Imagining the future of statistical education software

Amelia McNamara (@AmeliaMN)
Assistant Professor, Department of Computer & Information Sciences
University of St Thomas, St Paul, MN USA

flickr: takchabo

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

Taylor & Francis
Taylor & Francis Group

Check for updates

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 tools designed for **learning** statistics are typically:

- graphical
- interactive
- intuitive
- supportive of EDA

but:

- don't support reproducibility
- can't handle real data

tools designed for **doing** statistics are typically:

- powerful
- flexible
- reproducible
- supportive of extensions

but:

- hard to get started using
- not interactive

Fathom®

TinkerPlots™

StatKey to accompany *Statistics: Unlocking the Power of Data* by Lock, Lock, Lock, Lock, and Lock

Rossman/Chance Applet Collection

STATA®

R

SPSS® AN IBM® COMPANY

sas® THE POWER TO KNOW®

MATLAB® The Language of Technical Computing

We need a bridge between the two

Could be software, or curriculum. Today, I'm focused on software.

flickr: takchabo

http://bit.ly/Biehler1997

International Statistical Review (1997), **65**, 2, 167–189, Printed in Mexico

Software for Learning and for Doing Statistics

Rolf Biehler

Institut für Didaktik der Mathematik (IDM), Universität Bielefeld, Postfach 10 01 31, D-33501 Bielefeld, Germany.
e-mail: rolf.biehler@post.uni-bielefeld.de

Summary

The community of statisticians and statistics educators should take responsibility for the evaluation and improvement of software quality from the perspective of education. The paper will develop a perspective, an ideal system of requirements to critically evaluate existing software and to produce future software more adequate both for learning and doing statistics in introductory courses. Different kinds of tools and microworlds are needed. After discussing general requirements for such programs, a prototypical ideal software system will be presented in detail. It will be illustrated how such a system could be used to construct learning environments and to support elementary data analysis with exploratory working style.

Key words: Statistics education; Statistical software design; Evaluation of statistical software; Exploratory data analysis; Simulation.

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

	Accessibility	Easy entry	Data as first-order object	EDA/ CDA	Flexible plotting	Randomization	Interactivity	Inherent documentation	Narrative, publishing, reproducibility	Flexibility for extensions
Graphing calculators	*	✓								
Excel	*	✓					✓			
applets	*	✓				✓	✓			
TinkerPlots	*	✓		✓	✓	✓	✓	✓		
Fathom	*	✓		✓	✓	✓	✓	✓		
R	✓		✓	✓	✓	*	*		✓	✓
Python	✓		✓		*	*	*		✓	✓
SAS software			✓	✓		*			*	✓
Stata software			✓	✓		*			*	✓

Table 1: A summary of many currently-available tools for learning and doing statistics, and how they satisfy the attributes outlined in this paper. Asterisks indicate partial satisfaction of the attribute. For example, most tools are not accessible, either because of prohibitive cost or because they do not support disabled users. R and Python are free and can be used with adaptive technology. R, Python, SAS software, and Stata software get an asterisk for randomization because it is possible within the system, but difficult for novices. Similarly, R and Python can be used to create interactive graphics, but it is difficult, and SAS software and Stata software can be used to create reproducible reports, although it is difficult.

On the State of Computing in Statistics Education: Tools for Learning and for Doing. pre-print http://bit.ly/StateOfComputingPreprint

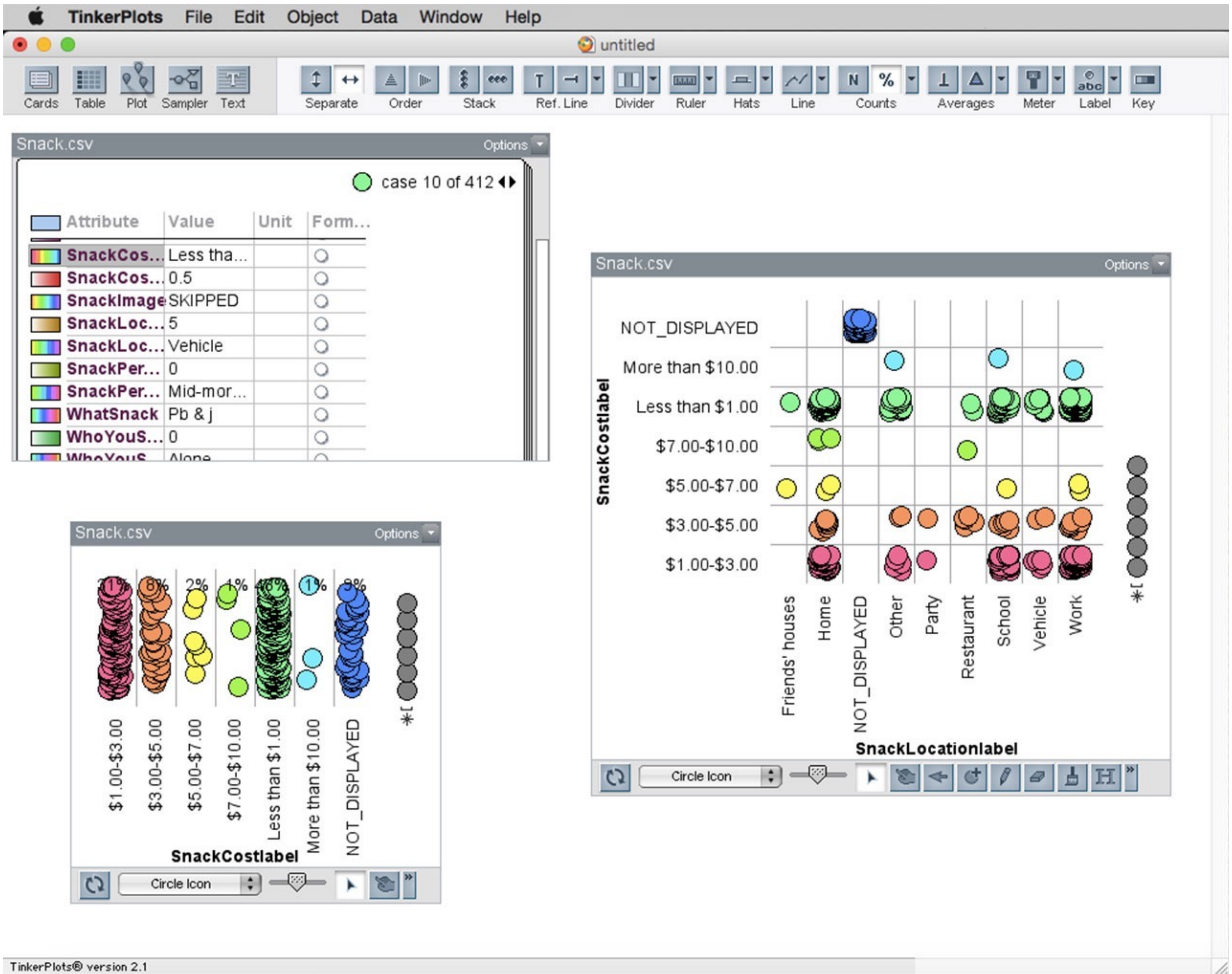
Accessibility

- free or inexpensive
- available on many platforms
- compatible with accessibility features

Easy entry for novice users

- the "complexity of tool problem" from Biehler
- known as "low threshold" in CS ed literature

TinkerPlots/Fathom



R Syntax Comparison : : CHEAT SHEET

http://bit.ly/R-syntax-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)
```

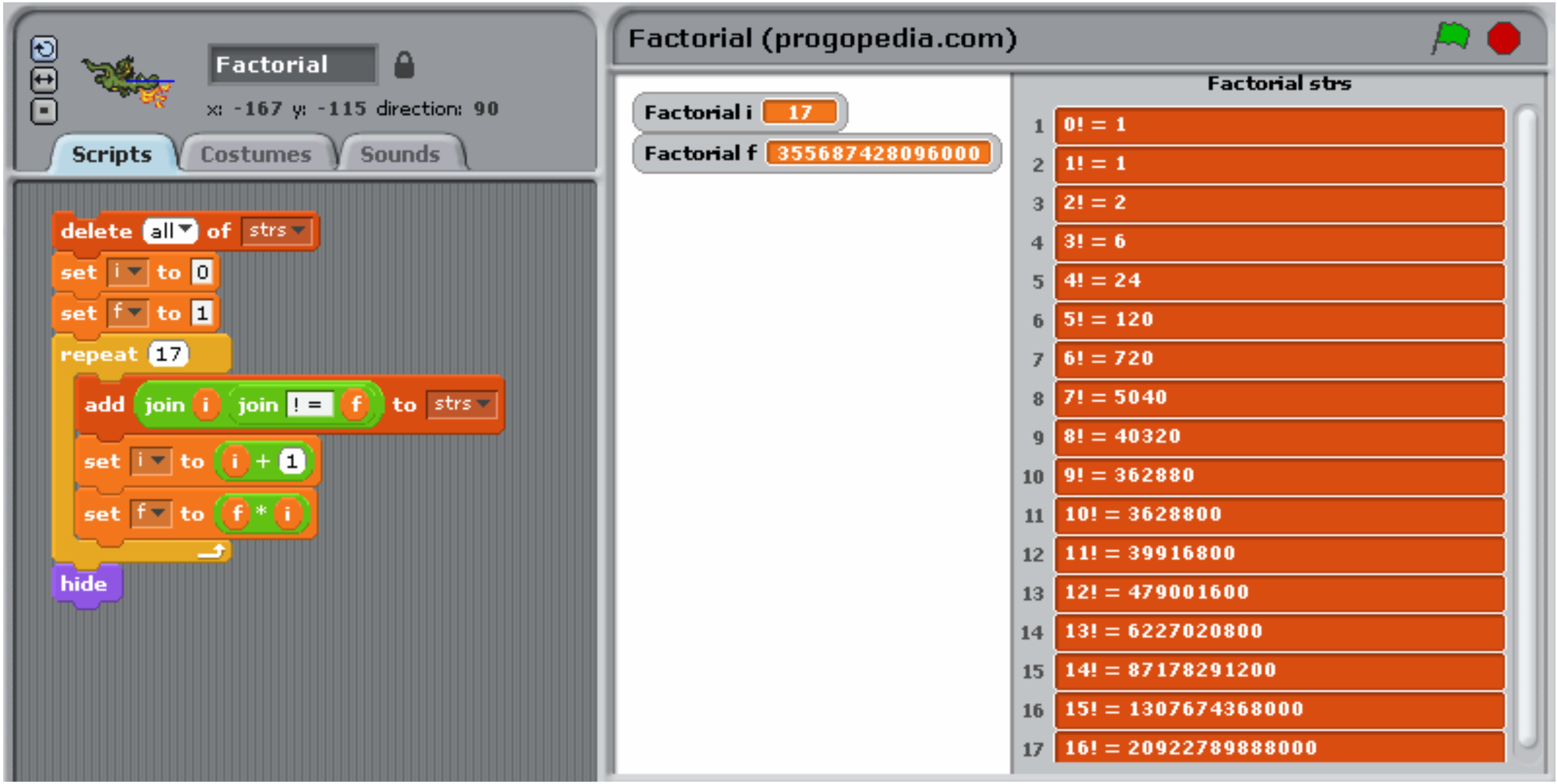
one continuous, one categorical:
```
ggplot2::qplot(x=disp, data=mtcars, geom = "histogram") +
       facet_grid(.~cyl)

