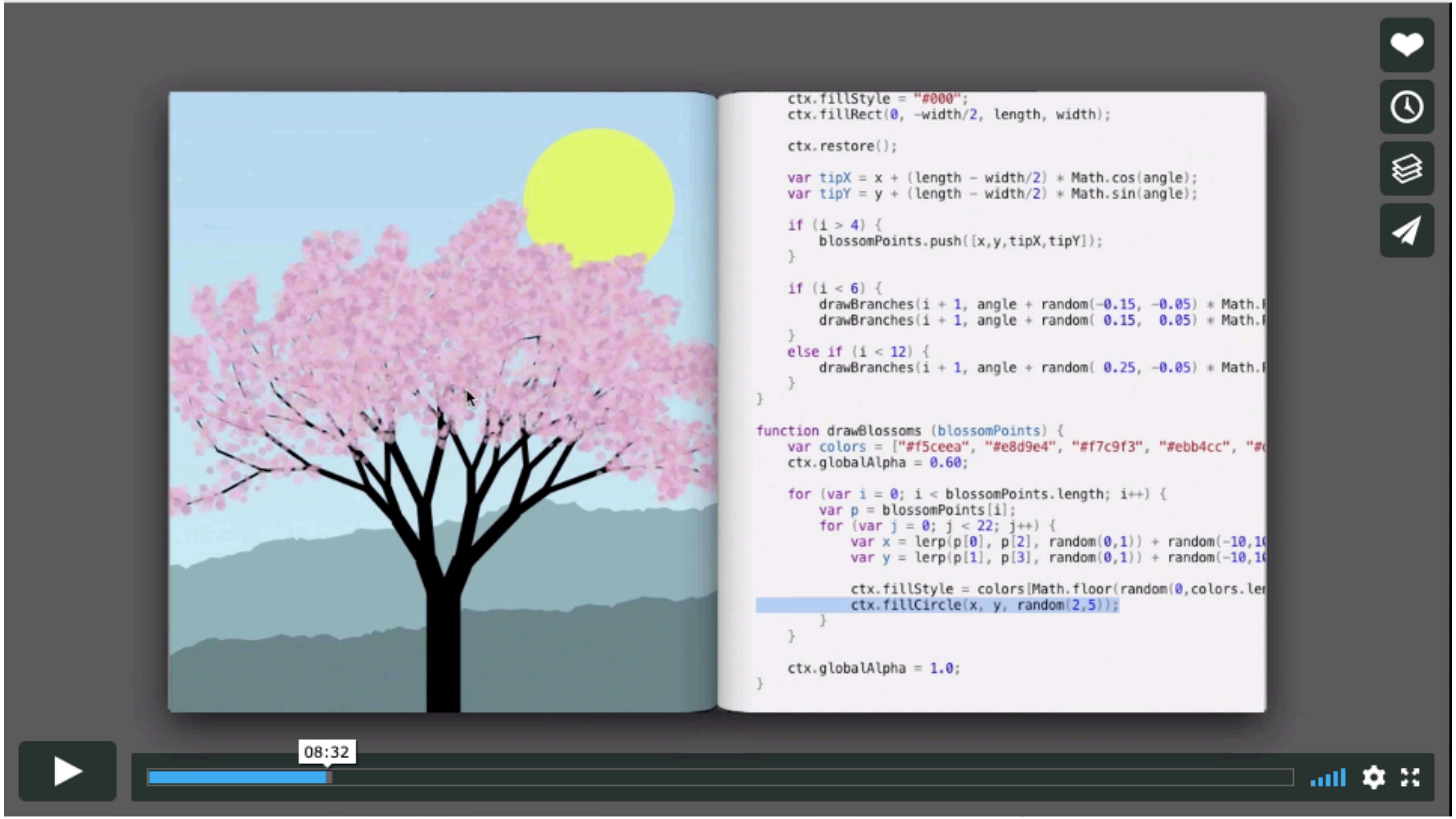


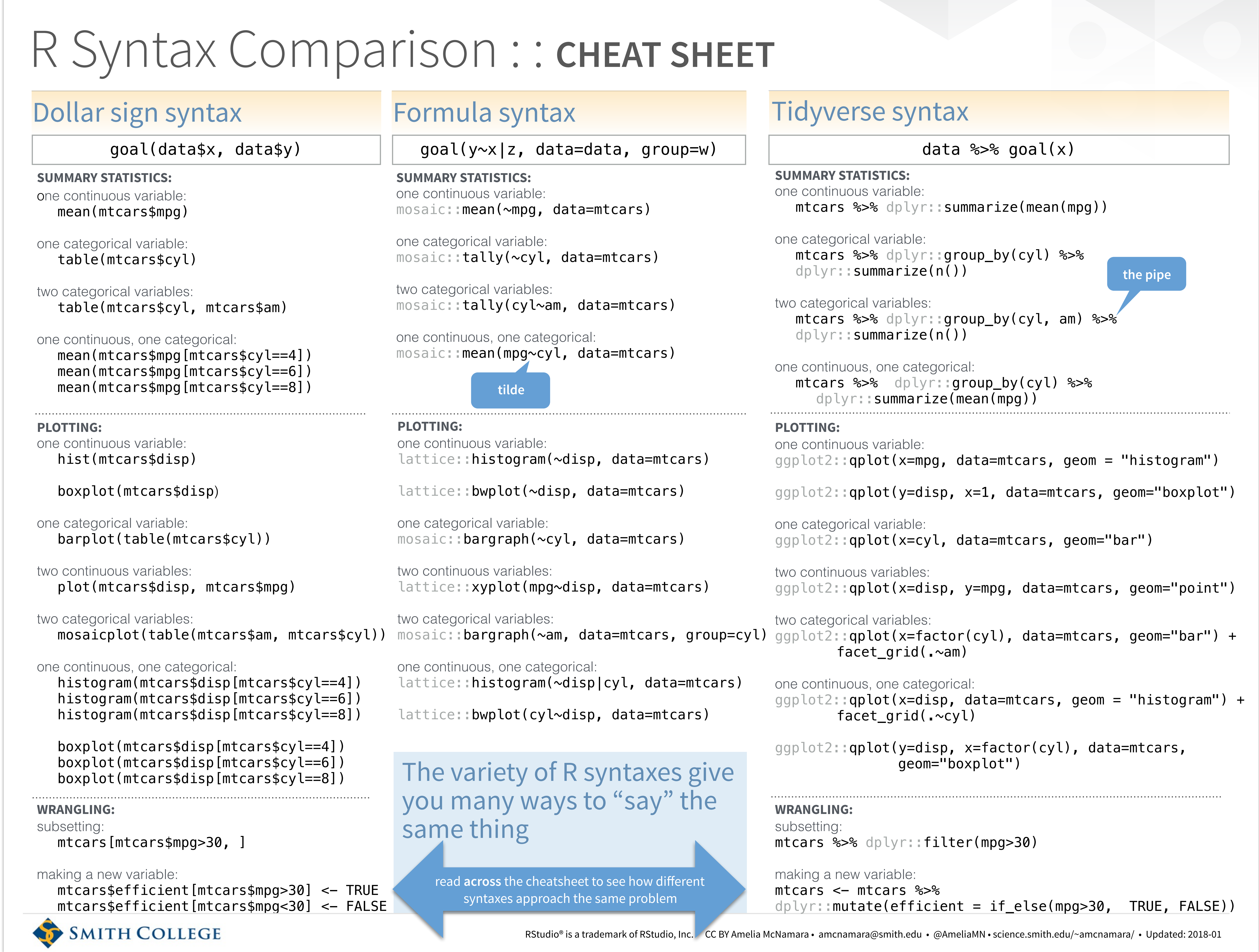
Table 1. Summary of attributes.

1. Accessibility
2. Easy entry for novice users
3. Data as a first-order persistent object
4. Support for a cycle of exploratory and confirmatory analysis
5. Flexible plot creation
6. Support for randomization throughout
7. Interactivity at every level
8. Inherent documentation
9. Simple support for narrative, publishing, and reproducibility
10. Flexibility to build extensions



```
    ctx.fillStyle = "#000";
    ctx.fillRect(0, -width/2, length, width);

    ctx.restore();

    var tipX = x + (length - width/2) * Math.cos(angle);
    var tipY = y + (length - width/2) * Math.sin(angle);

    if (i > 4) {
        blossomPoints.push([x,y,tipX,tipY]);
    }

    if (i < 6) {
        drawBranches(i + 1, angle + random(-0.15, -0.05) * Math.
        drawBranches(i + 1, angle + random( 0.15,  0.05) * Math.
    }
    else if (i < 12) {
        drawBranches(i + 1, angle + random( 0.25, -0.05) * Math.
    }
}

function drawBlossoms (blossomPoints) {
    var colors = ["#f5ceea", "#e8d9e4", "#f7c9f3", "#ebb4cc", "#
    ctx.globalAlpha = 0.60;

    for (var i = 0; i < blossomPoints.length; i++) {
        var p = blossomPoints[i];
        for (var j = 0; j < 22; j++) {
            var x = lerp(p[0], p[2], random(0,1)) + random(-10,1
            var y = lerp(p[1], p[3], random(0,1)) + random(-10,1

            ctx.fillStyle = colors[Math.floor(random(0,colors.le
            ctx.fillCircle(x, y, random(2,5));
        }
    }

    ctx.globalAlpha = 1.0;
}
```

Bret Victor - Inventing on Principle (video)



R Syntax Comparison : : CHEAT SHEET

Dollar sign syntax

```
goal(data$x, data$y)
```

SUMMARY STATISTICS:

one continuous variable:

```
   mean(mtcars$mpg)
```

one categorical variable:

```
   table(mtcars$cyl)
```

two categorical variables:

```
   table(mtcars$cyl, mtcars$am)
```

one continuous, one categorical:

```
   mean(mtcars$mpg[mtcars$cyl==4])
   mean(mtcars$mpg[mtcars$cyl==6])
   mean(mtcars$mpg[mtcars$cyl==8])
```

PLOTTING:

one continuous variable:

```
   hist(mtcars$disp)

   boxplot(mtcars$disp)
```

one categorical variable:

```
   barplot(table(mtcars$cyl))
```

two continuous variables:

```
   plot(mtcars$disp, mtcars$mpg)
```

two categorical variables:

```
   mosaicplot(table(mtcars$am, mtcars$cyl))
```

one continuous, one categorical:

```
   histogram(mtcars$disp[mtcars$cyl==4])
   histogram(mtcars$disp[mtcars$cyl==6])
   histogram(mtcars$disp[mtcars$cyl==8])

   boxplot(mtcars$disp[mtcars$cyl==4])
   boxplot(mtcars$disp[mtcars$cyl==6])
   boxplot(mtcars$disp[mtcars$cyl==8])
```

WRANGLING:

subsetting:

```
   mtcars[mtcars$mpg>30, ]
```

making a new variable:

```
   mtcars$efficient[mtcars$mpg>30] <- TRUE
   mtcars$efficient[mtcars$mpg<30] <- FALSE
```

Formula syntax

```
goal(y~x|z, data=data, group=w)
```

SUMMARY STATISTICS:

one continuous variable:

```
mosaic::mean(~mpg, data=mtcars)
```

one categorical variable:

```
mosaic::tally(~cyl, data=mtcars)
```

two categorical variables:

```
mosaic::tally(cyl~am, data=mtcars)
```

one continuous, one categorical:

```
mosaic::mean(mpg~cyl, data=mtcars)
```

tilde

PLOTTING:

one continuous variable:

```
lattice::histogram(~disp, data=mtcars)

lattice::bwplot(~disp, data=mtcars)
```

one categorical variable:

```
mosaic::bargraph(~cyl, data=mtcars)
```

two continuous variables:

```
lattice::xyplot(mpg~disp, data=mtcars)
```

two categorical variables:

```
mosaic::bargraph(~am, data=mtcars, group=cyl)
```

one continuous, one categorical:

```
lattice::histogram(~disp|cyl, data=mtcars)

lattice::bwplot(cyl~disp, data=mtcars)
```

The variety of R syntaxes give you many ways to “say” the same thing

read **across** the cheatsheet to see how different syntaxes approach the same problem