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))
```

Scratch

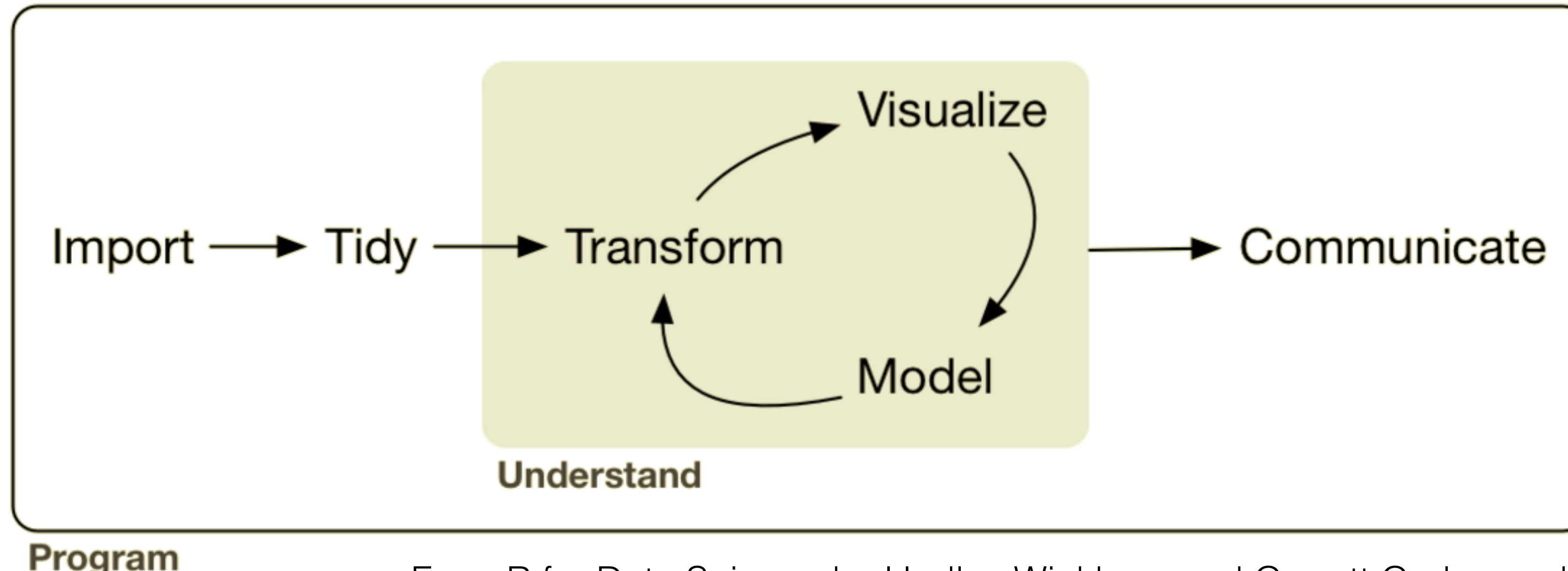


https://scratch.mit.edu/

Data as a first-order object

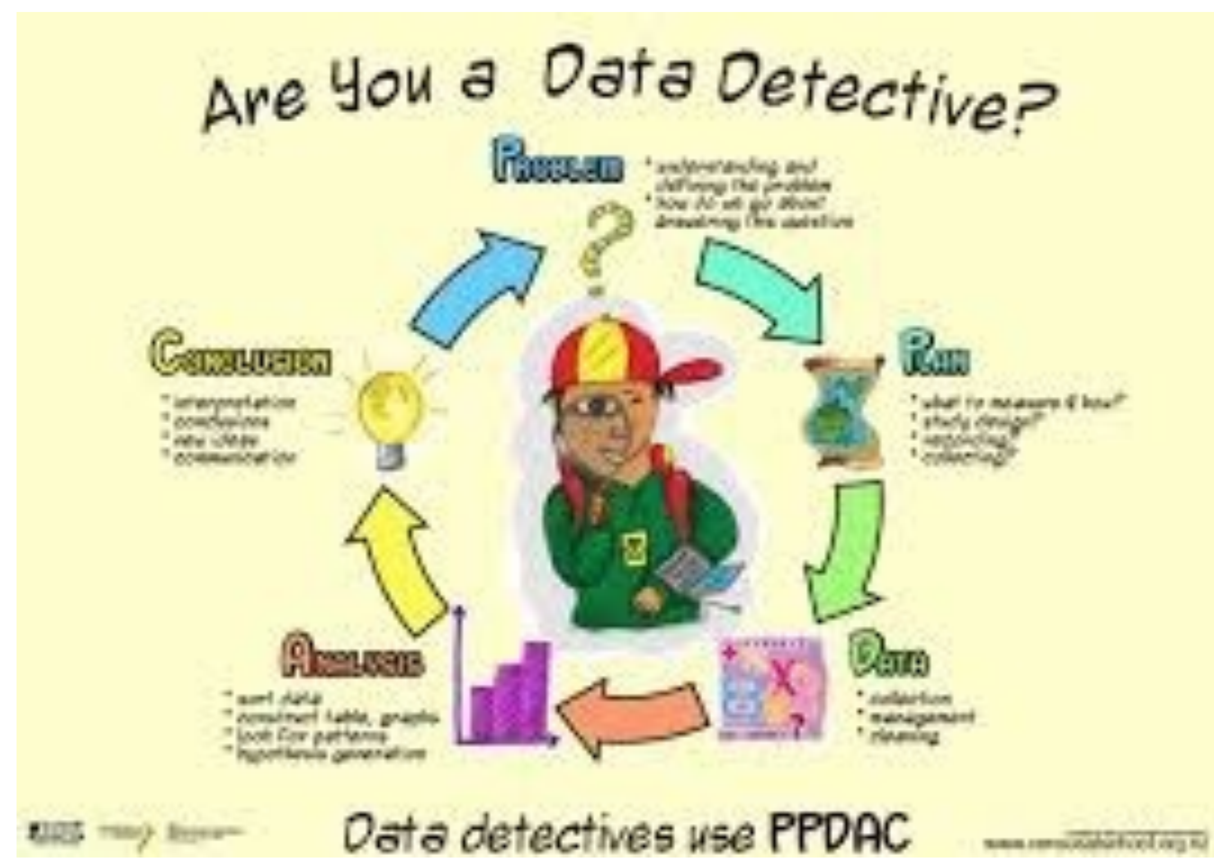
- focused on data
- data should be easily **human-readable**
- support many data types
- difficult to overwrite original data
- affordances for reproducibility

Support for a Cycle of Exploratory and Confirmatory Analysis



From R for Data Science by Hadley Wickham and Garrett Grolemund.

- Users need "scratch paper"

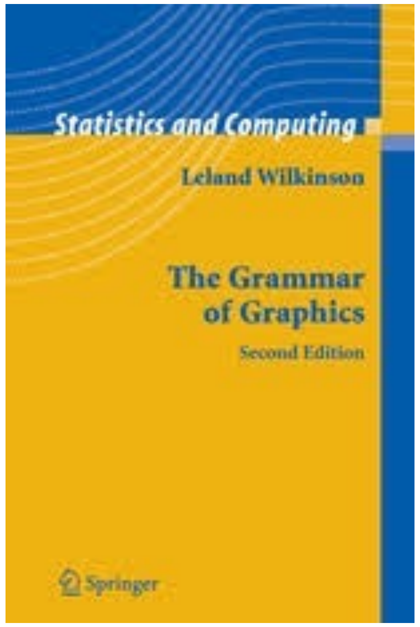


From Census at School

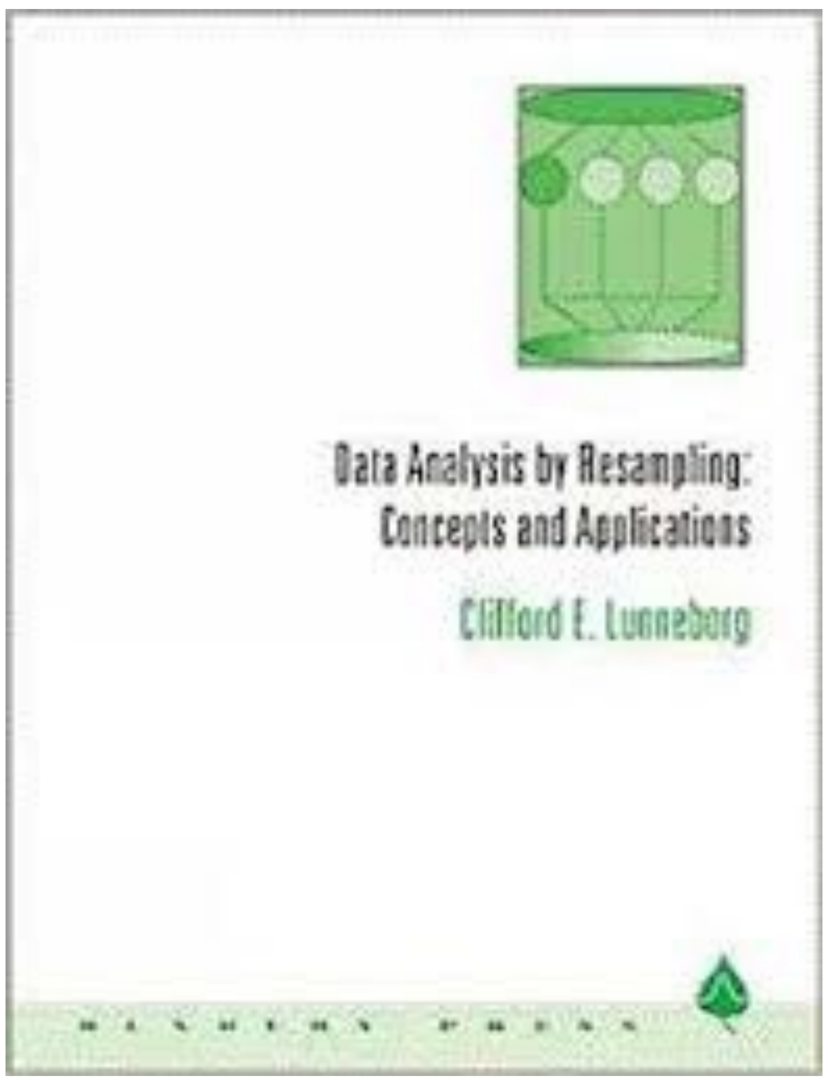
Flexible plot creation

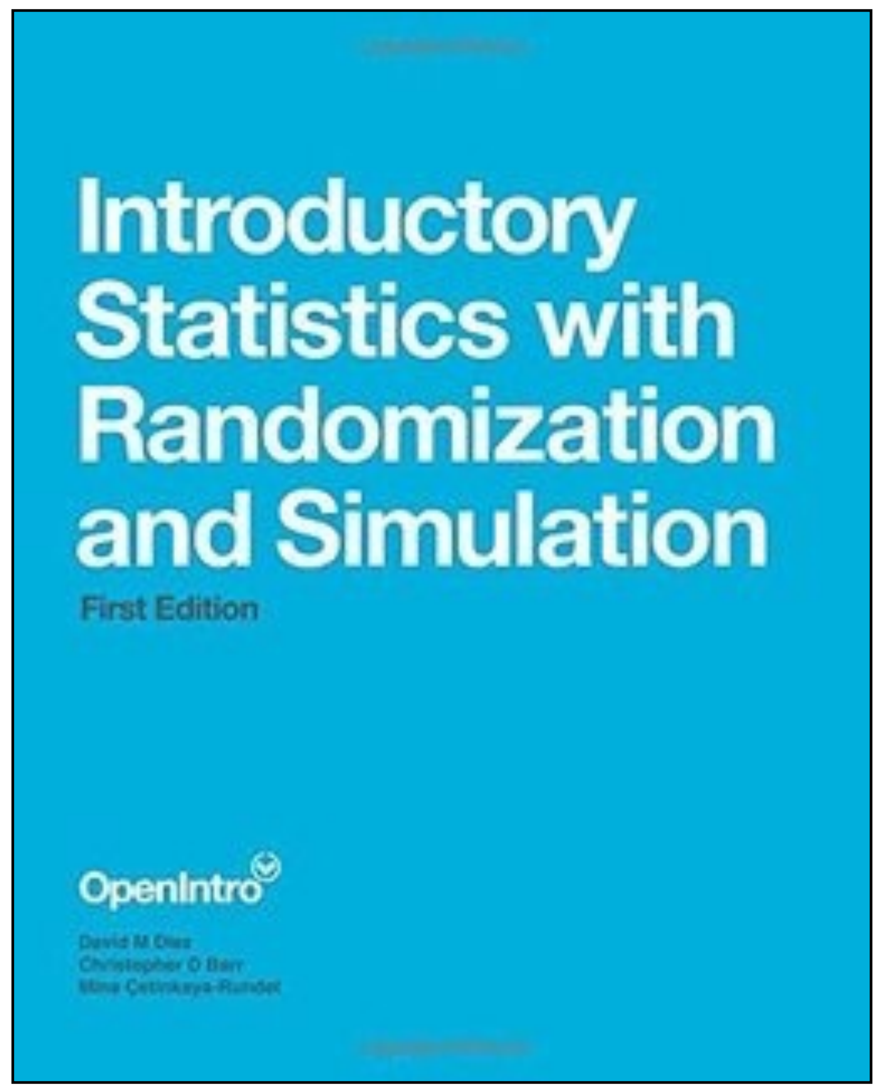
- rather than a few out-of-the-box visualizations, a user should be able to build their own

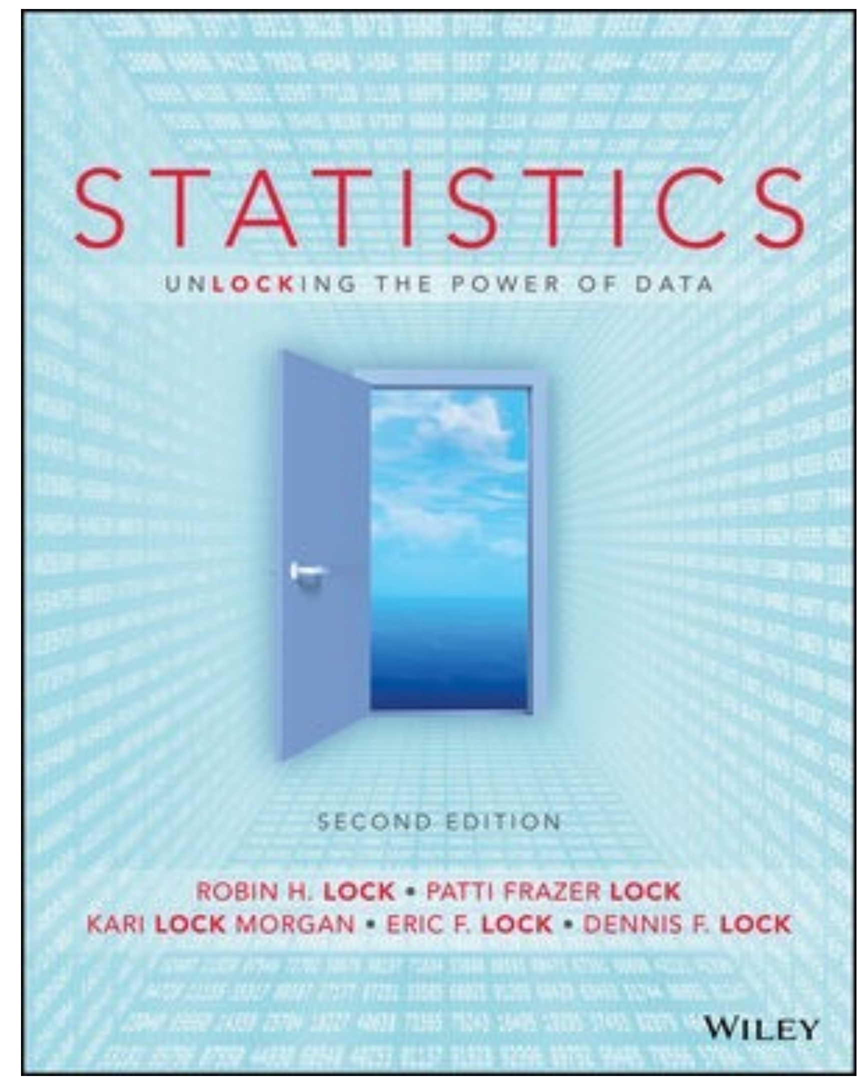
- perhaps following the Grammar of Graphics (like ggplot2 and d3.js)



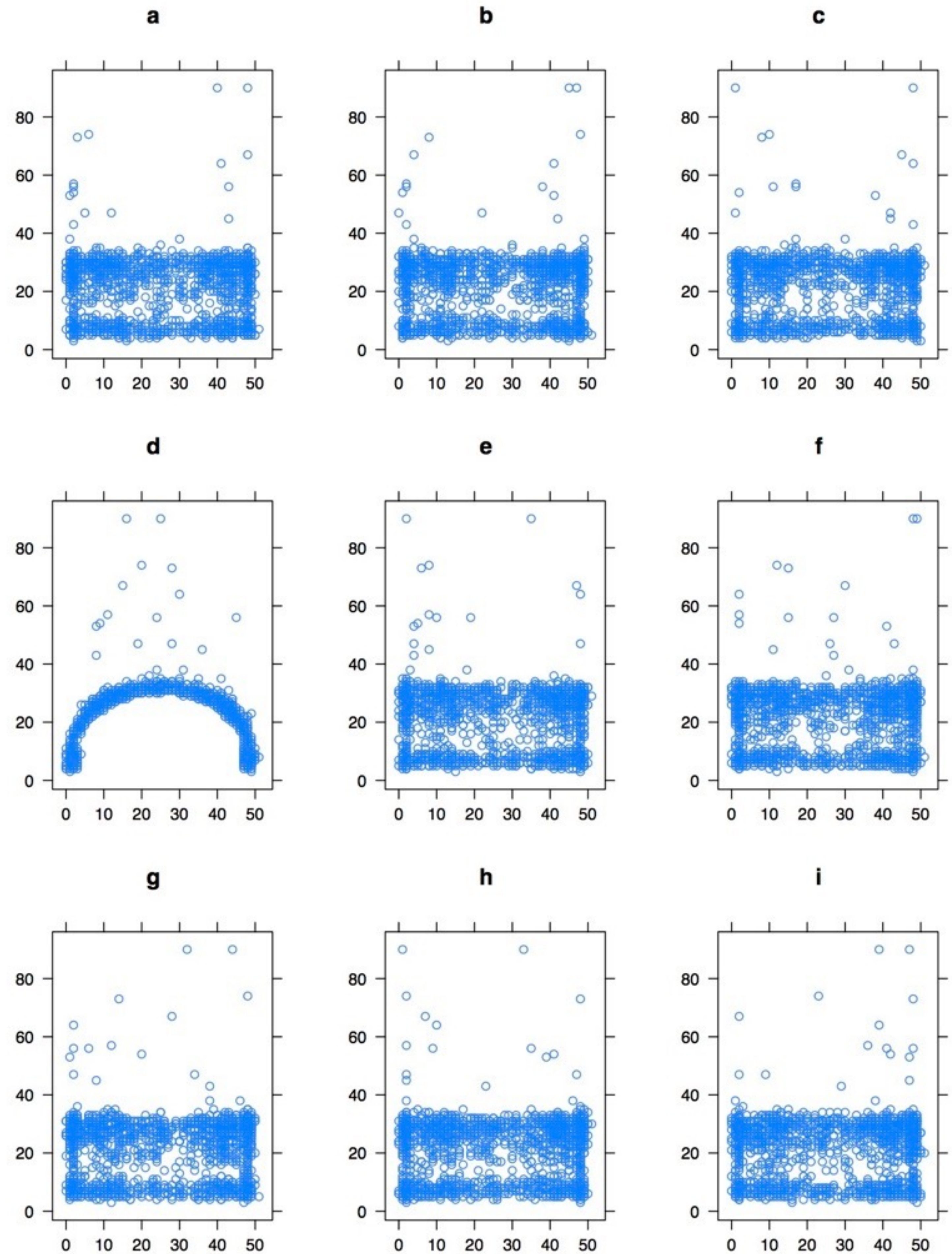
Support for randomization and the bootstrap

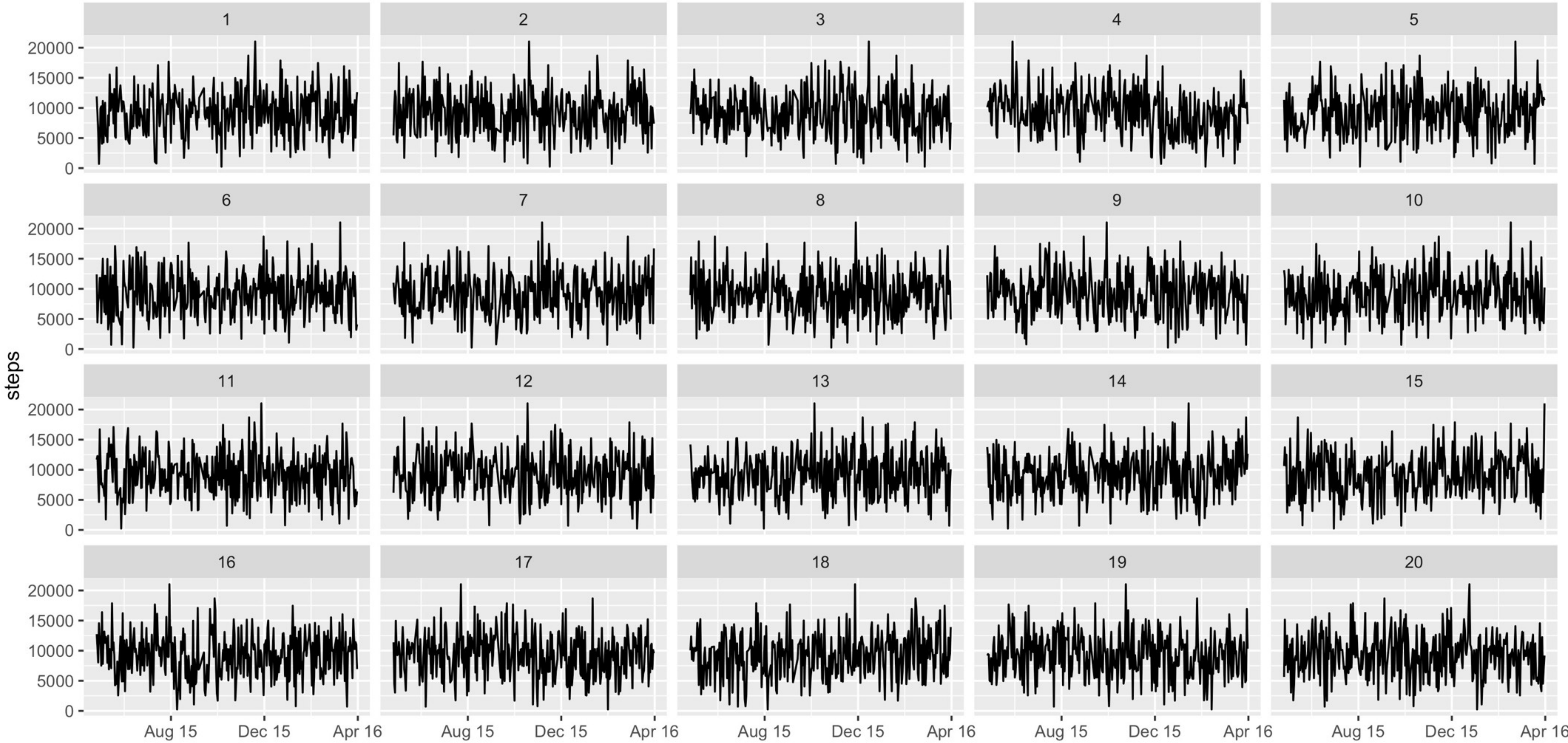




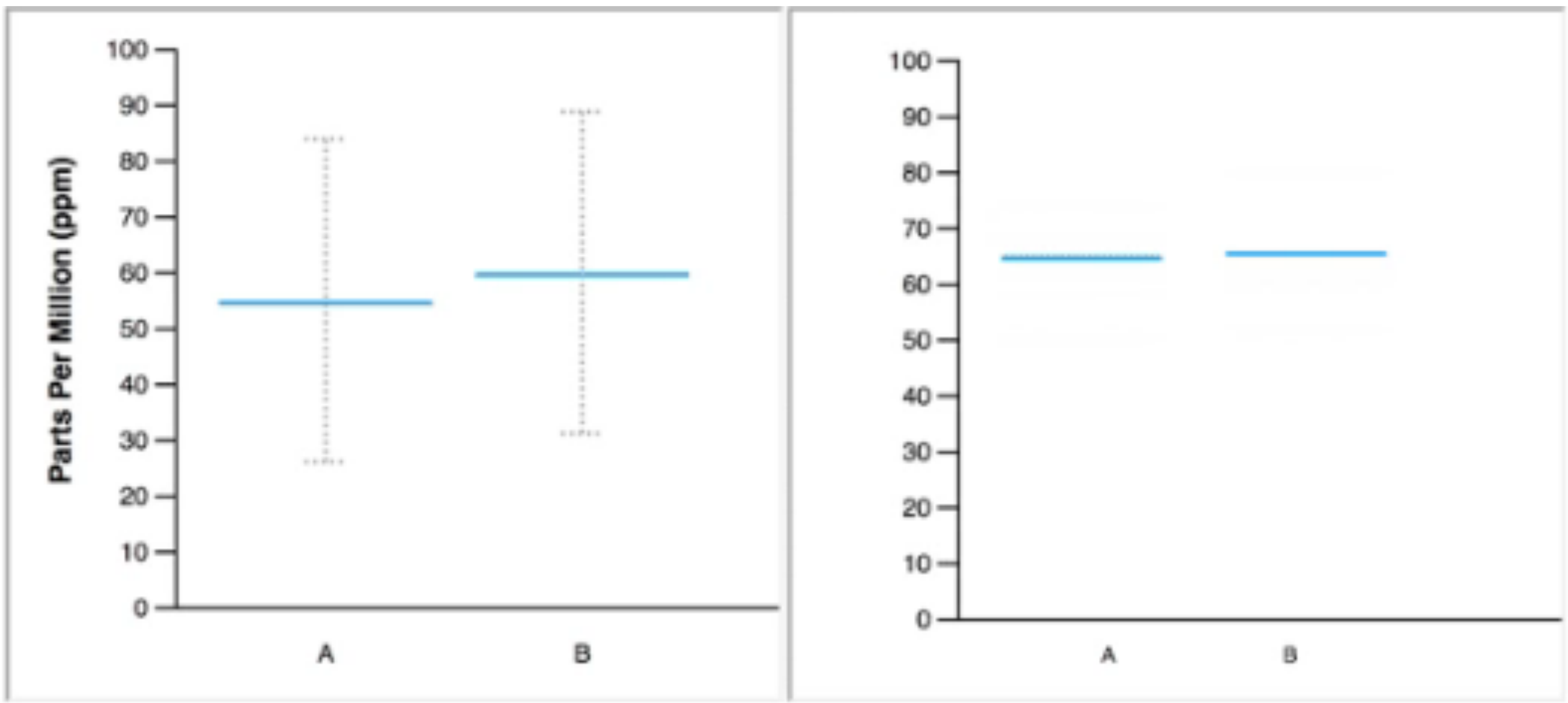


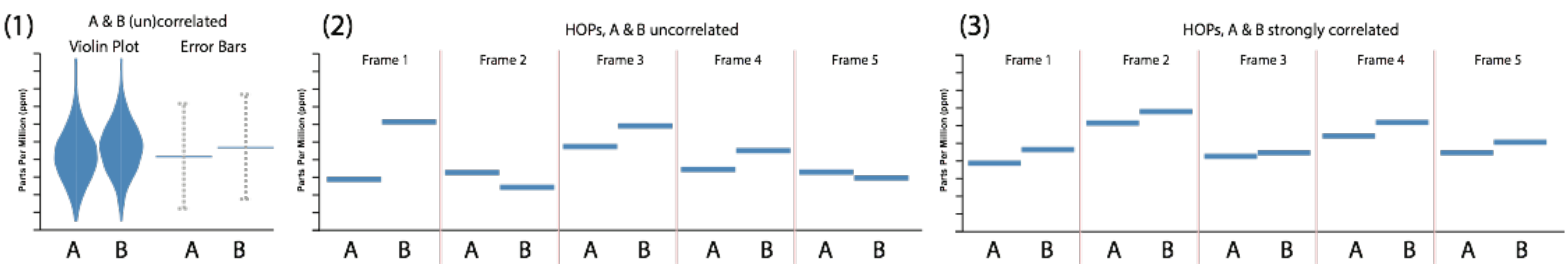
Permuted graphics




1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
steps
20000
15000
10000
5000
0
Aug 15
Dec 15
Apr 16

Jessica Hullman, Paul Resnick, Eytan Adar. (2015). Hypothetical Outcome Plots Outperform Error Bars and Violin Plots for Inferences About Reliability of Variable Ordering. *PLOS ONE,* 10(11). http://bit.ly/HypotheticalOutcomePlots

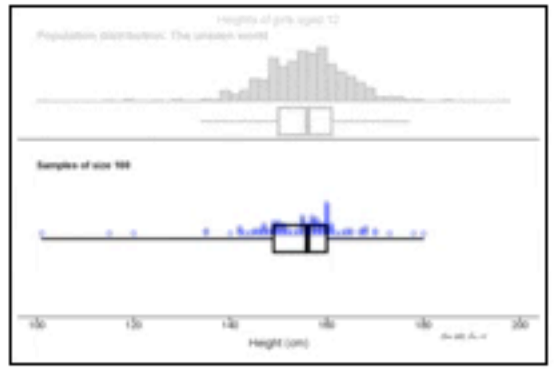


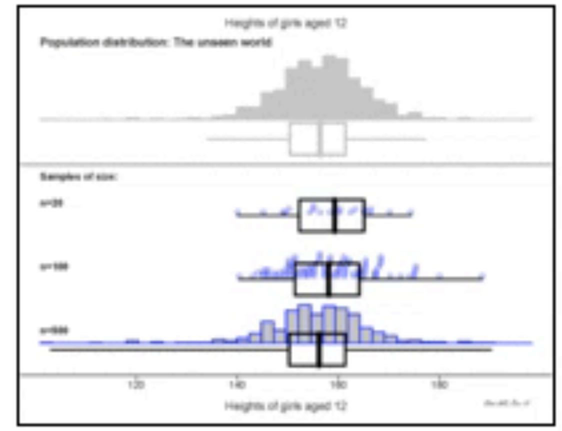


Making the Call

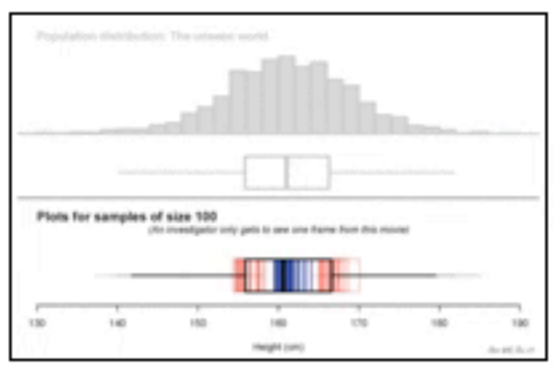
Chris Wild, Nick Horton, Maxine Pfannkuch, Matt Regan

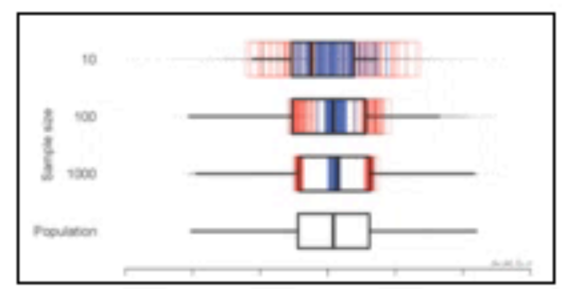
One Population





Animated Gifs	Click for:	n=30	n=100	n=300	Vew Full Size GIF
PDFs	Click for:	n=30	n=100	n=300	View Full size PDF
		1(a)			1(b)





Animated Gifs	Click for:	n=30	n=100	n=300	View full size GIF
PDFs	Click for:	n=30	n=100	n=300	View full size PDF
		2(a)			2(b)

http://bit.ly/GuidlinesForMakingTheCall http://bit.ly/AnimationsForMakingTheCall

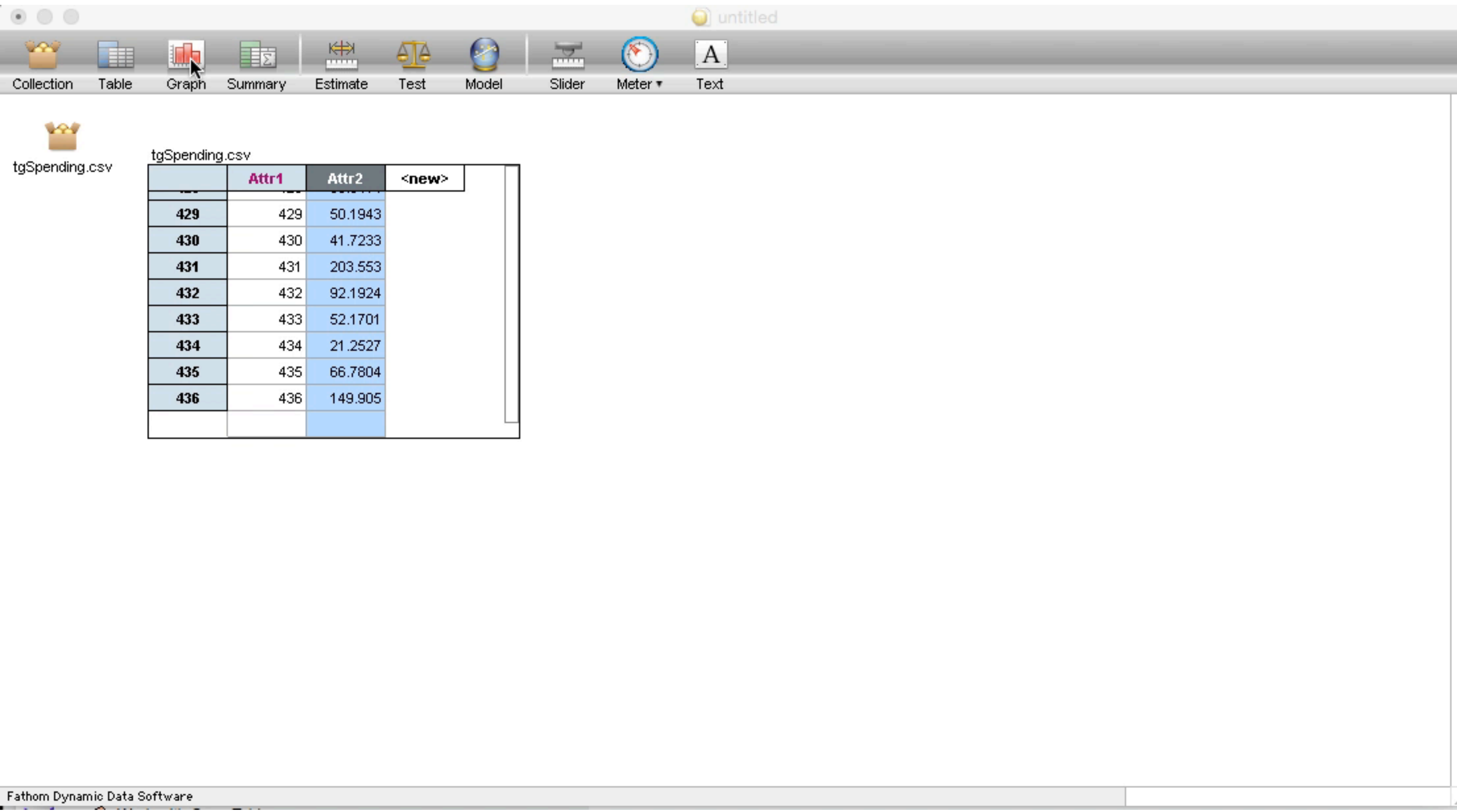
Interactivity at every level

- when developing an analysis
- when adjusting parameters in an analysis
- when a reader is exploring the analysis

prim9

http://bit.ly/prim_9

Fathom


untitled
Collection
Table
Graph
Summary
Estimate
Test
Model
Slider
Meter ▾
Text
tgSpending.csv
tgSpending.csv
Attr1
Attr2
<new>
429 429 50.1943
430 430 41.7233
431 431 203.553
432 432 92.1924
433 433 52.1701
434 434 21.2527
435 435 66.7804
436 436 149.905
Fathom Dynamic Data Software

Tableau

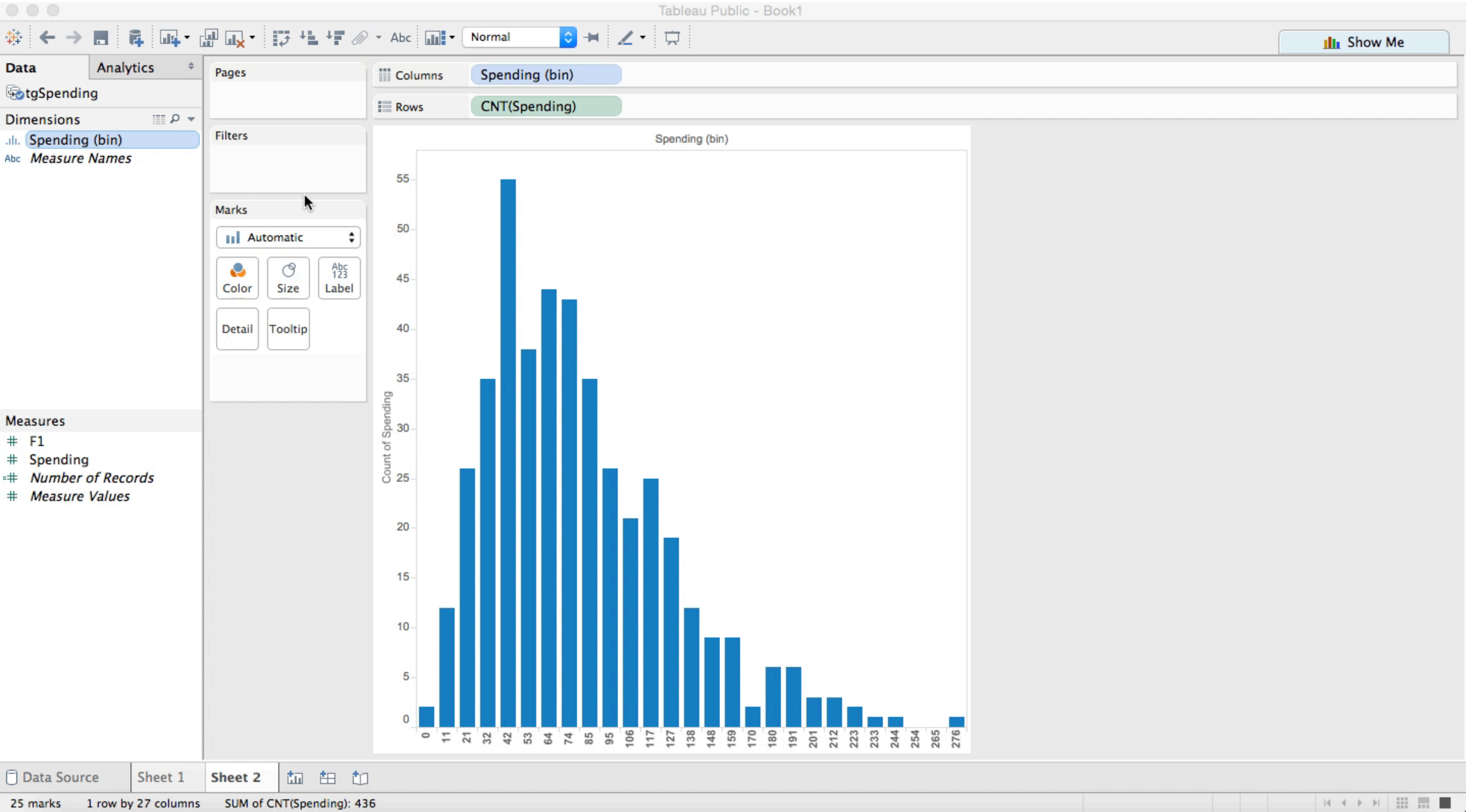
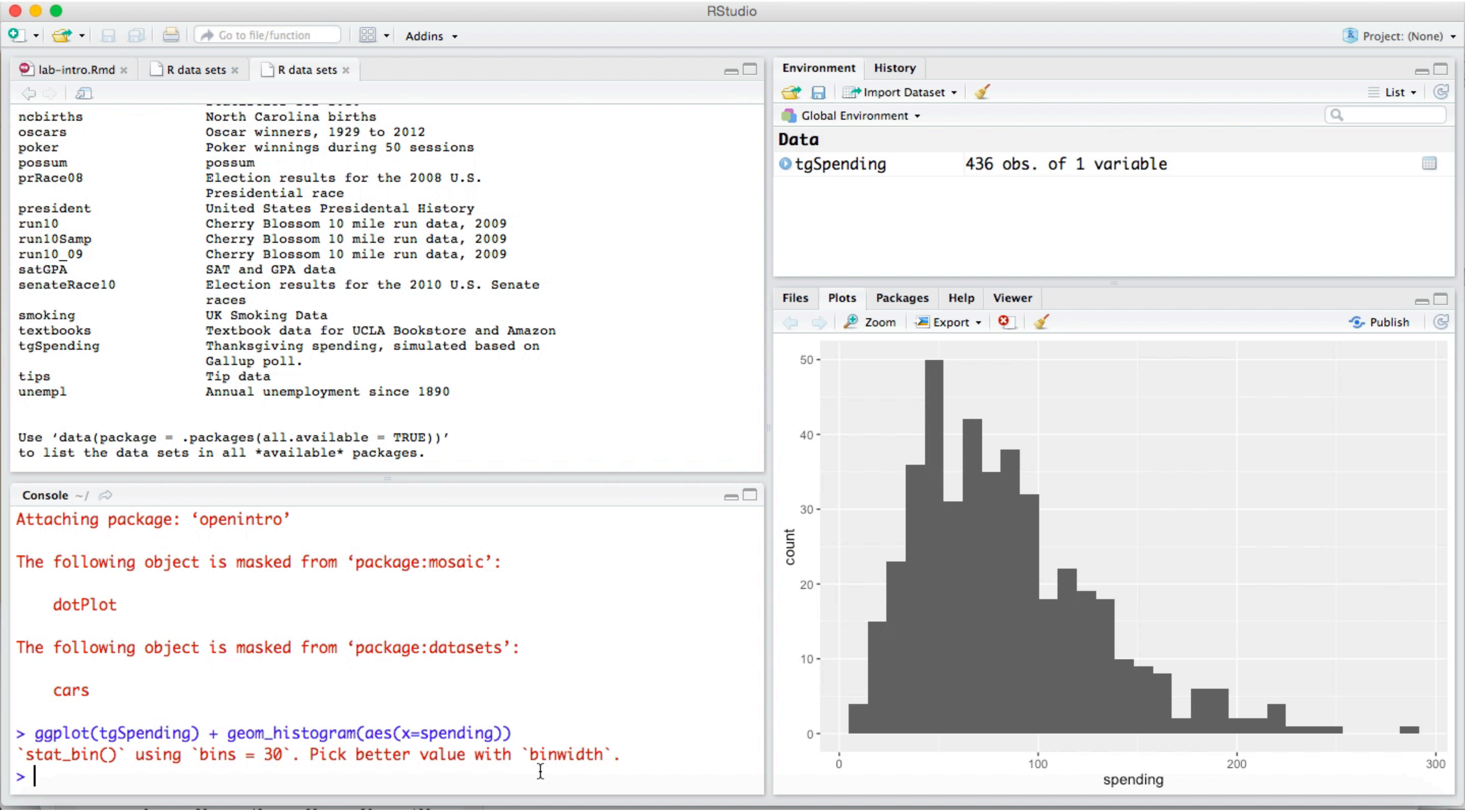
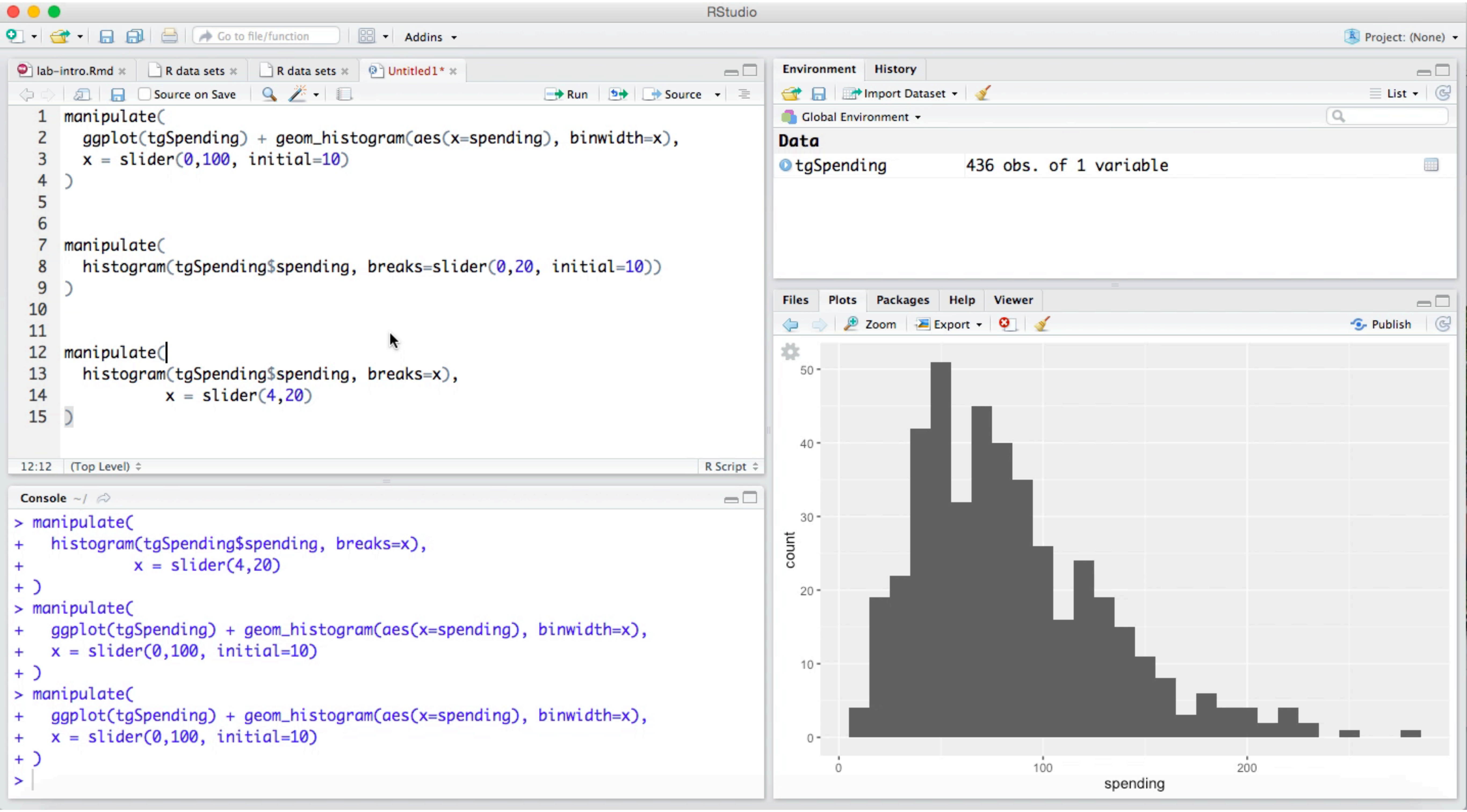


Tableau Public - Book1
Normal
Show Me
Data
Analytics
tgSpending
Dimensions
Spending (bin)
Measure Names
Measures
F1
Spending
Number of Records
Measure Values
Pages
Filters
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Columns
Spending (bin)
Rows
CNT(Spending)
Spending (bin)
Count of Spending
Data Source
Sheet 1
Sheet 2
25 marks
1 row by 27 columns
SUM of CNT(Spending): 436

R/ggplot2


RStudio
Go to file/function
Addins
Project: (None)
lab-intro.Rmd
R data sets
R data sets
ncbirths North Carolina births
oscars Oscar winners, 1929 to 2012
poker Poker winnings during 50 sessions
possum possum
prRace08 Election results for the 2008 U.S. Presidential race
president United States Presidental History
run10 Cherry Blossom 10 mile run data, 2009
run10Samp Cherry Blossom 10 mile run data, 2009
run10_09 Cherry Blossom 10 mile run data, 2009
satGPA SAT and GPA data
senateRace10 Election results for the 2010 U.S. Senate races
smoking UK Smoking Data
textbooks Textbook data for UCLA Bookstore and Amazon
tgSpending Thanksgiving spending, simulated based on Gallup poll.
tips Tip data
unempl Annual unemployment since 1890
Use 'data(package = .packages(all.available = TRUE))'
to list the data sets in all *available* packages.
Console ~/
Attaching package: 'openintro'
The following object is masked from 'package:mosaic':
dotPlot
The following object is masked from 'package:datasets':
cars
> ggplot(tgSpending) + geom_histogram(aes(x=spending))
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
>
Environment
History