Tidyverse syntax

```
data %>% goal(x)
```

SUMMARY STATISTICS:

one continuous variable:

```
   mtcars %>% dplyr::summarize(mean(mpg))
```

one categorical variable:

```
   mtcars %>% dplyr::group_by(cyl) %>%
   dplyr::summarize(n())
```

the pipe

two categorical variables:

```
   mtcars %>% dplyr::group_by(cyl, am) %>%
   dplyr::summarize(n())
```

one continuous, one categorical:

```
   mtcars %>%  dplyr::group_by(cyl) %>%
     dplyr::summarize(mean(mpg))
```

PLOTTING:

one continuous variable:

```
ggplot2::qplot(x=mpg, data=mtcars, geom = "histogram")

ggplot2::qplot(y=disp, x=1, data=mtcars, geom="boxplot")
```

one categorical variable:

```
ggplot2::qplot(x=cyl, data=mtcars, geom="bar")
```

two continuous variables:

```
ggplot2::qplot(x=disp, y=mpg, data=mtcars, geom="point")
```

two categorical variables:

```
ggplot2::qplot(x=factor(cyl), data=mtcars, geom="bar") +
        facet_grid(.~am)
```

one continuous, one categorical:

```
ggplot2::qplot(x=disp, data=mtcars, geom = "histogram") +
        facet_grid(.~cyl)

ggplot2::qplot(y=disp, x=factor(cyl), data=mtcars,
            geom="boxplot")
```

WRANGLING:

subsetting:

```
mtcars %>% dplyr::filter(mpg>30)
```

making a new variable:

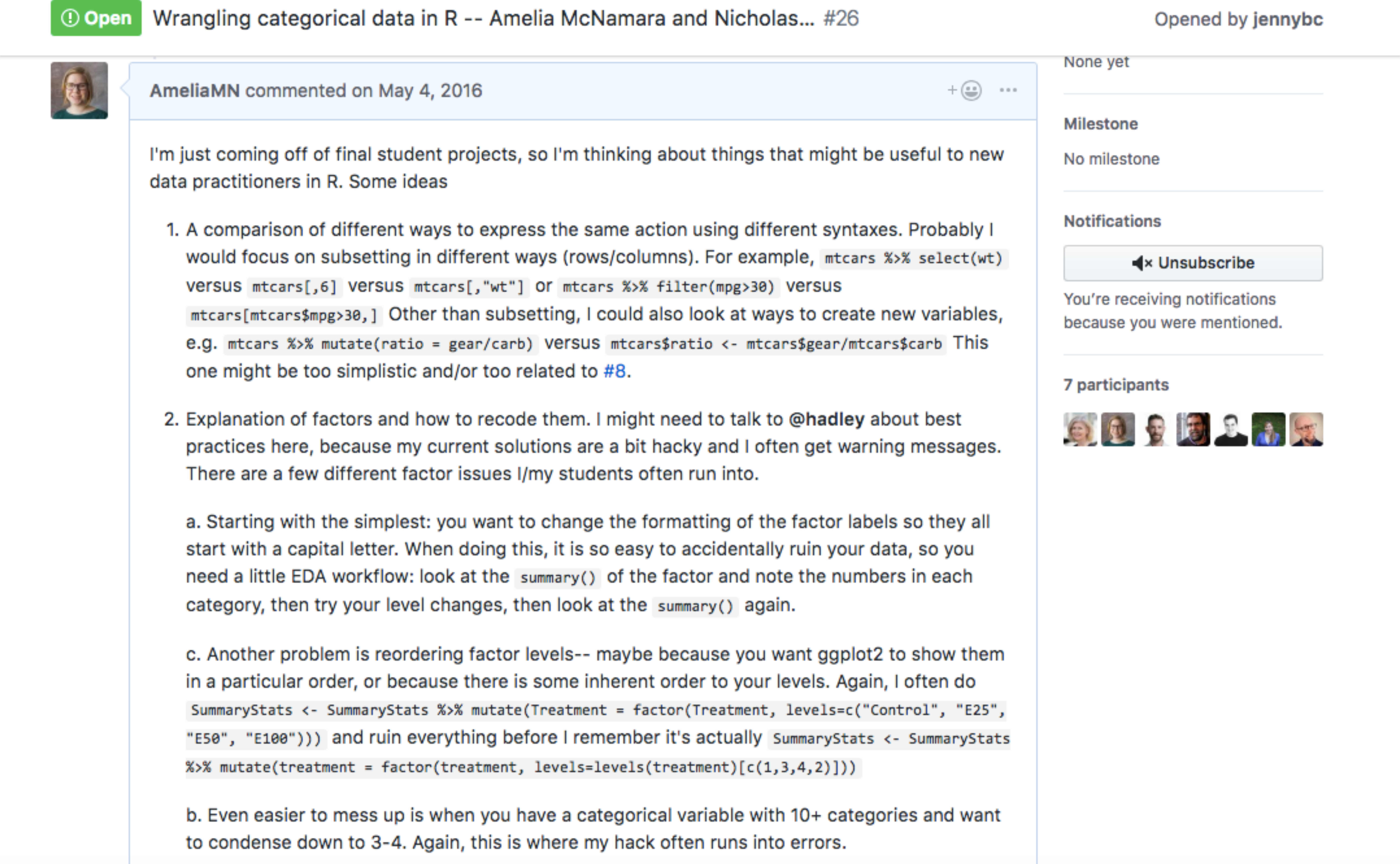
```
mtcars <- mtcars %>%
dplyr::mutate(efficient = if_else(mpg>30,  TRUE, FALSE))
```

SMITH COLLEGE

RStudio® is a trademark of RStudio, Inc. • CC BY Amelia McNamara • amcnamara@smith.edu • @AmeliaMN • science.smith.edu/~amcnamara/ • Updated: 2018-01

https://www.rstudio.com/resources/cheatsheets/

Safety



Wrangling categorical data in R -- Amelia McNamara and Nicholas... #26

Opened by **jennybc**

AmeliaMN commented on May 4, 2016

I'm just coming off of final student projects, so I'm thinking about things that might be useful to new data practitioners in R. Some ideas

1. A comparison of different ways to express the same action using different syntaxes. Probably I would focus on subsetting in different ways (rows/columns). For example, `mtcars %>% select(wt)` versus `mtcars[,6]` versus `mtcars[,"wt"]` or `mtcars %>% filter(mpg>30)` versus `mtcars[mtcars$mpg>30,]` Other than subsetting, I could also look at ways to create new variables, e.g. `mtcars %>% mutate(ratio = gear/carb)` versus `mtcars$ratio <- mtcars$gear/mtcars$carb` This one might be too simplistic and/or too related to #8.

2. Explanation of factors and how to recode them. I might need to talk to **@hadley** about best practices here, because my current solutions are a bit hacky and I often get warning messages. There are a few different factor issues I/my students often run into.

   a. Starting with the simplest: you want to change the formatting of the factor labels so they all start with a capital letter. When doing this, it is so easy to accidentally ruin your data, so you need a little EDA workflow: look at the `summary()` of the factor and note the numbers in each category, then try your level changes, then look at the `summary()` again.

   c. Another problem is reordering factor levels-- maybe because you want ggplot2 to show them in a particular order, or because there is some inherent order to your levels. Again, I often do `SummaryStats <- SummaryStats %>% mutate(Treatment = factor(Treatment, levels=c("Control", "E25", "E50", "E100")))` and ruin everything before I remember it's actually `SummaryStats <- SummaryStats %>% mutate(treatment = factor(treatment, levels=levels(treatment)[c(1,3,4,2)]))`

   b. Even easier to mess up is when you have a categorical variable with 10+ categories and want to condense down to 3-4. Again, this is where my hack often runs into errors.

None yet

Milestone

No milestone

Notifications

You're receiving notifications because you were mentioned.

7 participants

NOT PEER-REVIEWED

"*PeerJ Preprints*" is a venue for early communication or feedback before peer review. Data may be preliminary.

Learn more about preprints or browse peer-reviewed articles instead.

Wrangling categorical data in R

Me

@askdrstats

Amelia McNamara[1], Nicholas J Horton[2]

August 30, 2017

http://bit.ly/WranglingCats

Highlighted in Practical Data Science for Stats

Data wrangling is a critical foundation of data science, and wrangling of categorical data is an important component of this process. However,

I published in PeerJ and it i
very fast, has good editors
has consistently given goo
quality and rigorous review
of my work, and produces
visually appealing
manuscripts.

Matthew Jacks
PeerJ aut

Visitors	5,332
Views	6,125

Taylor & Francis Online
Log in
Registe
Enter keywords, authors, DOI etc.
Check for updates
Full Article
Figures & data
References
Supplemental
Citations
Metrics
Reprints & Permissions
Get access

Journal
The American Statistician ›
Volume 72, 2018 - Issue 1: Special Issue on Data Science

907
Views

0
CrossRef citations to date

8
Altmetric

Article

Wrangling Categorical Data in R

Amelia McNamara & Nicholas J. Horton

Pages 97-104 | Received 01 May 2017, Accepted author version posted online: 27 Jul 2017, Published online: 27 Jul 2017

Download citation https://doi.org/10.1080/00031305.2017.1356375

Amelia McNamara[a*] **http://orcid.org/0000-0003-4916-2433 & Nicholas J. Horton**[b] **http://orcid.org/0000-0003-3332-4311**

[a] Program in Statistical and Data Sciences, Smith College, Northampton, MA

[b] Department of Mathematics and Statistics, Amherst College, Amherst, MA

CONTACT Amelia McNamara *amcnamara@smith.edu* Program in Statistical and Data Sciences,

People also read

Article

Data Organizati
in Spreadsheets

```
> badApproach <- GSS$OpinionOfIncome
> summary(badApproach)
    Above average           Average     Below average        Don't know Far above average 
              483              1118               666                21                65 
Far below average         No answer              NA's 
              179                 6                 2 
> levels(badApproach) <- c("Far above average", "Above average",
+                          "Average", "Below Average", "Far below average",
+                          "Don't know", "No answer")
> summary(badApproach)
Far above average     Above average           Average     Below Average Far below average 
              483              1118               666                21                65 
       Don't know         No answer              NA's 
              179                 6                 2 
> 
```

```
> summary(GSS$BaseOpinionOfIncome)
    Above average           Average     Below average        Don't know Far above average 
              483              1118               666                21                65 
Far below average         No answer              NA's 
              179                 6                 2 
> GSS$BaseOpinionOfIncome <-
+   factor(GSS$BaseOpinionOfIncome,
+          levels = c("Far above average", "Above average", "Average ", "Below Average",
+                     "Far below average", "Don't know", "No answer"))
> summary(GSS$BaseOpinionOfIncome)
Far above average     Above average          Average      Below Average Far below average 
               65               483                 0                 0               179 
       Don't know         No answer              NA's 
               21                 6              1786 
> 
```

```
> library(forcats)
> summary(GSS$OpinionOfIncome)
    Above average           Average     Below average        Don't know Far above average 
              483              1118               666                21                65 
Far below average         No answer              NA's 
              179                 6                 2 
> GSS <- GSS %>%
+   mutate(tidyOpinionOfIncome =
+            fct_relevel(OpinionOfIncome,
+                        "Far above average",
+                        "Above average",
+                        "Average",
+                        "Below average",
+                        "Far below average"))
> summary(GSS$tidyOpinionOfIncome)
Far above average     Above average           Average     Below average Far below average 
               65               483              1118               666               179 
       Don't know         No answer              NA's 
               21                 6                 2 
> 
```