Import Dataset
List
Global Environment
Data
tgSpending 436 obs. of 1 variable
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
count
spending
0
10
20
30
40
50
0
100
200
300

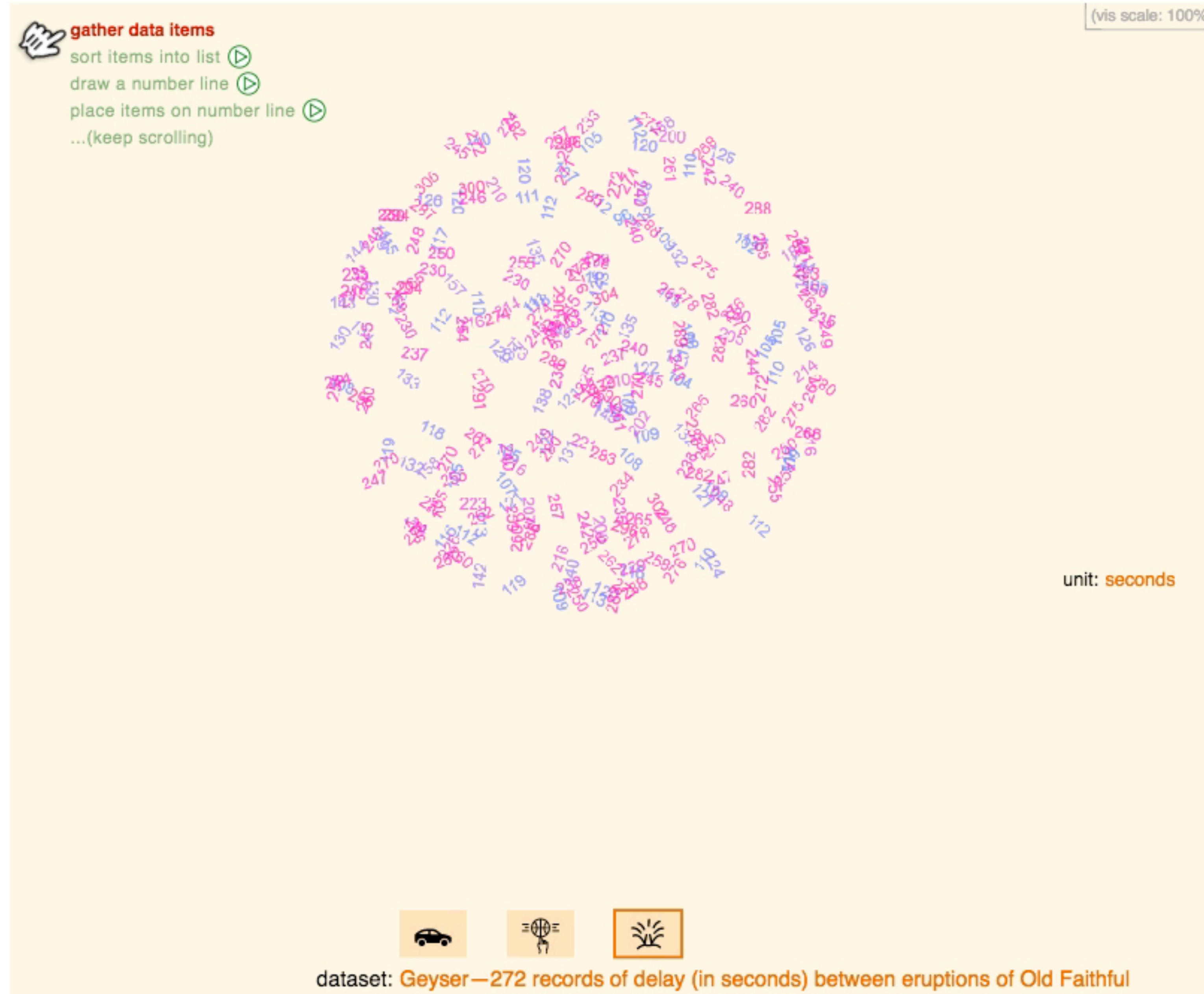
manipulate


RStudio
Go to file/function
Addins
Project: (None)
lab-intro.Rmd
R data sets
R data sets
Untitled1*
Source on Save
Run
Source
manipulate(
ggplot(tgSpending) + geom_histogram(aes(x=spending), binwidth=x),
x = slider(0,100, initial=10)
)
manipulate(
histogram(tgSpending$spending, breaks=slider(0,20, initial=10))
)
manipulate(
histogram(tgSpending$spending, breaks=x),
x = slider(4,20)
)
12:12
(Top Level)
R Script
Console ~/
> manipulate(
+ histogram(tgSpending$spending, breaks=x),
+ x = slider(4,20)
+)
> manipulate(
+ ggplot(tgSpending) + geom_histogram(aes(x=spending), binwidth=x),
+ x = slider(0,100, initial=10)
+)
> manipulate(
+ ggplot(tgSpending) + geom_histogram(aes(x=spending), binwidth=x),
+ x = slider(0,100, initial=10)
+)
>
Environment
History
Import Dataset
List
Global Environment
Data
tgSpending
436 obs. of 1 variable
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
50
40
30
20
10
0
count
0
100
200
spending

Exploring Histograms, an essay by Aran Lunzer and Amelia McNamara

Gather your data

A histogram is based on a collection of data about a numeric variable. Our first step is to gather some values for that variable. The initial dataset we will consider consists of fuel consumption (in miles per gallon) from a sample of car models available in 1974 (yes, rather out of date). We can visualize the dataset as a pool of items, with each item identified by its value—which in theory lets us "see" all the items, but makes it hard to get the gestalt of the variable. What are some common values? Is there a lot of variation?



Sort into an ordered list

A useful first step towards describing the variable's distribution is to sort the items into a list. Now we can see the maximum value and the minimum value. Beyond that, it is hard to say much about the center, shape, and spread of the distribution. Part of the problem is that the list is completely filled; the space between any two items is the same, no matter how dissimilar their values may be. We need a way to see how the items relate to each other. Are they clustered around a few specific values? Is there one lonely item, with a value far removed from all the others?