Working with categorical data in R without losing your mind

Amelia McNamara @AmeliaMN

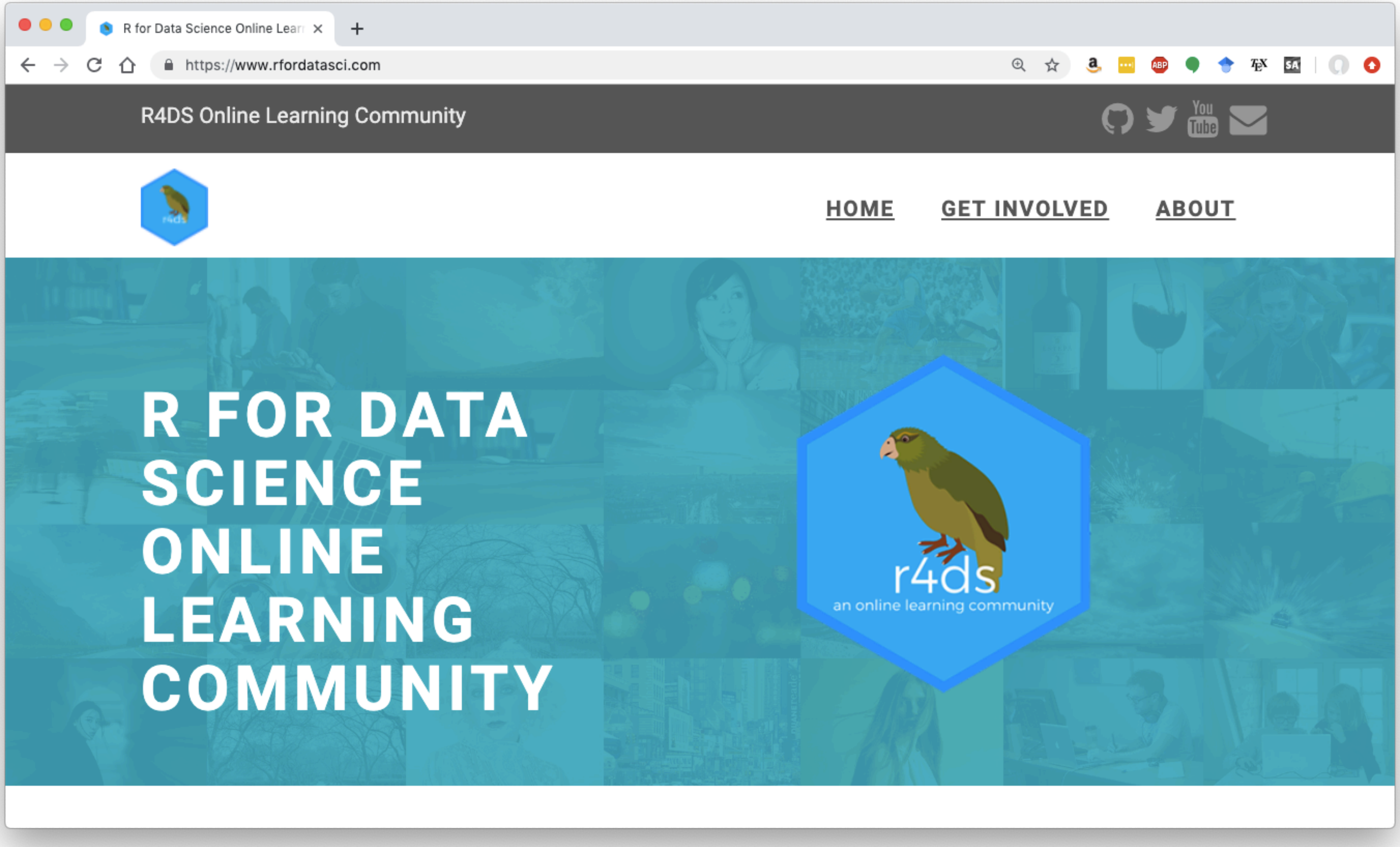
www.amelia.mn

University of St Thomas Department of Computer and Info

ickr: allenjaelee

Amelia McNamara,
"Working with categorical data in R without losing your mind" (video, slides)

Community



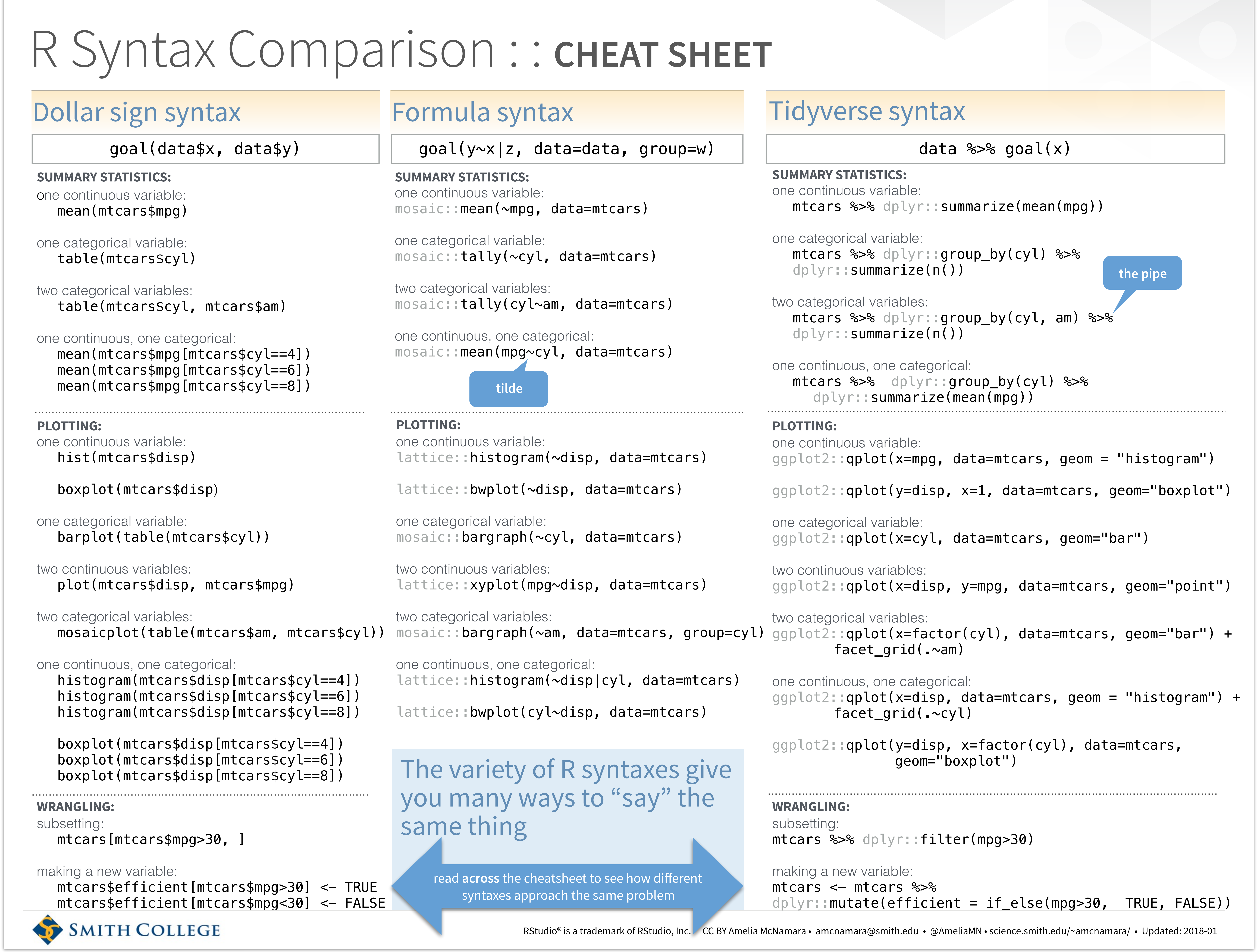
R4DS Online Learning Community

HOME GET INVOLVED ABOUT

R FOR DATA SCIENCE ONLINE LEARNING COMMUNITY

R for Data Science Online Lear
https://www.rfordatasci.com
r4ds
an online learning community

R for Data Science Online Learning Community
(r4ds book, community)



R Syntax Comparison : : CHEAT SHEET

Dollar sign syntax

```
goal(data$x, data$y)
```

SUMMARY STATISTICS:

one continuous variable:
```
   mean(mtcars$mpg)
```

one categorical variable:
```
   table(mtcars$cyl)
```

two categorical variables:
```
   table(mtcars$cyl, mtcars$am)
```

one continuous, one categorical:
```
   mean(mtcars$mpg[mtcars$cyl==4])
   mean(mtcars$mpg[mtcars$cyl==6])
   mean(mtcars$mpg[mtcars$cyl==8])
```

PLOTTING:

one continuous variable:
```
   hist(mtcars$disp)

   boxplot(mtcars$disp)
```

one categorical variable:
```
   barplot(table(mtcars$cyl))
```

two continuous variables:
```
   plot(mtcars$disp, mtcars$mpg)
```

two categorical variables:
```
   mosaicplot(table(mtcars$am, mtcars$cyl))
```

one continuous, one categorical:
```
   histogram(mtcars$disp[mtcars$cyl==4])
   histogram(mtcars$disp[mtcars$cyl==6])
   histogram(mtcars$disp[mtcars$cyl==8])

   boxplot(mtcars$disp[mtcars$cyl==4])
   boxplot(mtcars$disp[mtcars$cyl==6])
   boxplot(mtcars$disp[mtcars$cyl==8])
```

WRANGLING:

subsetting:
```
   mtcars[mtcars$mpg>30, ]
```

making a new variable:
```
   mtcars$efficient[mtcars$mpg>30] <- TRUE
   mtcars$efficient[mtcars$mpg<30] <- FALSE
```

Formula syntax

```
goal(y~x|z, data=data, group=w)
```

SUMMARY STATISTICS:

one continuous variable:
```
mosaic::mean(~mpg, data=mtcars)
```

one categorical variable:
```
mosaic::tally(~cyl, data=mtcars)
```

two categorical variables:
```
mosaic::tally(cyl~am, data=mtcars)
```

one continuous, one categorical:
```
mosaic::mean(mpg~cyl, data=mtcars)
```

tilde

PLOTTING:

one continuous variable:
```
lattice::histogram(~disp, data=mtcars)

lattice::bwplot(~disp, data=mtcars)
```

one categorical variable:
```
mosaic::bargraph(~cyl, data=mtcars)
```

two continuous variables:
```
lattice::xyplot(mpg~disp, data=mtcars)
```

two categorical variables:
```
mosaic::bargraph(~am, data=mtcars, group=cyl)
```

one continuous, one categorical:
```
lattice::histogram(~disp|cyl, data=mtcars)

lattice::bwplot(cyl~disp, data=mtcars)
```

The variety of R syntaxes give you many ways to “say” the same thing

read **across** the cheatsheet to see how different syntaxes approach the same problem

Tidyverse syntax

```
data %>% goal(x)
```

SUMMARY STATISTICS:

one continuous variable:
```
   mtcars %>% dplyr::summarize(mean(mpg))
```

one categorical variable:
```
   mtcars %>% dplyr::group_by(cyl) %>%
   dplyr::summarize(n())
```

the pipe

two categorical variables:
```
   mtcars %>% dplyr::group_by(cyl, am) %>%
   dplyr::summarize(n())
```

one continuous, one categorical:
```
   mtcars %>%  dplyr::group_by(cyl) %>%
     dplyr::summarize(mean(mpg))
```

PLOTTING:

one continuous variable:
```
ggplot2::qplot(x=mpg, data=mtcars, geom = "histogram")

ggplot2::qplot(y=disp, x=1, data=mtcars, geom="boxplot")
```

one categorical variable:
```
ggplot2::qplot(x=cyl, data=mtcars, geom="bar")
```

two continuous variables:
```
ggplot2::qplot(x=disp, y=mpg, data=mtcars, geom="point")
```

two categorical variables:
```
ggplot2::qplot(x=factor(cyl), data=mtcars, geom="bar") +
       facet_grid(.~am)
```

one continuous, one categorical:
```
ggplot2::qplot(x=disp, data=mtcars, geom = "histogram") +
       facet_grid(.~cyl)

ggplot2::qplot(y=disp, x=factor(cyl), data=mtcars,
             geom="boxplot")
```

WRANGLING:

subsetting:
```
mtcars %>% dplyr::filter(mpg>30)
```

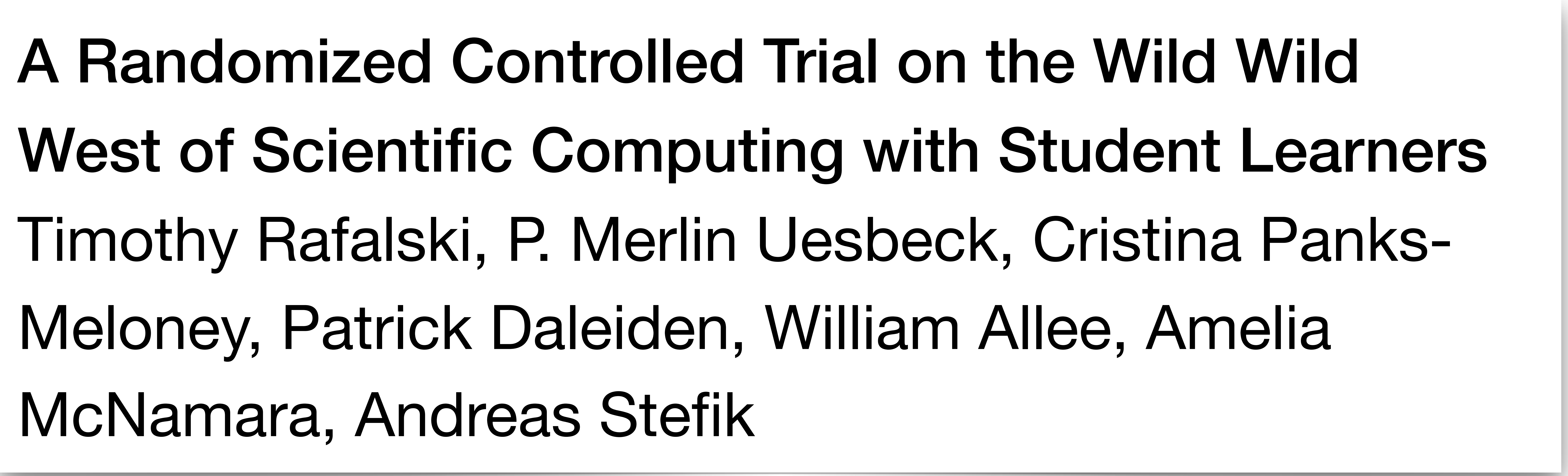
making a new variable:
```
mtcars <- mtcars %>%
dplyr::mutate(efficient = if_else(mpg>30,  TRUE, FALSE))
```

SMITH COLLEGE

RStudio® is a trademark of RStudio, Inc. • CC BY Amelia McNamara • amcnamara@smith.edu • @AmeliaMN • science.smith.edu/~amcnamara/ • Updated: 2018-01

https://www.rstudio.com/resources/cheatsheets/

Learnability?

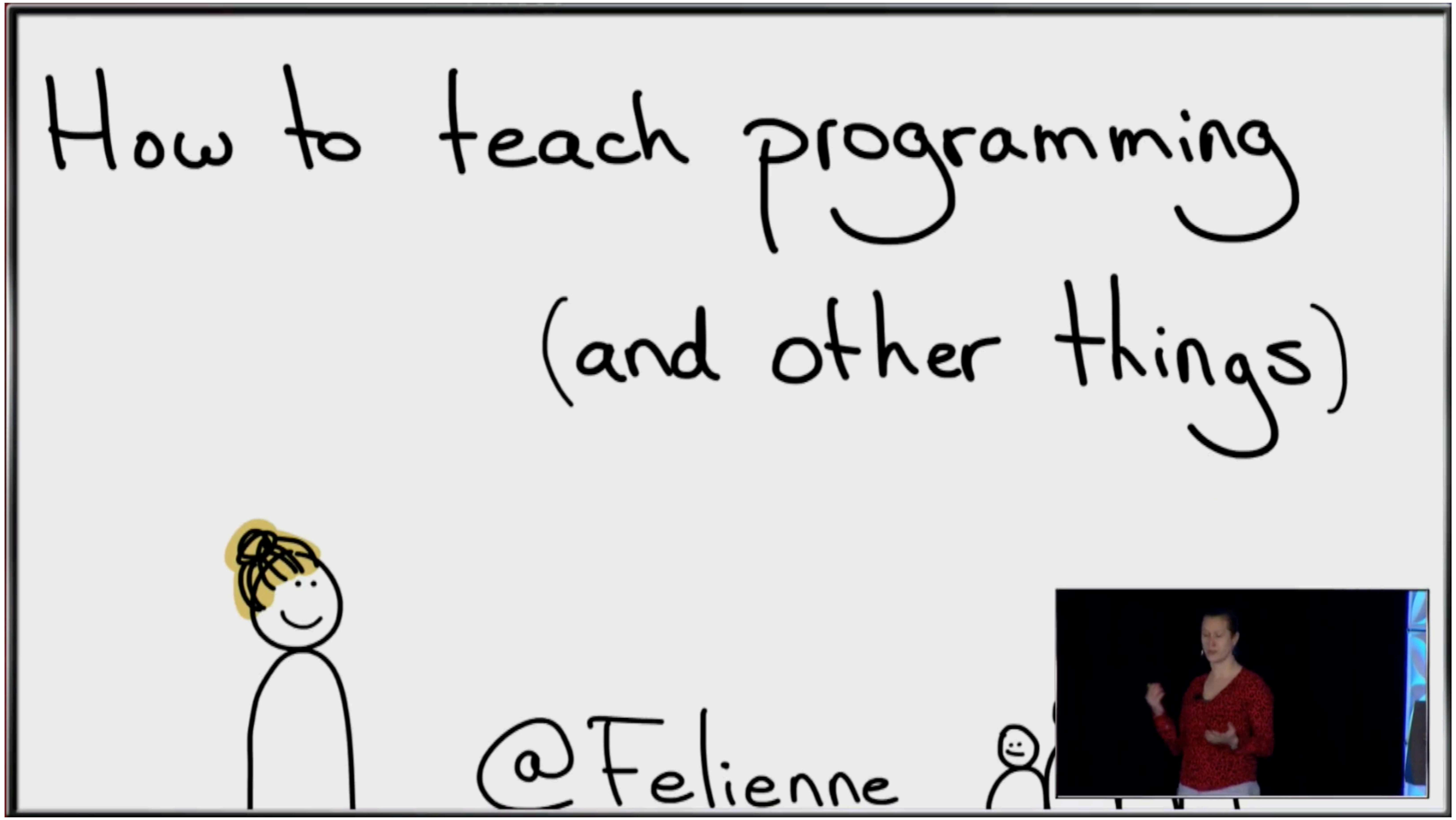


A Randomized Controlled Trial on the Wild Wild West of Scientific Computing with Student Learners

Timothy Rafalski, P. Merlin Uesbeck, Cristina Panks-Meloney, Patrick Daleiden, William Allee, Amelia McNamara, Andreas Stefik

tl;dr they (we?) didn't find a difference between base, formula, and tidyverse syntaxes in completion time or number of errors (qualitative or interpreter)

https://dl.acm.org/citation.cfm?id=3291279.3339421



Felienne’s rstudio::conf 2019 keynote,
Explicit Direct Instruction in Programming Education (video)

THE CARPENTRIES

software carpentry

DATA CARPENTRY

library carpentry

Teach the most useful thing first

Reduce cognitive load

Don’t touch the learner’s keyboard

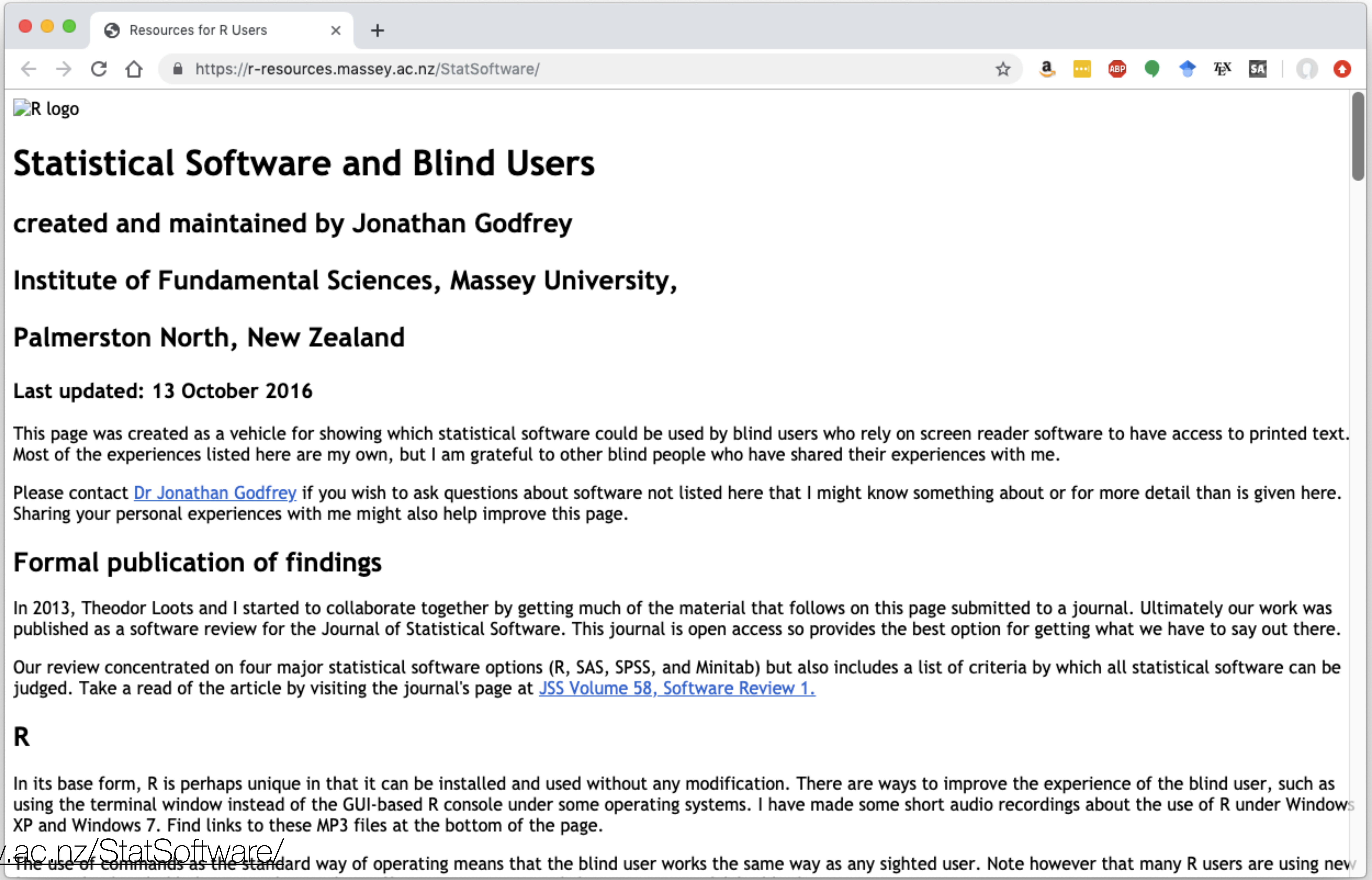
Verbalize what you are doing

Universal Design

“Universal Design is the design and composition of an environment so that it can be accessed, understood and used to the greatest extent possible by all people regardless of their age, size, ability or disability.”

- Centre for Excellence in Universal Design

tl;dr R is best, but could be better



R logo

Statistical Software and Blind Users

created and maintained by Jonathan Godfrey

Institute of Fundamental Sciences, Massey University,

Palmerston North, New Zealand

Last updated: 13 October 2016

This page was created as a vehicle for showing which statistical software could be used by blind users who rely on screen reader software to have access to printed text. Most of the experiences listed here are my own, but I am grateful to other blind people who have shared their experiences with me.

Please contact Dr Jonathan Godfrey if you wish to ask questions about software not listed here that I might know something about or for more detail than is given here. Sharing your personal experiences with me might also help improve this page.

Formal publication of findings

In 2013, Theodor Loots and I started to collaborate together by getting much of the material that follows on this page submitted to a journal. Ultimately our work was published as a software review for the Journal of Statistical Software. This journal is open access so provides the best option for getting what we have to say out there.

Our review concentrated on four major statistical software options (R, SAS, SPSS, and Minitab) but also includes a list of criteria by which all statistical software can be judged. Take a read of the article by visiting the journal's page at JSS Volume 58, Software Review 1.

R

In its base form, R is perhaps unique in that it can be installed and used without any modification. There are ways to improve the experience of the blind user, such as using the terminal window instead of the GUI-based R console under some operating systems. I have made some short audio recordings about the use of R under Windows XP and Windows 7. Find links to these MP3 files at the bottom of the page.

The use of commands as the standard way of operating means that the blind user works the same way as any sighted user. Note however that many R users are using new

https://r-resources.massey.ac.nz/StatSoftware/

THE PERL CONFERENCE 2019 IN PITTSBURGH

Perl Out Loud 🗣

1/48

Emily Shea
“Perl Out Loud”

Emily Shea - "Perl Out Loud" (video)

“Due to this disability, I cannot type or write by hand. Many people have asked me about the stack that enables me to be productive in spite of this limitation. I hope this information is helpful both for people with more severe limitations, and for programmers with mild repetitive stress injuries who can benefit from reducing their keyboard use.”

–Naomi Saphra, "What Does a Coder Do If They Can't Type?"

> “[…] by the time I was 20 I had developed a chronic wrist injury that I still have decades later. Maybe I should have walked away from computers, but by then I was hooked! That was a dark time, but eventually I struck upon a solution… I could dictate my code to someone else who could be my hands.”