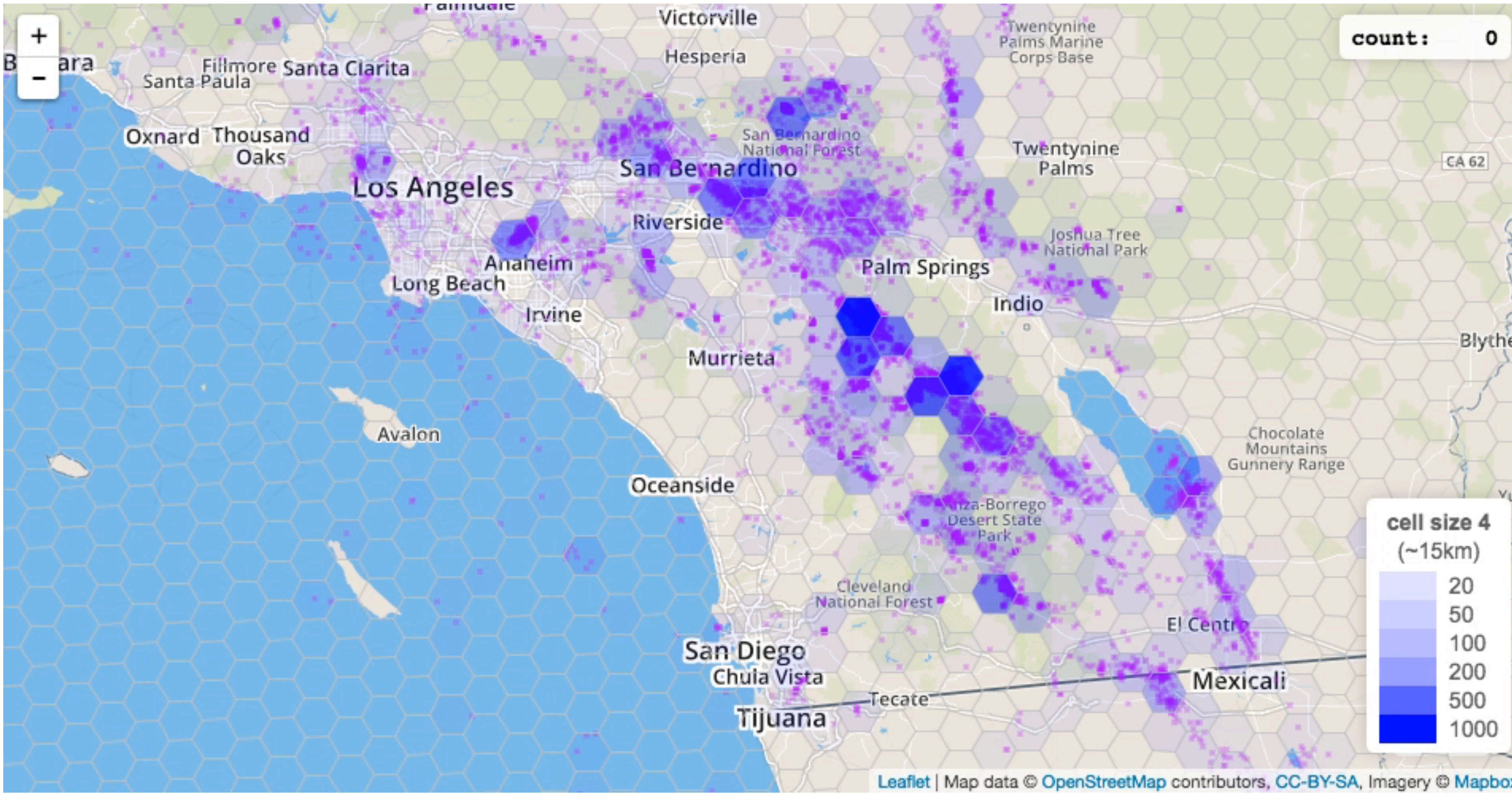
Draw the number line

A common convention is to use a number line, on which higher values are displayed to the right and smaller (or negative) values to the left. We can draw a line representing all possible numbers between the minimum and maximum data values.

Add data to the number line

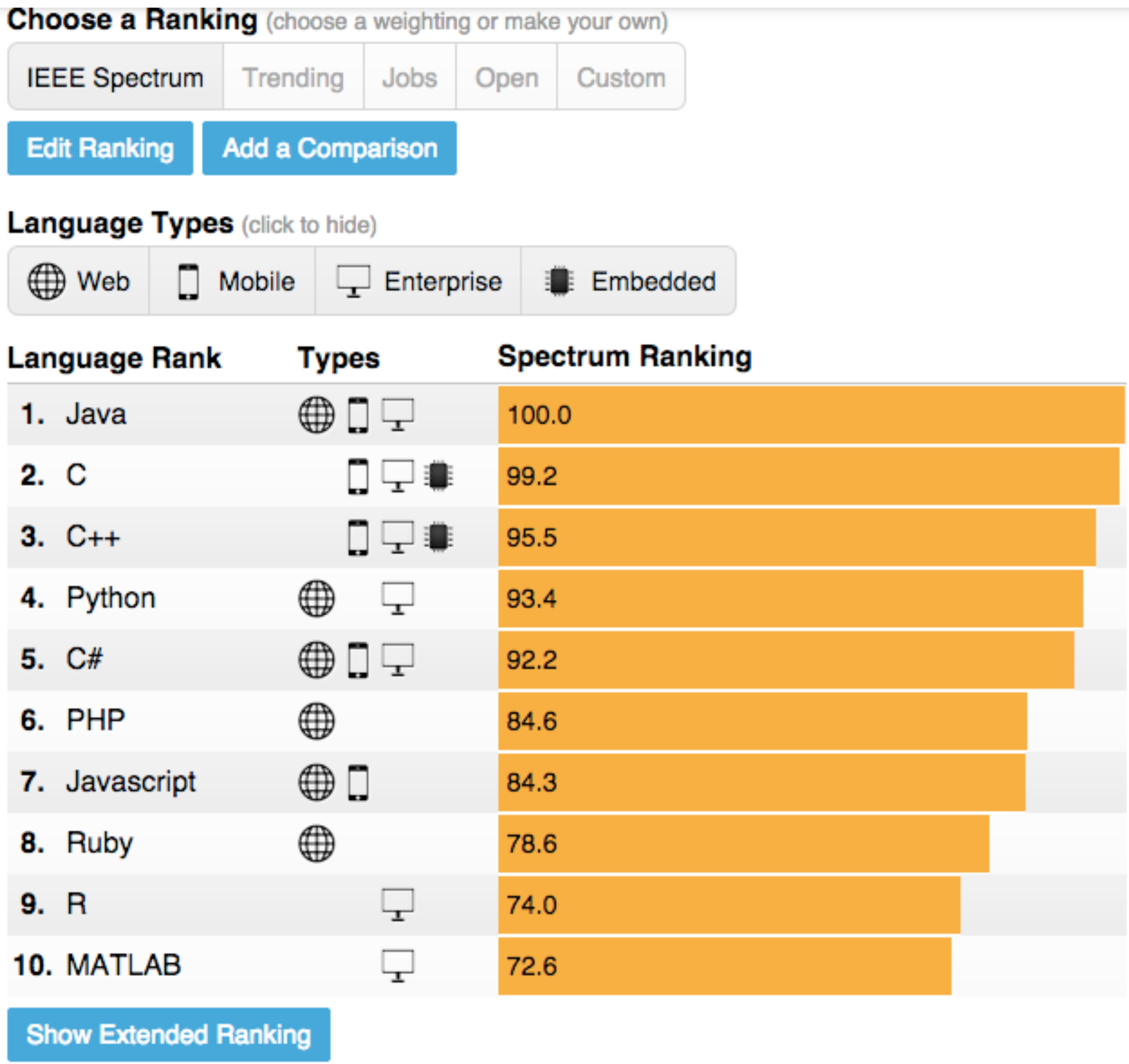
Now, we map each item to a dot at the appropriate point along the number line. In our visualization we draw the path followed by each item on its way from the list to the line, helping to reveal how adjacent list items end up close or far apart on the number line

Spatial aggregation toy



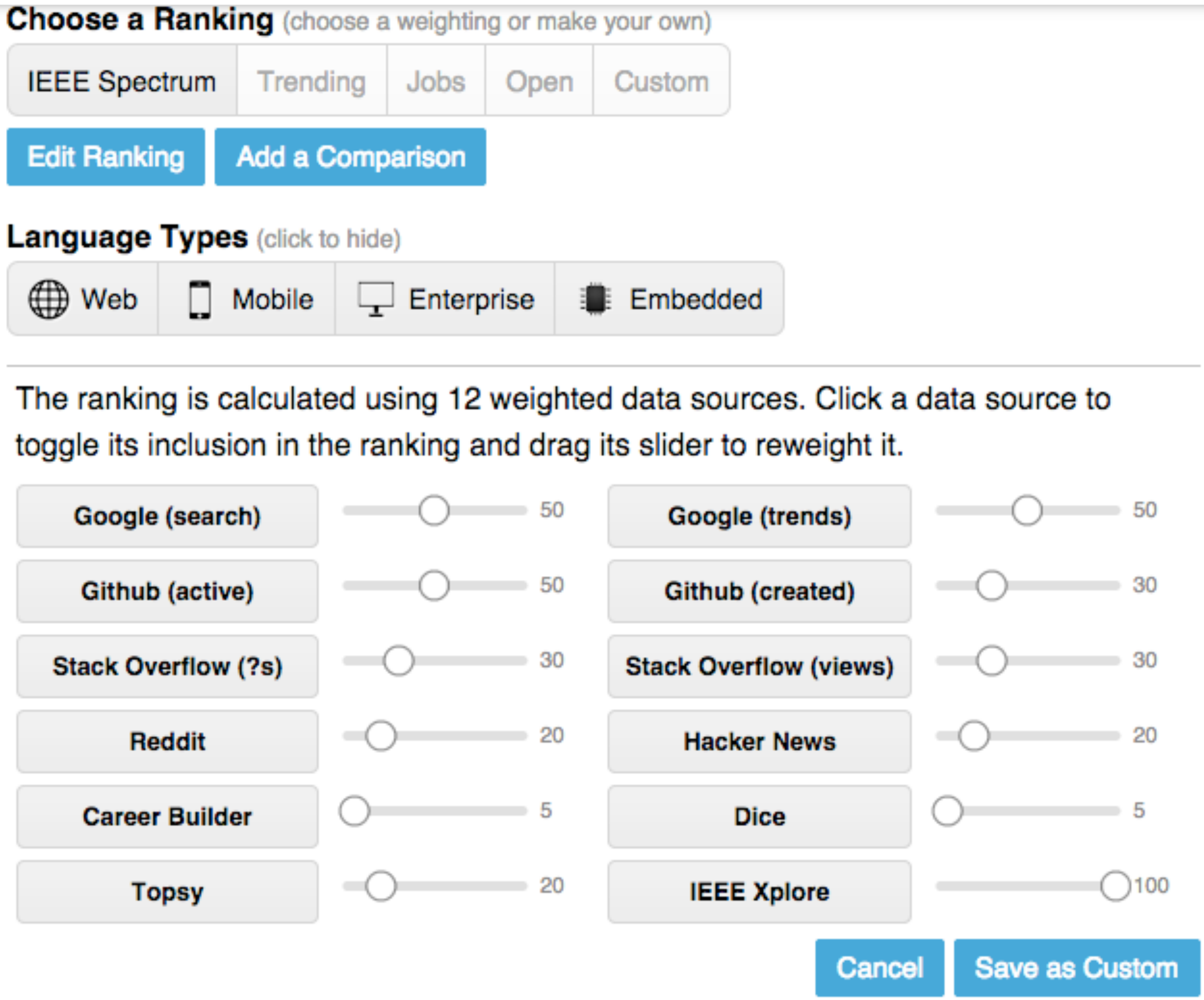
http://bit.ly/spatial_agg

Auditable products



Interactive: The Top Programming Languages 2017. Nick Diakopoulos and Stephen Cass
https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2017

Auditable products



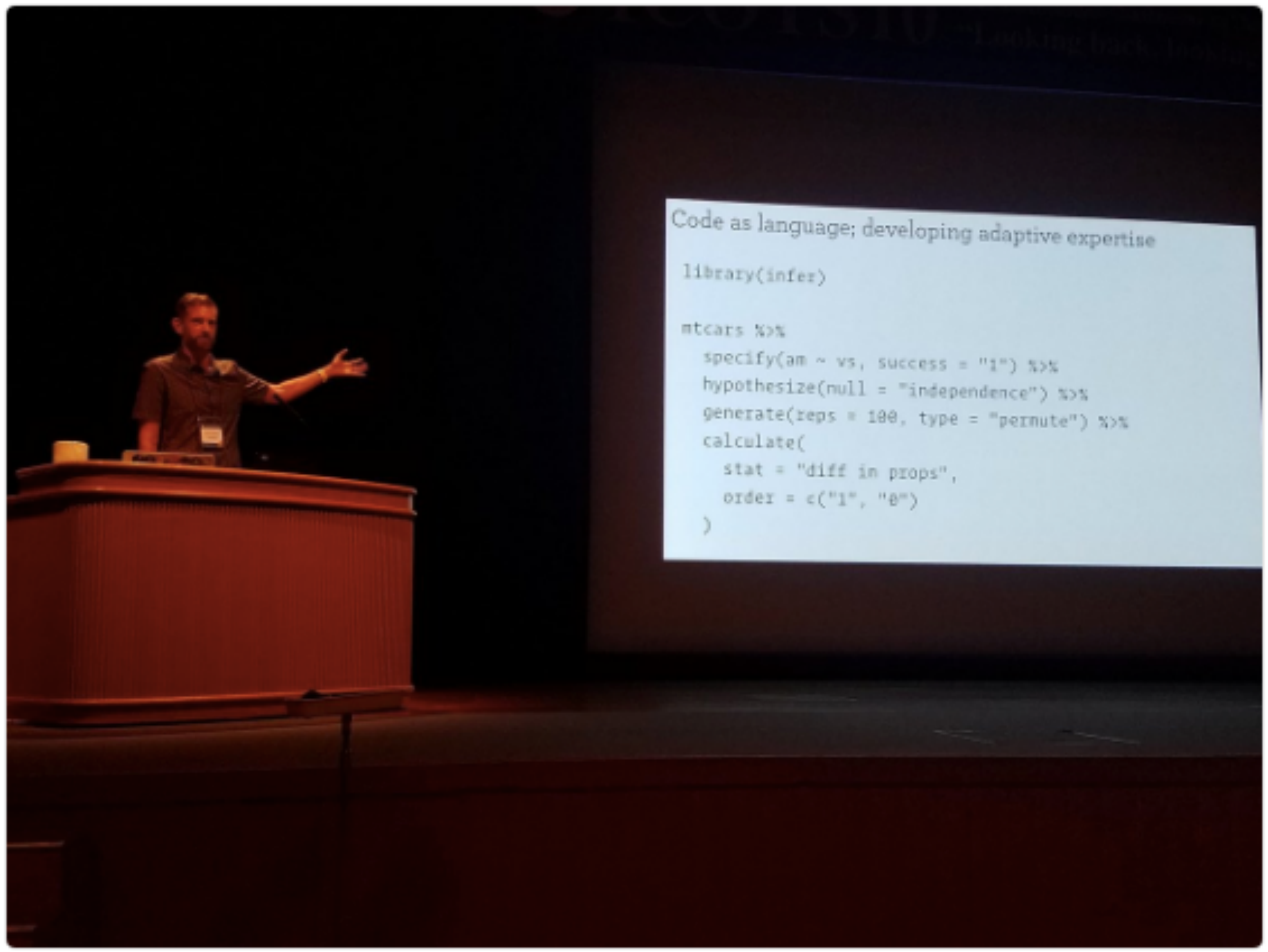
Interactive: The Top Programming Languages 2017. Nick Diakopoulos and Stephen Cass
https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2017

Inherent documentation

- The system or language should show or tell you what each function does
- Computing tools should "highlight the logic of what is going on" (Kaplan).

Amelia McNamara
@AmeliaMN

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



12:26 AM - 10 Jul 2018

30 Retweets **129** Likes

Support for narrative, publishing and reproducibility

- narrative and code should be intermixed
- publishing should support others reading your work
- reproducibility should be encouraged

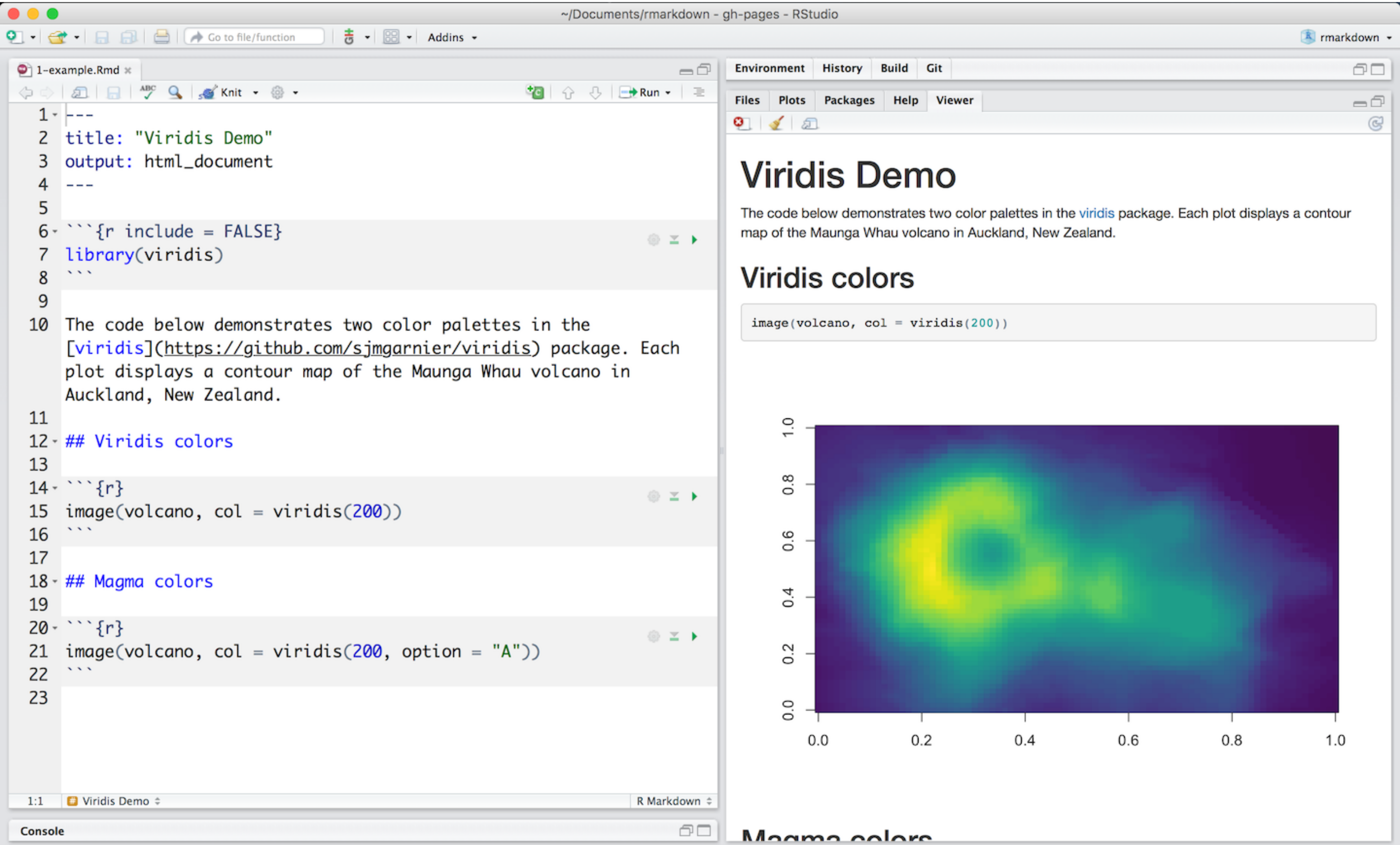
Amelia McNamara
@AmeliaMN

.@xieyihui has an evil exercise that seems torn from @kwbroman's emails: students do analysis, then send them the updated data. #JSM2016

9:15 AM - 3 Aug 2016

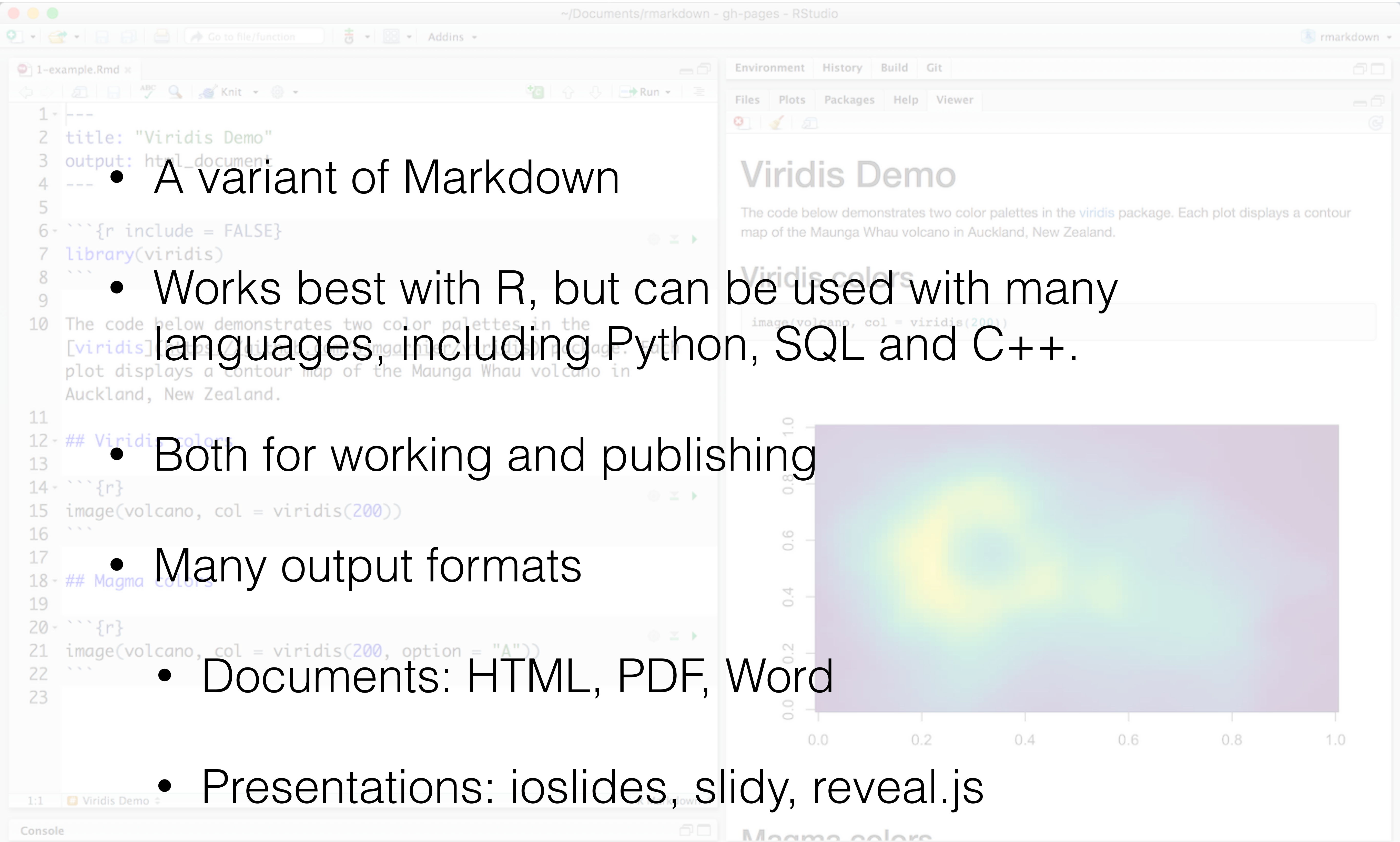