–Ian Gilman, "Wrists & Apprentices"

Pair programming

https://www.youtube.com/watch?v=vgkahOzFH2Q

Readability

Summary statistics three ways

base

```
> mean(mtcars$mpg[mtcars$cyl==4])
[1] 26.66364
> mean(mtcars$mpg[mtcars$cyl==6])
[1] 19.74286
> mean(mtcars$mpg[mtcars$cyl==8])
[1] 15.1
```

mosaic

```
> mean(mpg~cyl, data=mtcars)
       4        6        8
26.66364 19.74286 15.10000
```

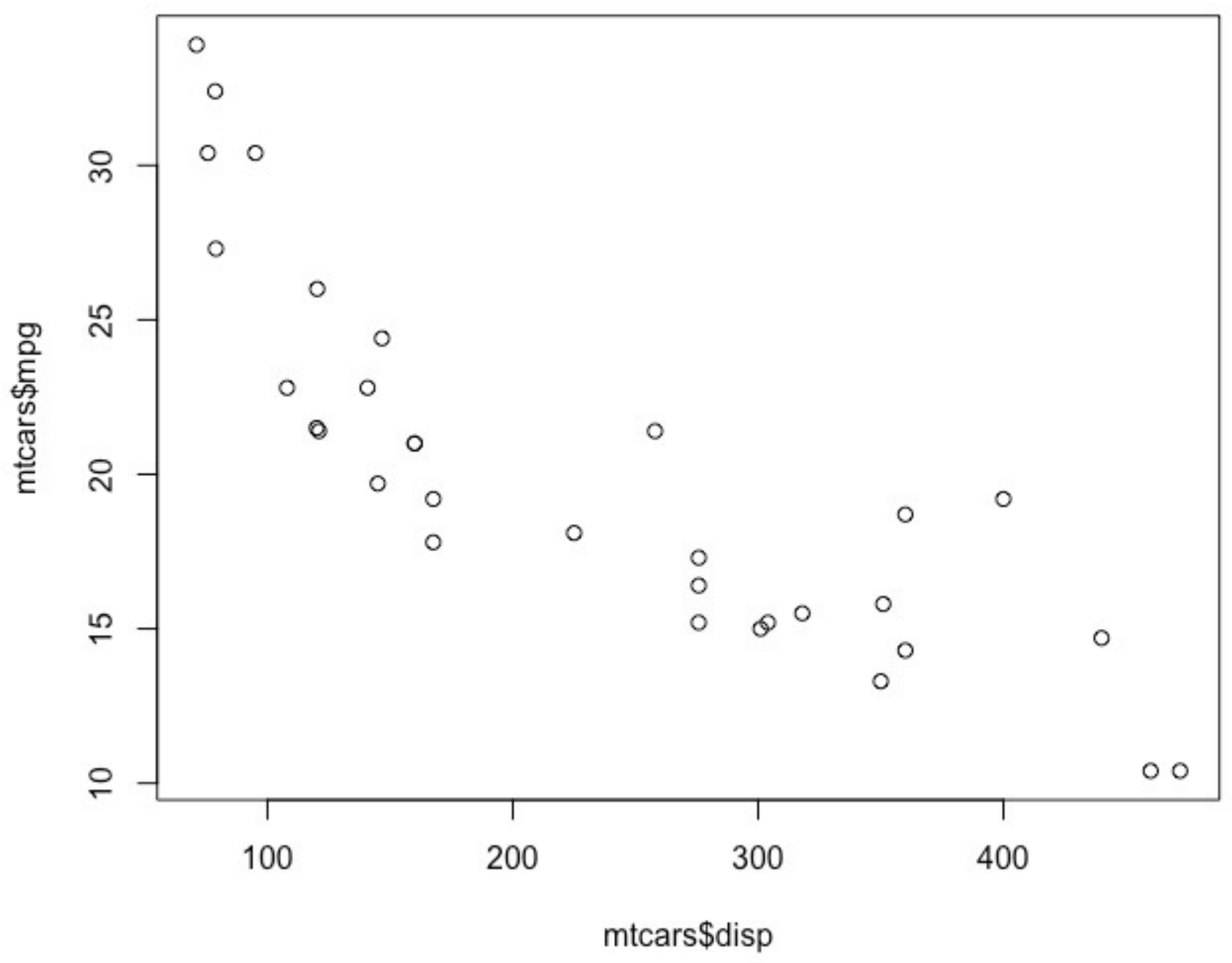
dplyr

```
> mtcars %>%
+   group_by(cyl) %>%
+   summarize(mean(mpg))
# A tibble: 3 x 2
    cyl `mean(mpg)`
  <dbl>       <dbl>
1     4        26.7
2     6        19.7
3     8        15.1
```

Scatterplot three ways

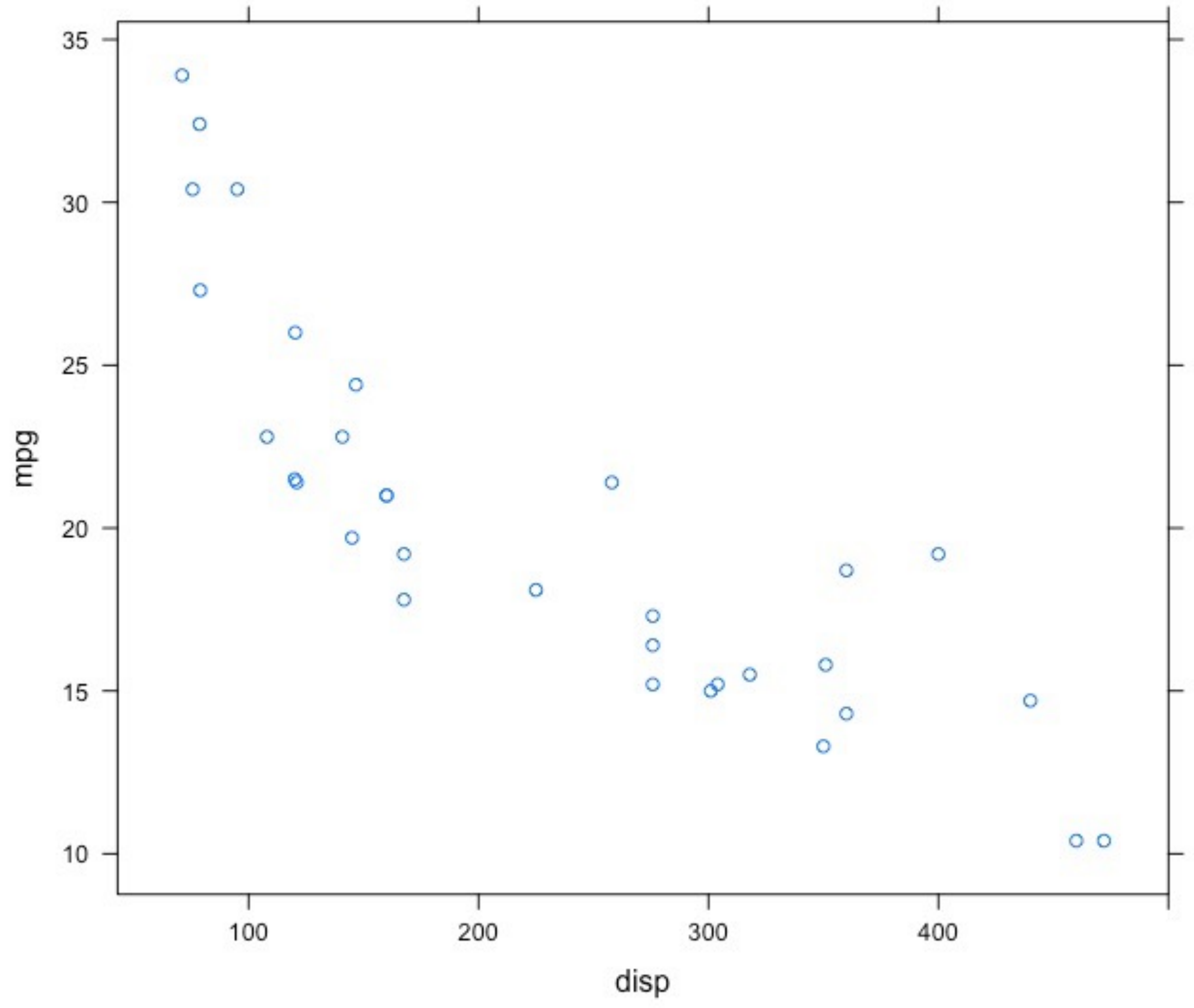
base

```
plot(mtcars$disp, mtcars$mpg)
```



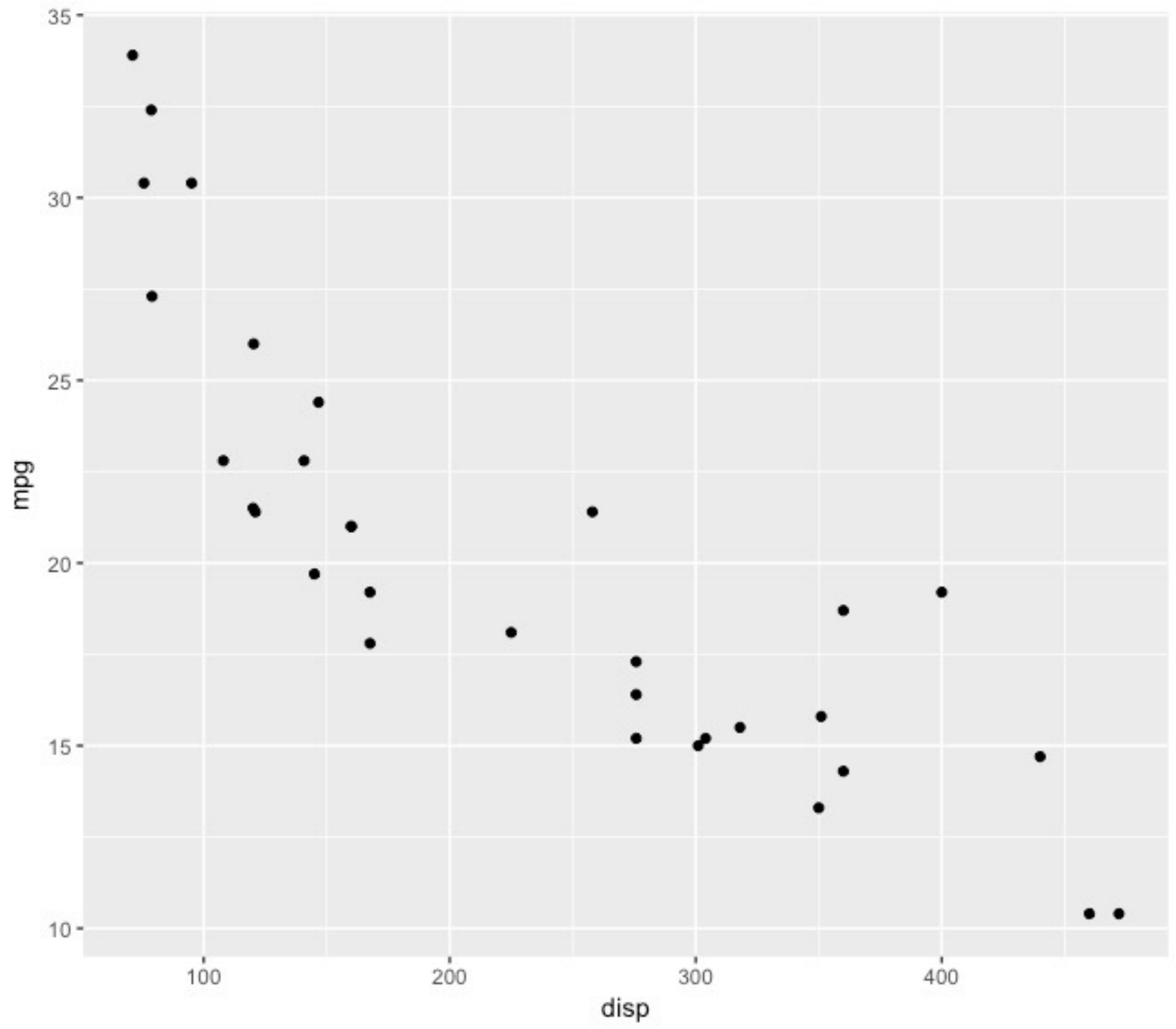
lattice

```
xyplot(mpg~disp, data=mtcars)
```