1 Retweet 4 Likes

RMarkdown



http://rmarkdown.rstudio.com/

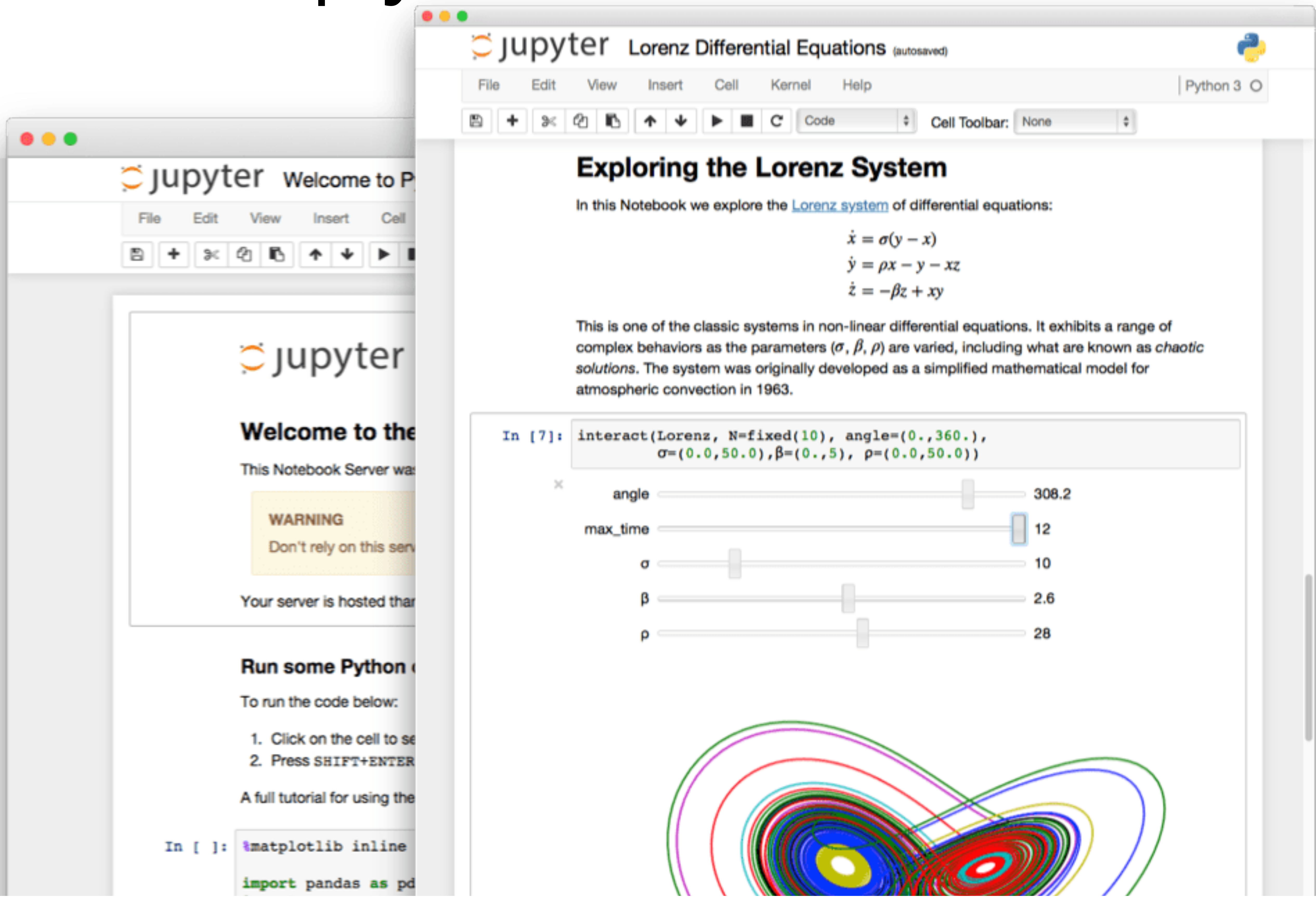
RMarkdown



- A variant of Markdown
- Works best with R, but can be used with many languages, including Python, SQL and C++.
- Both for working and publishing
- Many output formats
  - Documents: HTML, PDF, Word
  - Presentations: ioslides, slidy, reveal.js

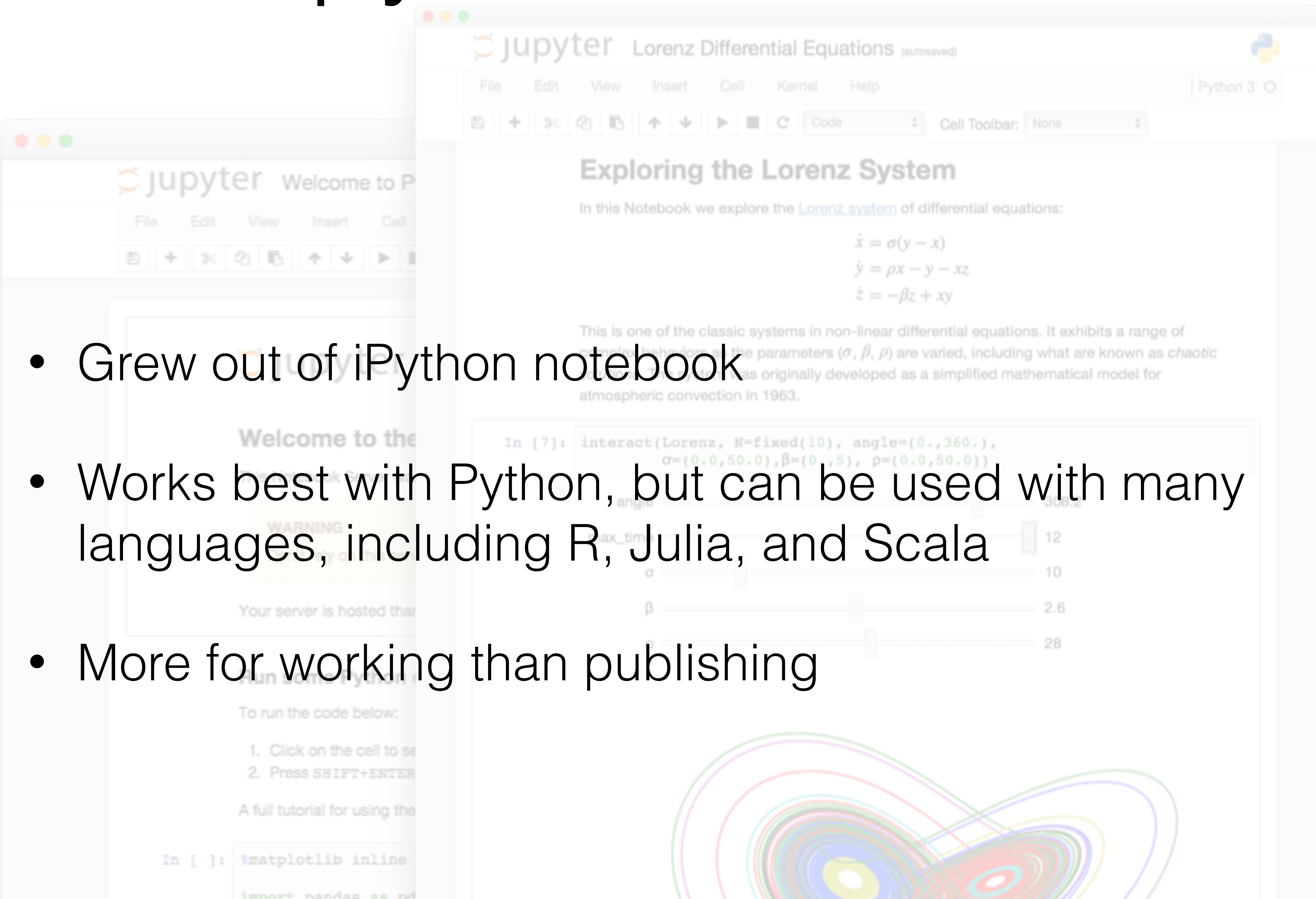
http://rmarkdown.rstudio.com/

Jupyter notebooks



http://jupyter.org/

Jupyter notebooks



- Grew out of iPython notebook
- Works best with Python, but can be used with many languages, including R, Julia, and Scala
- More for working than publishing

http://jupyter.org/

Our path to better science in less time using open data science tools

Julia Stewart Lowndes, et al. Nature Ecology & Evolution v1.
https://www.nature.com/articles/s41559-017-0160

We thought we were doing reproducible science. For the first global OHI assessment in 2012 we employed an approach to reproducibility that is standard to our field, which focused on scientific methods, not data science methods. Data from nearly one hundred sources were prepared manually—that is, without coding, typically in Microsoft Excel—which included organizing, transforming, rescaling, gap-filling and formatting data. Processing decisions were documented primarily within the Excel files themselves, e-mails, and Microsoft Word documents. We programmatically coded models and meticulously documented their development, (resulting in the 130-page supplemental materials), and upon publication we also made the model inputs (that is, prepared data and metadata) freely available to download. This level of documentation and transparency is beyond the norm for environmental science.

Our path to better science in less time using open data science tools. Julia Stewart Lowndes, et al. Nature Ecology & Evolution v1. https://www.nature.com/articles/s41559-017-0160

We decided to base our work in R and RStudio for coding and visualization, Git for version control, GitHub for collaboration, and a combination of GitHub and RStudio for organization, documentation, project management, online publishing, distribution and communication.

Data preparation: coding and documenting. Our first priority was to code all data preparation, create a standard format for final data layers, and do so using a single programmatic language, R. Code enables us to reproduce the full process of data preparation, from data download to final model inputs, and a single language makes it more practical for our team to learn and contribute collaboratively. We code in R and use RStudio to power our workflow because it has a user-friendly interface and built-in tools useful for coders of all skill levels, and, importantly, it can be configured with Git to directly sync with GitHub online (See 'Collaboration').

Sharing methods and instruction. We use R Markdown not only for data preparation but also for broader communication. R Markdown files can be generated into a wide variety of formatted outputs, including PDFs, slides, Microsoft Word documents, HTML files, books or full websites.

Our path to better science in less time using open data science tools. Julia Stewart Lowndes, et al. Nature Ecology & Evolution v1. https://www.nature.com/articles/s41559-017-0160

Flexibility to build extensions

- Biehler called this the "closed microworld problem"
- CS ed literature calls this a "high ceiling"

Extensible—

features 12744 available packages.

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

Available Packages

Currently, the CRAN package repository features 12744 available packages.

Table of available packages, sorted by date of publication

Table of available packages, sorted by name

Installation of Packages

Please type `help("INSTALL")` or `help("install.packages")` in R for information on how to install packages from this repository. The manual R Installation and Administration (also contained in the R base sources) explains the process in detail.

CRAN Task Views allow you to browse packages by topic and provide tools to automatically install all packages for special areas of interest. Currently, 36 views are available.

Package Check Results