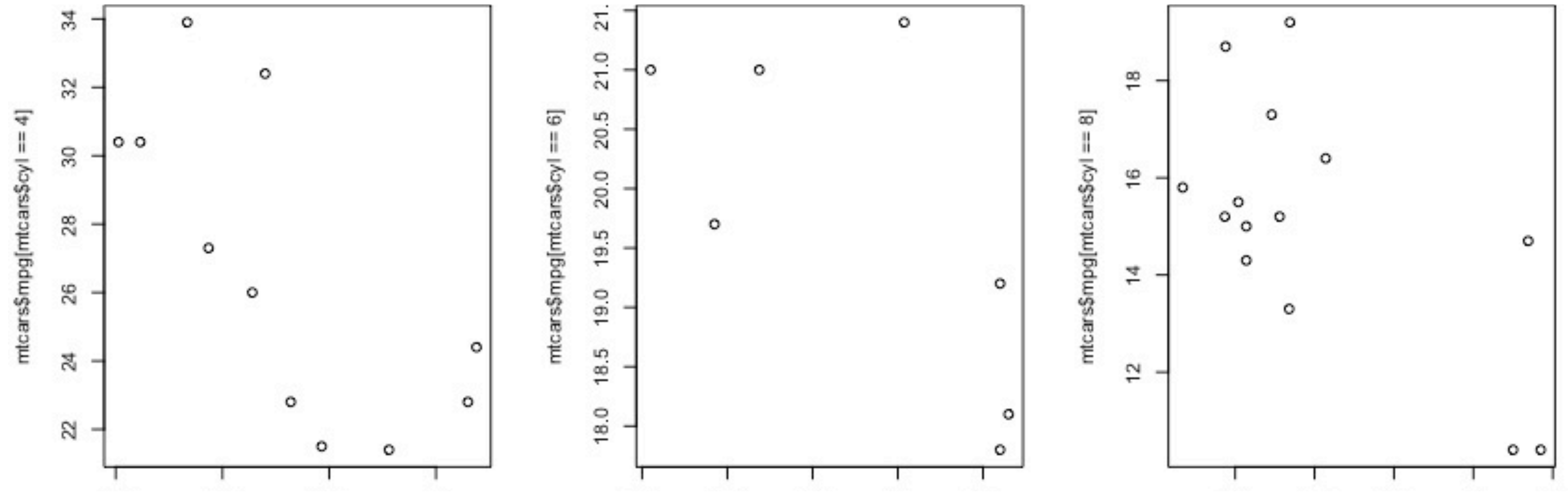
ggplot2

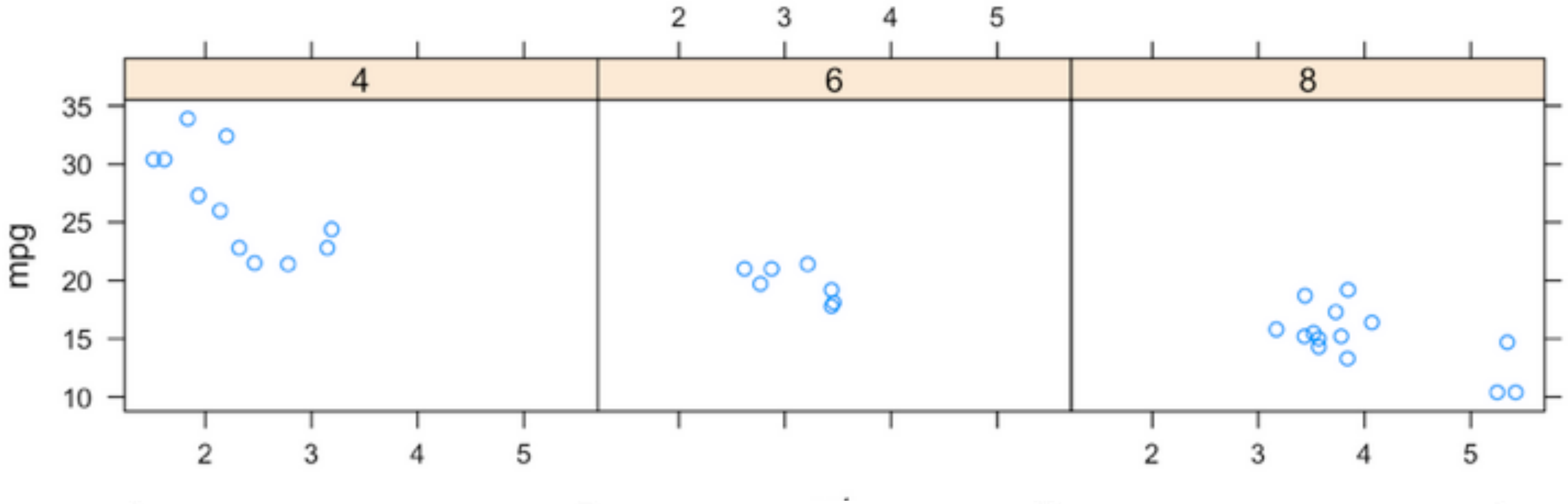
```
qplot(x=disp, y=mpg,
     data=mtcars, geom="point")
```

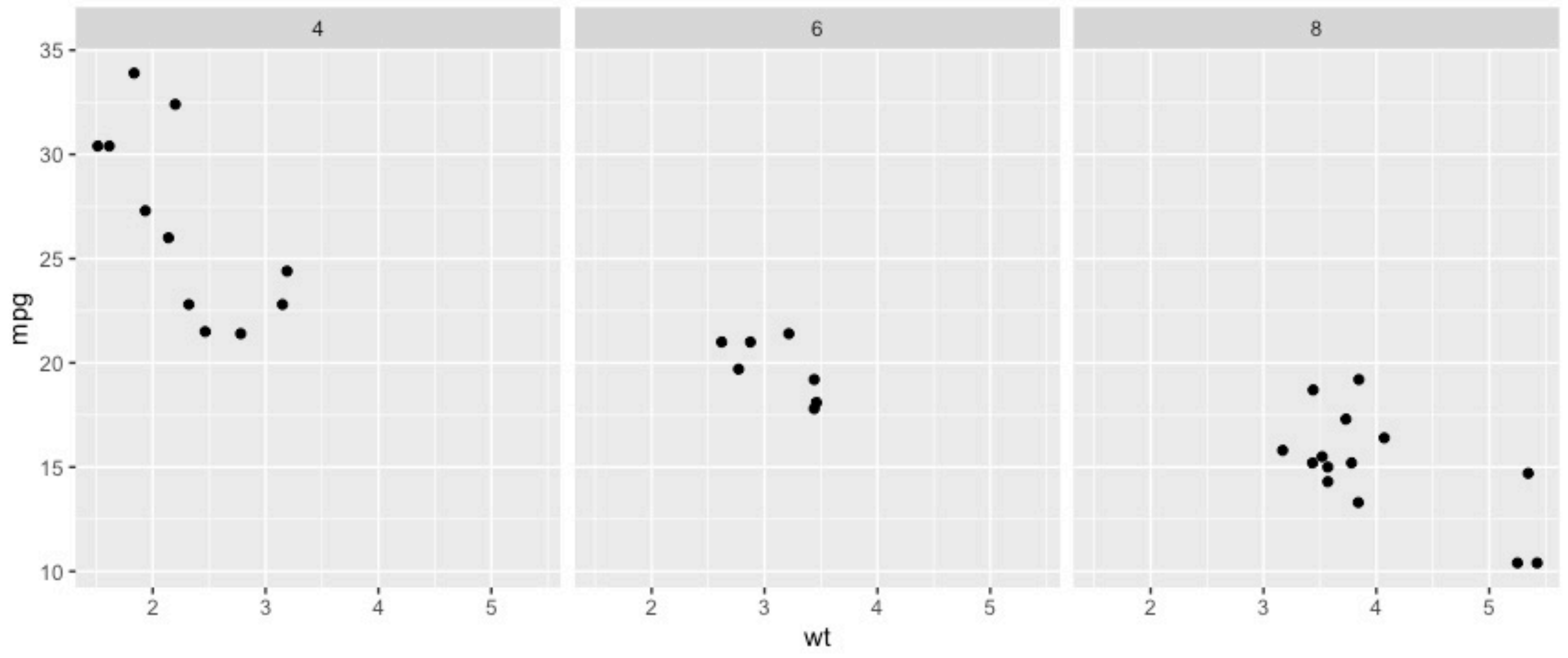


Sets of scatterplots three ways

```r
xyplot(mpg ~ wt | as.factor(cyl), data = mtcars)

par(mfrow = c(1, 3))
plot(mtcars$wt[mtcars$cyl == 4], mtcars$mpg[mtcars$cyl == 4])
plot(mtcars$wt[mtcars$cyl == 6], mtcars$mpg[mtcars$cyl == 6])
plot(mtcars$wt[mtcars$cyl == 8], mtcars$mpg[mtcars$cyl == 8])


ggplot(mtcars, aes(x=wt, y = mpg)) + geom_point() +
 facet_grid(~cyl)
```

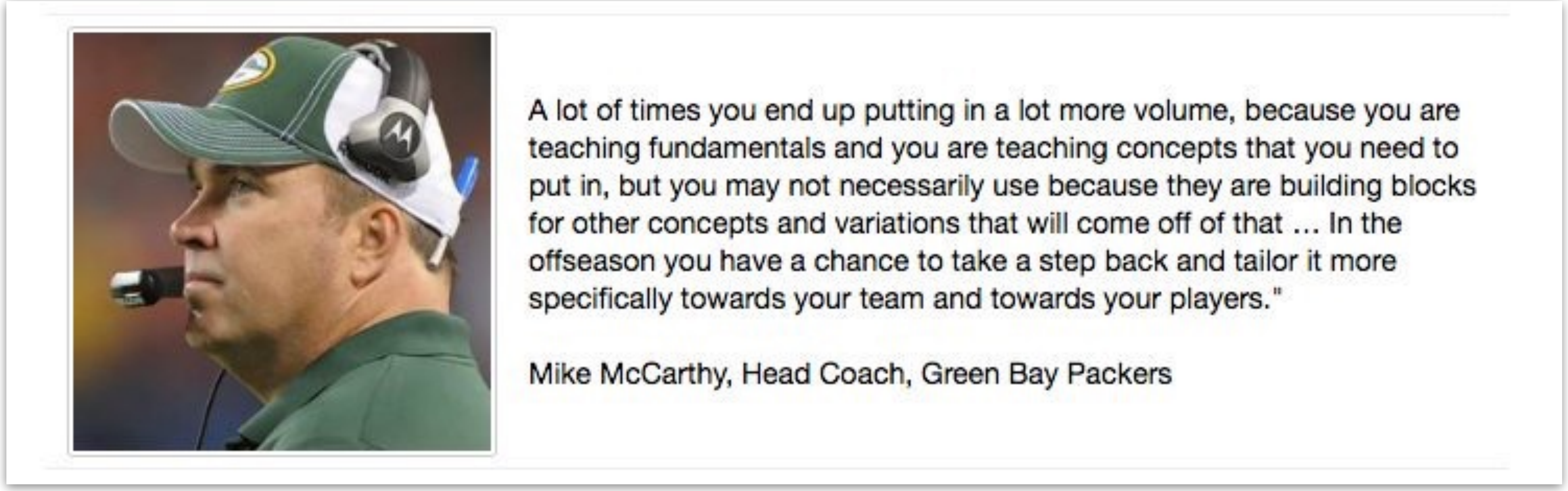

mtcars$mpg[mtcars$cyl == 4]
22
24
26
28
30
32
34
mtcars$mpg[mtcars$cyl == 6]
18.0
18.5
19.0
19.5
20.0
20.5
21.0
21.
mtcars$mpg[mtcars$cyl == 8]
12
14
16
18