All packages are tested regularly on machines running Debian GNU/Linux, Fedora, OS X, Solaris and Windows.

The results are summarized in the check summary (some timings are also available). Additional details for Windows checking and building can be found in the Windows check summary.

Writing Your Own Packages

The manual Writing R Extensions (also contained in the R base sources) explains how to write new packages and how to contribute them to CRAN.

Repository Policies

The manual CRAN Repository Policy [PDF] describes the policies in place for the CRAN package repository.

Related Directories

Archive
    Previous versions of the packages listed above, and other packages formerly available.
Orphaned
    Packages with no active maintainer, see the corresponding README.

https://cran.r-project.org/

Extensible— GP

GP

About

Gallery

Run GP!

Resources

Docs

Forum

Download

About

Frequently Asked Questions
Team and Credits

GP is a free, general-purpose blocks language that is powerful yet easy to learn.

GP can:

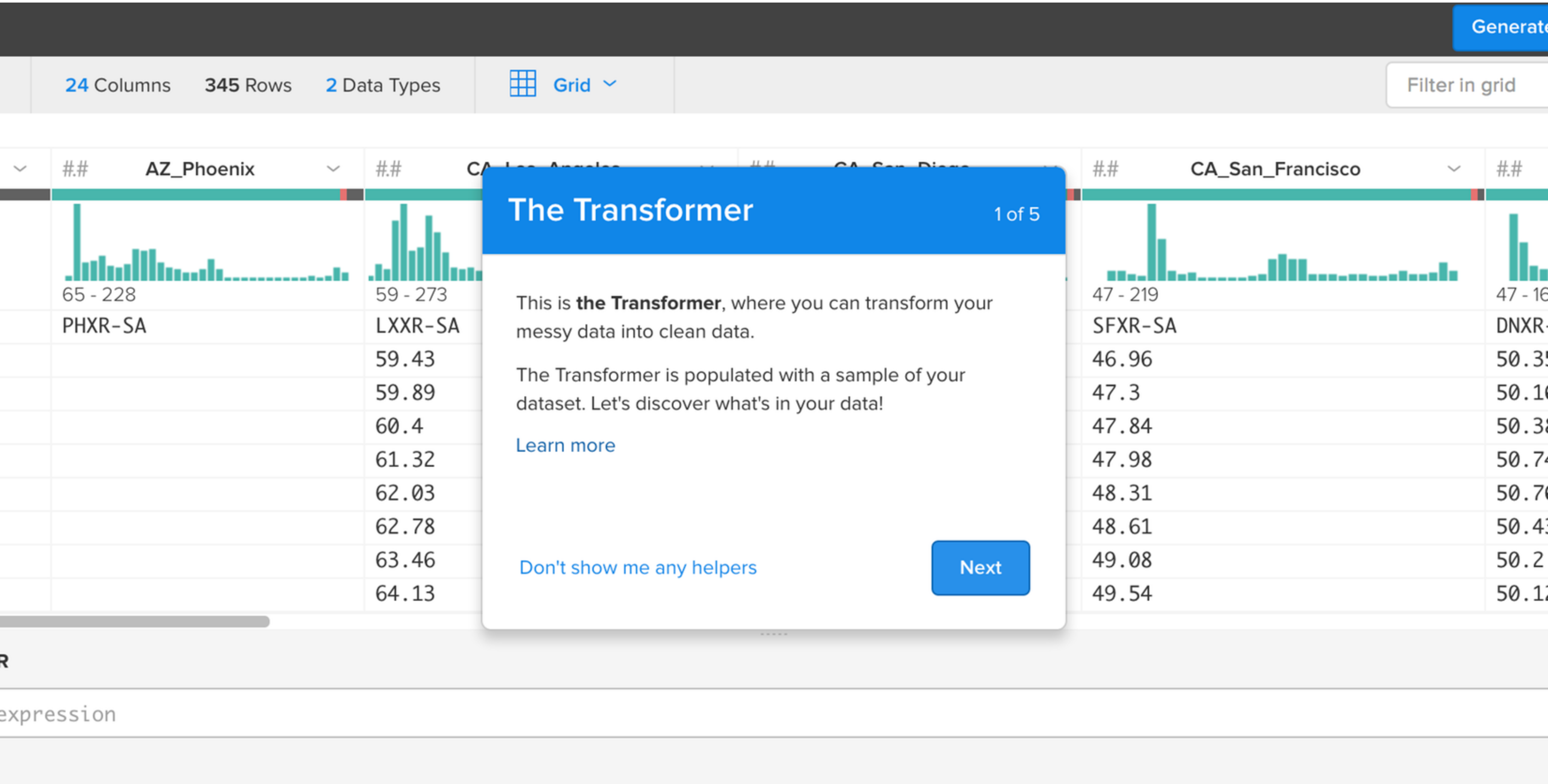
- generate high-quality graphics computationally
- manipulate images and sounds
- analyze text files or CSV data sets
- simulate physical, biological, or economic systems
- access the web and use cloud data
- connect to hardware via the serial port
- deploy projects on the web or as stand-alone apps

https://gpblocks.org/about/

Looking into the future

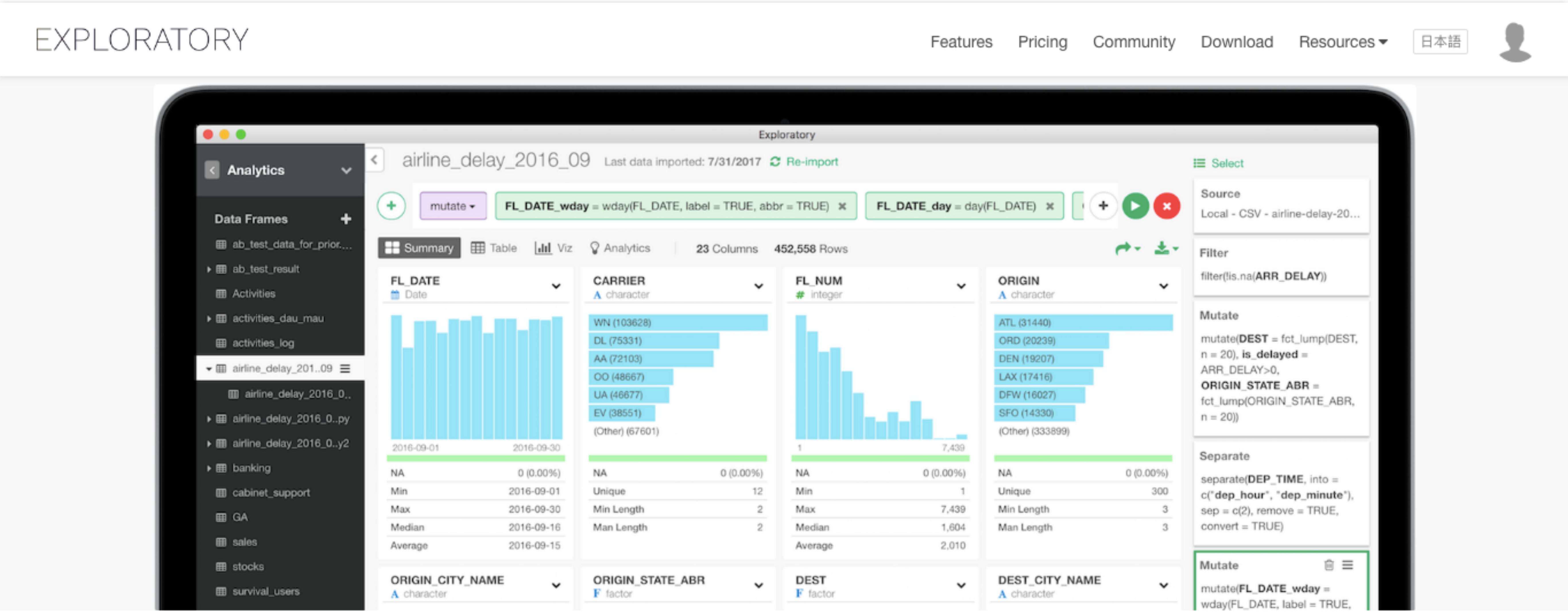
flickr: takchabo

Trifacta



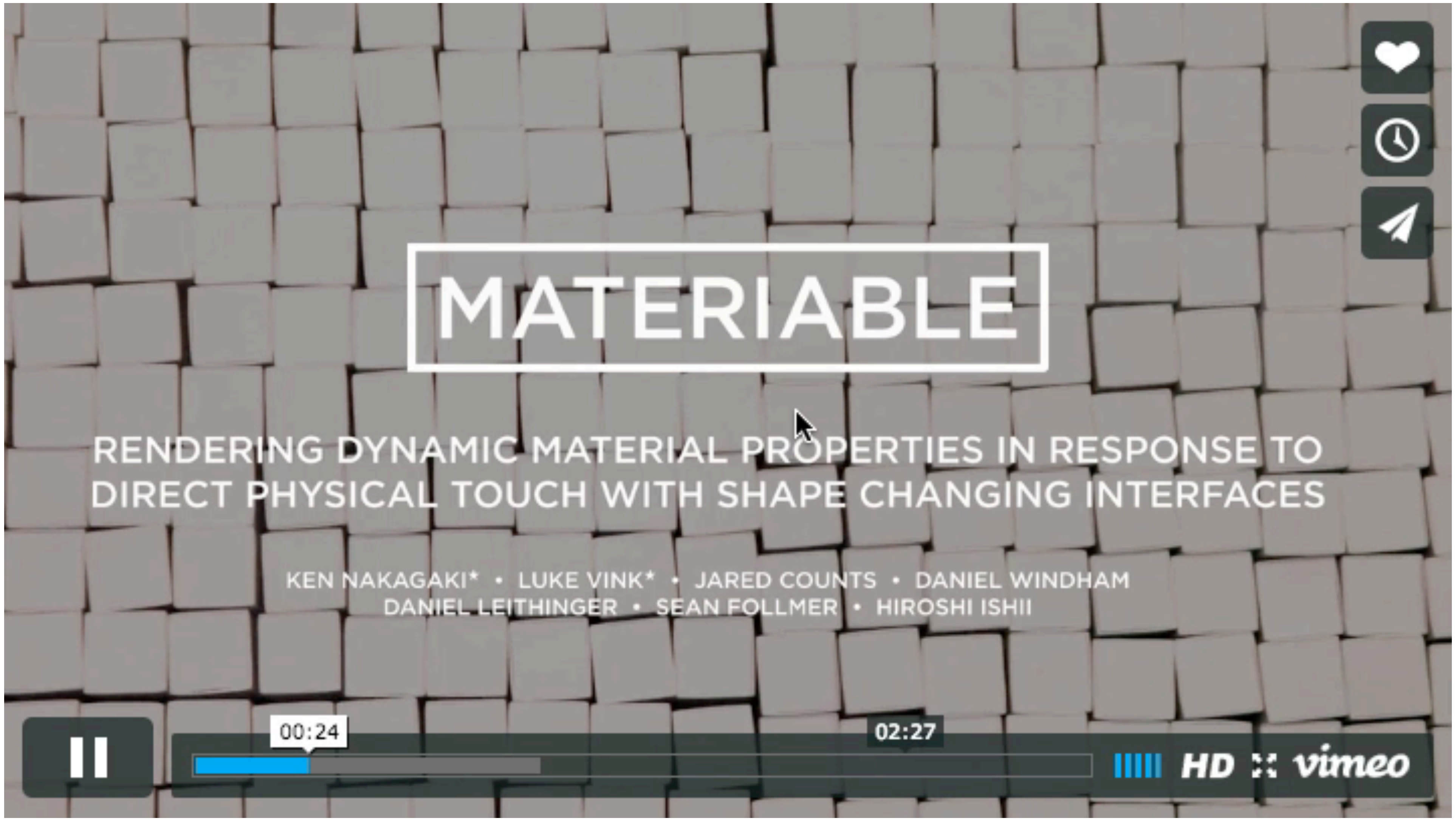
https://www.trifacta.com/

Exploratory



https://exploratory.io/

Trusted By



Ken Nakagaki, Luke Vink, Jared Counts, Daniel Windham, Daniel Leithinger, Sean Folder and Hiroshi Ishii. Materiable: Rendering Dynamic Material Properties in Response to Direct Physical Touch with Shape Changing Interfaces CHI 2016. http://tangible.media.mit.edu/project/materiable/

Bret Victor. Seeing Spaces.
http://worrydream.com/#!/SeeingSpaces

Thank you

Amelia McNamara (@AmeliaMN)
Assistant Professor, Department of Computer & Information Sciences
University of St Thomas, St Paul, MN USA

flickr: takchabo