2
3
4
5
4
6
8
mpg
10
15
20
25
30
35
2
3
4
5
2
3
4
5


4
6
8
mpg
10
15
20
25
30
35
2
3
4
5
2
3
4
5
2
3
4
5
wt

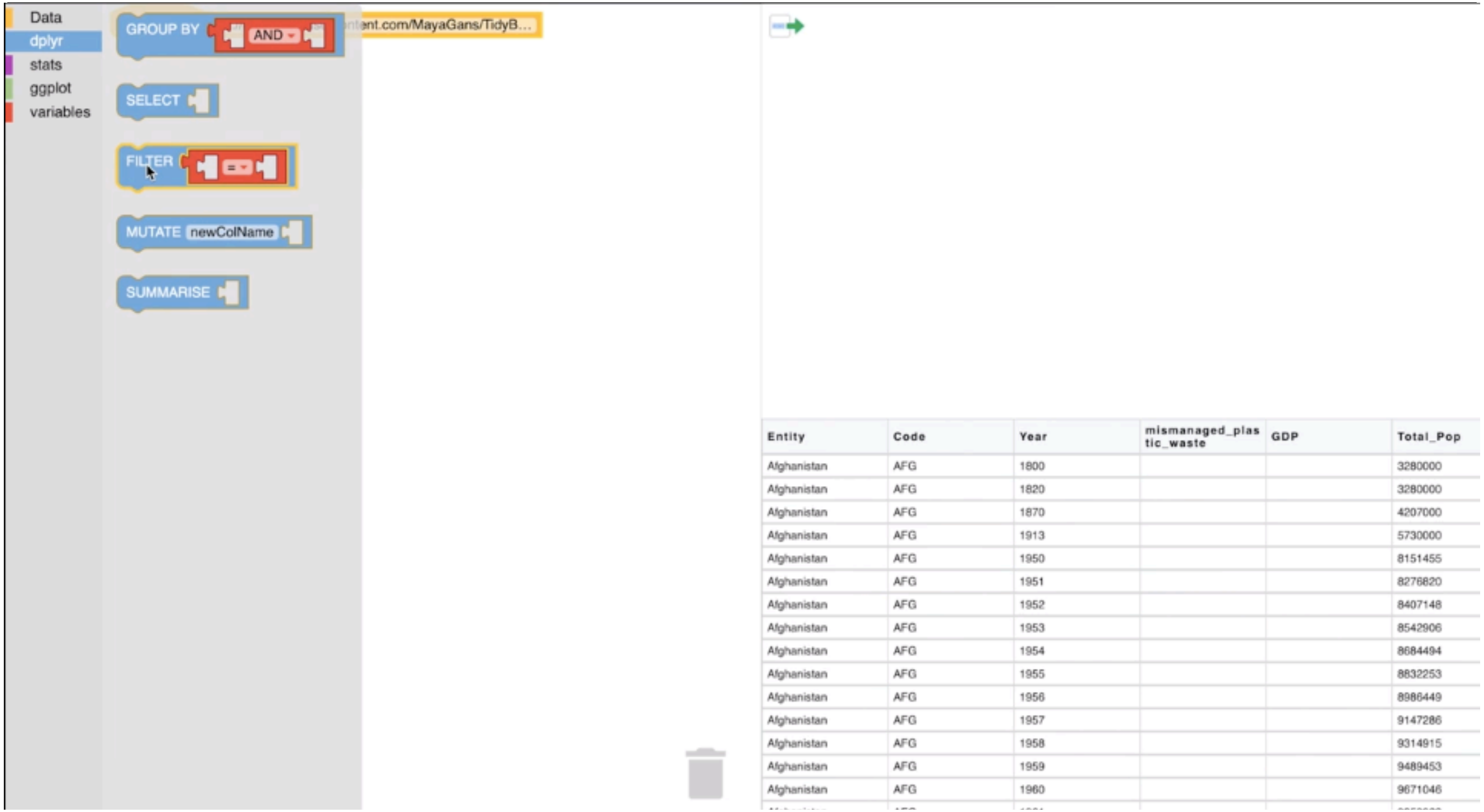
“Less volume, more creativity.”

–Mike McCarthy / `mosaic` package philosophy



A lot of times you end up putting in a lot more volume, because you are teaching fundamentals and you are teaching concepts that you need to put in, but you may not necessarily use because they are building blocks for other concepts and variations that will come off of that … In the offseason you have a chance to take a step back and tailor it more specifically towards your team and towards your players."

Mike McCarthy, Head Coach, Green Bay Packers



Entity	Code	Year	mismanaged_plastic_waste	GDP	Total_Pop
Afghanistan	AFG	1800			3280000
Afghanistan	AFG	1820			3280000
Afghanistan	AFG	1870			4207000
Afghanistan	AFG	1913			5730000
Afghanistan	AFG	1950			8151455
Afghanistan	AFG	1951			8276820
Afghanistan	AFG	1952			8407148
Afghanistan	AFG	1953			8542906
Afghanistan	AFG	1954			8684494
Afghanistan	AFG	1955			8832253
Afghanistan	AFG	1956			8986449
Afghanistan	AFG	1957			9147286
Afghanistan	AFG	1958			9314915
Afghanistan	AFG	1959			9489453
Afghanistan	AFG	1960			9671046

Maya Gans, tidyblocks demo (video)

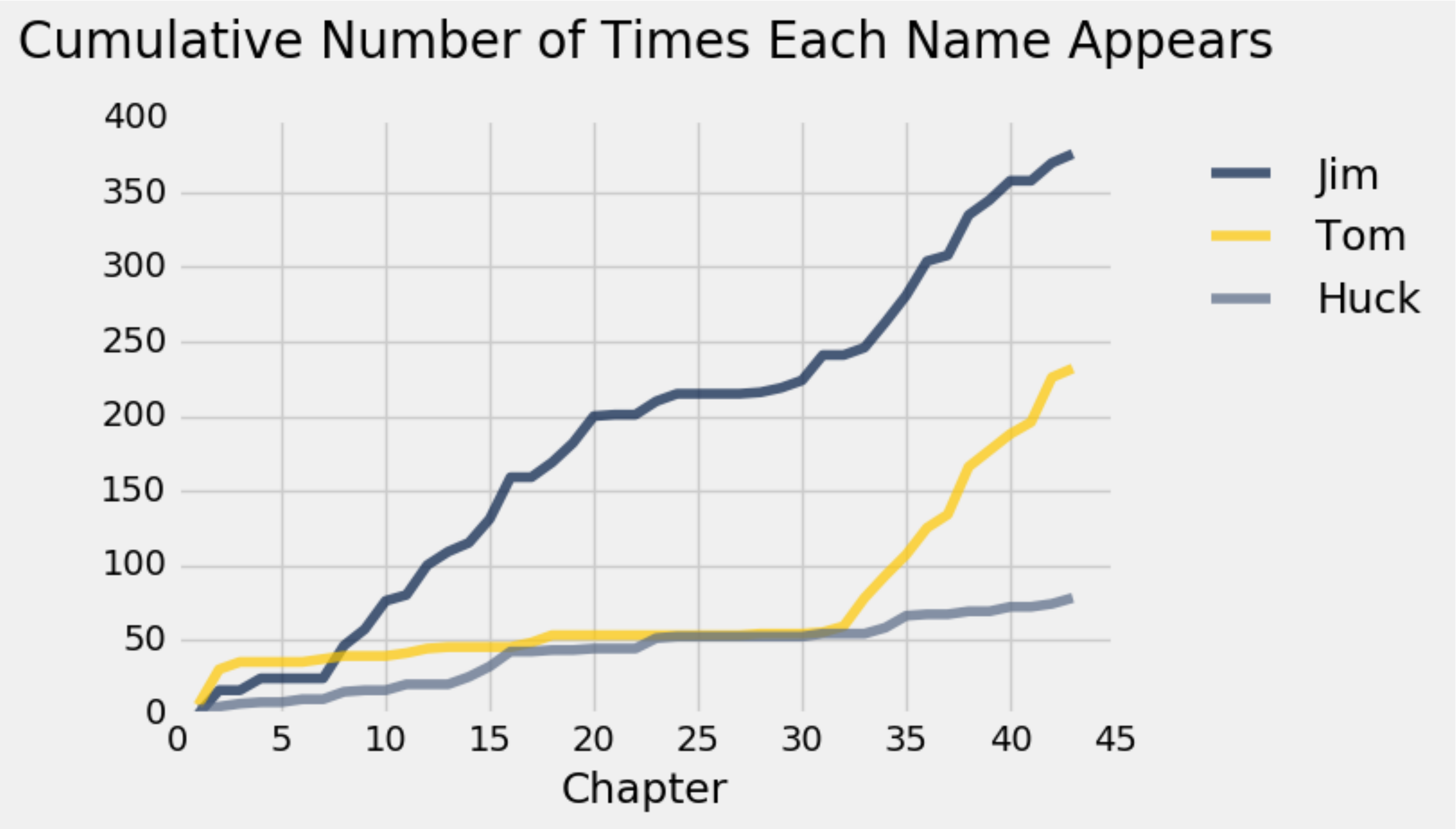
data8 exercise 1 (python)

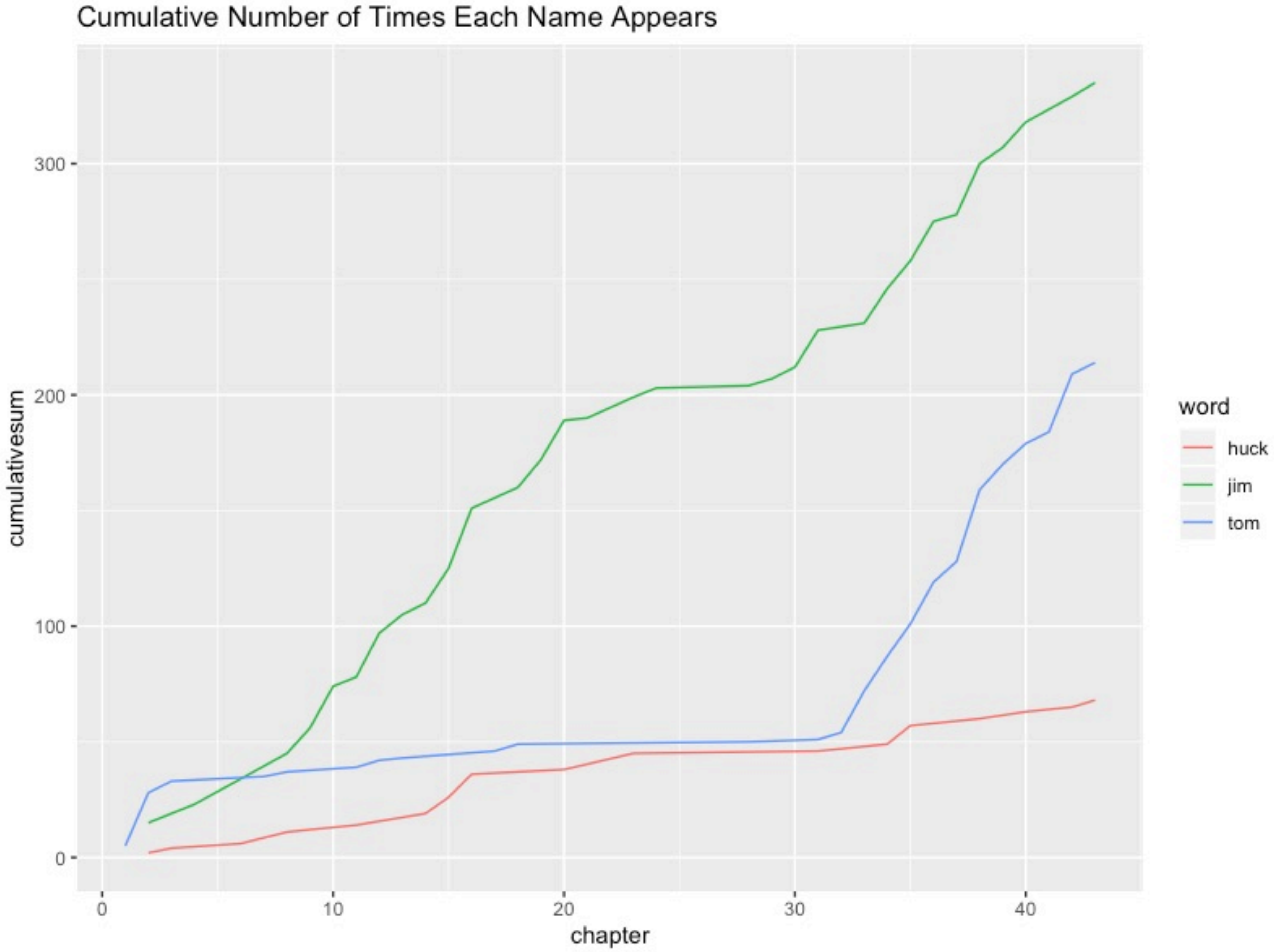
```python
# Count how many times the names Jim, Tom, and Huck appear in each chapter.

counts = Table().with_columns([
        'Jim', np.char.count(huck_finn_chapters, 'Jim'),
        'Tom', np.char.count(huck_finn_chapters, 'Tom'),
        'Huck', np.char.count(huck_finn_chapters, 'Huck')
    ])

# Plot the cumulative counts:
# how many times in Chapter 1, how many times in Chapters 1 and 2, and so on.

cum_counts = counts.cumsum().with_column('Chapter', np.arange(1, 44, 1))
cum_counts.plot(column_for_xticks=3)
plots.title('Cumulative Number of Times Each Name Appears', y=1.08);
```

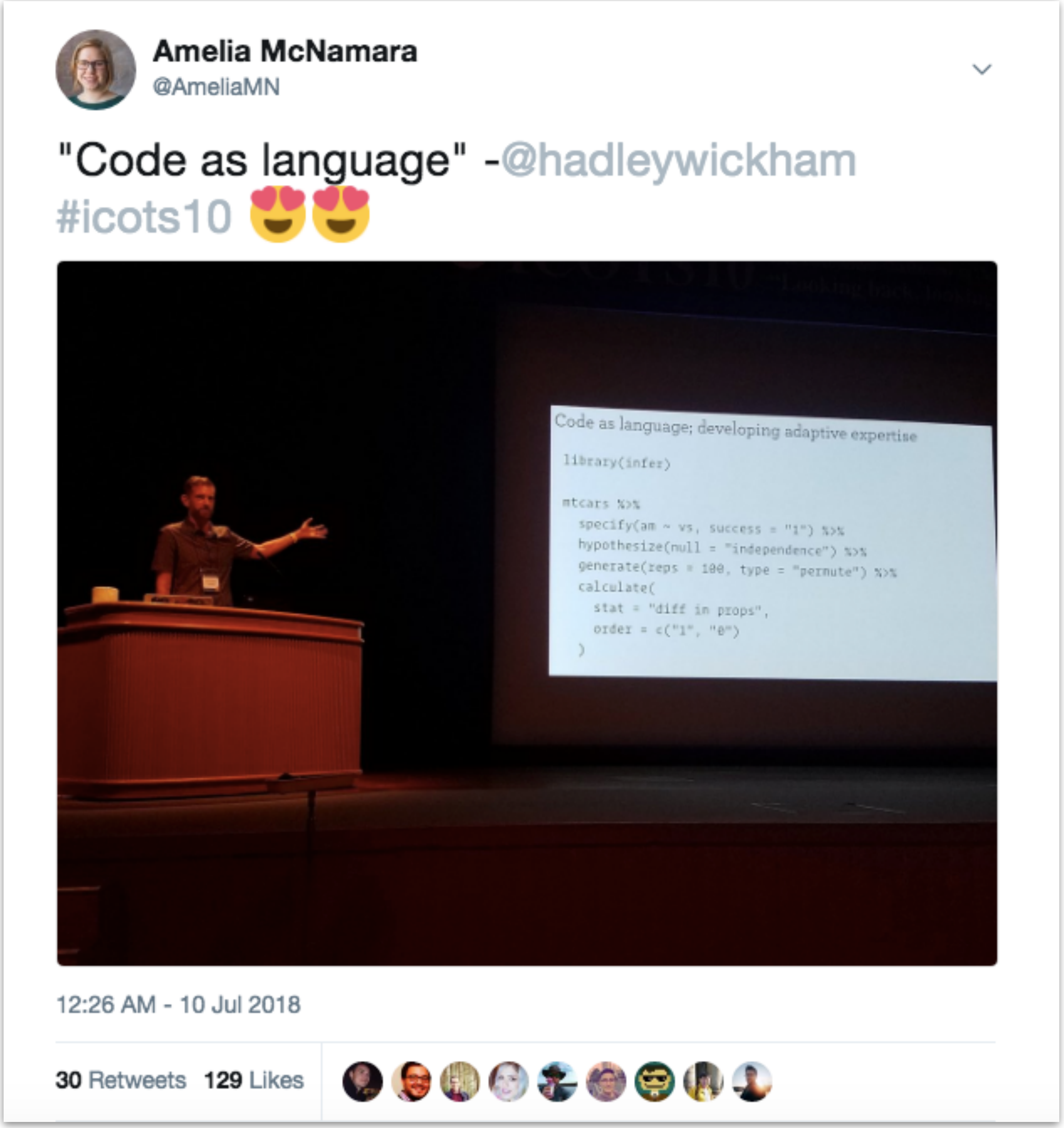
https://predictablynoisy.com/data8-template/textbook/01/3/1/literary-characters/

Exercise ported to R

```
# Count how many times the names Jim, Tom, and Huck appear in each chapter.

counts = huck_finn_chapters %>%
  filter(word %in% c('jim', 'tom', 'huck')) %>%
  group_by(chapter, word) %>%
  summarize(count = n())

# Plot the cumulative counts:
# how many times in Chapter 1, how many times in Chapters 1 and 2, and so on.

cum_counts = counts %>%
  group_by(word) %>%
  mutate(cumulativesum = cumsum(count))

ggplot(cum_counts) +
  geom_line(aes(x=chapter, y = cumulativesum, color = word)) +
  ggtitle("Cumulative Number of Times Each Name Appears")
```


Cumulative Number of Times Each Name Appears
400
350
300
250
200
150
100
50
0
0
5
10
15
20
25
30
35
40
45
Chapter
Jim
Tom
Huck


Cumulative Number of Times Each Name Appears
cumulativesum
300
200
100
0
0
10
20
30
40
chapter
word
huck
jim
tom



Amelia McNamara
@AmeliaMN

"Code as language" -@hadleywickham #icots10 😍😍

Code as language; developing adaptive expertise

```
library(infer)

mtcars %>%
  specify(am ~ vs, success = "1") %>%
  hypothesize(null = "independence") %>%
  generate(reps = 100, type = "permute") %>%
  calculate(
    stat = "diff in props",
    order = c("1", "0")
  )
```

12:26 AM - 10 Jul 2018

30 Retweets **129** Likes

TED Ideas worth spreading
WATCH
DISCOVER
Share
Add to list
Like
Rate
Lera Boroditsky | TEDWomen 2017
How language shapes the way we think
14:13

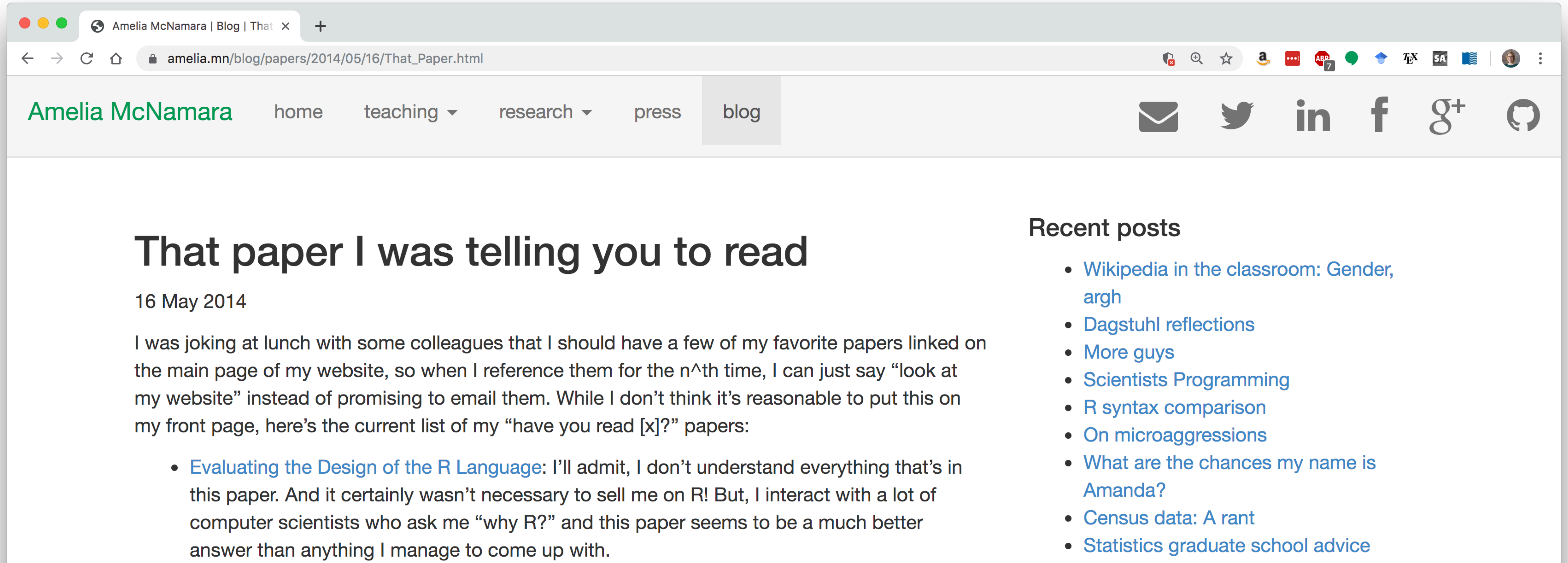
Lera Boroditsky, "How language shapes the way we think" (video)

“It’s a language for data analysis. And if you think that the language is a little incoherent, a little confusing, a bit of a maze, well then all I have to say is welcome to data analysis.”

– Roger Peng, useR keynote

Evaluating the Design of the R Language: Objects and Functions For Data Analysis

Floreal Morandat, Brandon Hill, Leo Osvald and Jan Vitek. ECOOP'12 Proceedings of the 26th European conference on Object-Oriented Programming. 2012. https://dl.acm.org/citation.cfm?id=2367172



Amelia McNamara | Blog | That
amelia.mn/blog/papers/2014/05/16/That_Paper.html
Amelia McNamara
home
teaching
research
press
blog

That paper I was telling you to read

16 May 2014

I was joking at lunch with some colleagues that I should have a few of my favorite papers linked on the main page of my website, so when I reference them for the n^th time, I can just say “look at my website” instead of promising to email them. While I don’t think it’s reasonable to put this on my front page, here’s the current list of my “have you read [x]?” papers:

- Evaluating the Design of the R Language: I’ll admit, I don’t understand everything that’s in this paper. And it certainly wasn’t necessary to sell me on R! But, I interact with a lot of computer scientists who ask me “why R?” and this paper seems to be a much better answer than anything I manage to come up with.

Recent posts

- Wikipedia in the classroom: Gender, argh
- Dagstuhl reflections
- More guys
- Scientists Programming
- R syntax comparison
- On microaggressions
- What are the chances my name is Amanda?
- Census data: A rant
- Statistics graduate school advice

“We will be remiss in our duty to our students if we do not see that they learn to use the computer more easily, flexibly, and thoroughly than we ever have; we will be remiss in our duties to ourselves if we do not try to improve and broaden our own uses.”

–John Tukey,
The Technical Tools of Statistics
talking about the class of 1970

Takeaways

Design for humans

Universal Design is good for everyone

If you're teaching code, read it out loud

Tidyverse syntax seems more readable

Data analysis is sometimes incoherent!

Our goal is to make it more coherent over time

forcats
googlesheets
plumber
fs
yardstick
devtools
lobstr
stringr
feather
broom
lot2
blogdown
lubridate
Jane Austen
janeaustenr
readxl
knitr
rmarkdown
bookdown
haven
withr
scales
dev
parsnip
test that
%>%
Ceci n'est pas un pipe.
TIBBLE
reprex
readr
glue
roxygen2
tidyverse
xaringan
Shiny
lyr
RStudio
visdat
Skimr
packrat
R Ladies
ggplot2
blogdown
yardstick
googlesheets
%>%
Ceci n'est pas un pipe.
sca
rmarkdown
roxygen2
tidyverse
stringr
forcats
Jane Austen
readxl
bookdown
adr
%>%
Ceci n'est pas un pipe.
yardstick
glue
roxygen2
dplyr
%>%
Ceci n'est pas un pipe.
ggplot2
tidyverse
withr
glue
rep
R Ladies
plumber
fs
blogdown
TIBBLE
devtools
usethis
broom
Skimr
packrat
yardstick

Thank you!

An aside — the dress

Fabric design is on GitHub
https://github.com/ameliamn/hexfabric

Fabric for purchase is on Spoonflower
https://www.spoonflower.com/profiles/ameliamn

I don’t take dress orders. Making this one was challenging enough!