rstudio::conf(2017L)

#recap

LAWRENCE WU // 2017-01-26

GOAL FOR TODAY

- Given the nature of this talk, I can't go in depth into any one topic. There will be many links to presentations and resources
- "Sampler platter" - you need to figure out what entree you want to order
- So the takeaway: pick 2 or 3 topics / packages that you want to investigate further

OPINIONATED ANALYSIS DEVELOPMENT

IN THE LIFE OF AN "ANALYSIS DEVELOPER" (HILARY PARKER)

- Creating analysis is hard and error-ridden. We don't talk much about the process right now
- There are known, common problems when creating an analysis (as well as solutions to these problems)
- "Opinionated" - have an opinion on what software solution is best to solve the problem
- Making these easy to avoid frees up time for creativity

See slides: http://www.slideshare.net/hilaryparker/opinionated-analysis-development

Reproducible	You re-run the analysis and get different results.
	Someone else can't repeat the analysis.
	You can't re-run the analysis on different data.
	An external library you're using is updated, and you can't recreate the results.
	You change code but don't re-run downstream code, so the results aren't consistently updated.
	You change code but don't re-execute it, so the results are out of date.
	You update your analysis and can't compare it to previous results.
	You can't point to code changes that resulted in a different analysis results.
	A second analyst can't understand your code.
Accurate	Can you re-use logic in different parts of the analysis?
	You change the logic used in analysis, but only in some places where it's implemented.
	Your code is not performing as expected, but you don't notice.
	Your data becomes corrupted, but you don't notice.
	You use a new dataset that renders your statistical methods invalid, but you don't notice.
	You make a mistake in your code.
	You use inefficient code.
Collaborative	A second analyst wants to contribute code to the analysis, but can't do so.
	Two analysts want to combine code but cannot.
	You aren't able to track and communicate known next steps in your analysis.
	Your collaborators can only make requests in email or meetings, and they aren't incorporated into the project.

“Opinions”

Executable analysis scripts	You re-run the analysis and get different results.
	Someone else can’t repeat the analysis.
	You can’t re-run the analysis on different data.
Defined dependencies	An external library you’re using is updated, and you can’t recreate the results.
	You change code but don’t re-run downstream code, so the results aren’t consistently updated.
Watchers for changed code and data	You change code but don’t re-execute it, so the results are out of date.
Version control (individual)	You update your analysis and can’t compare it to previous results.
	You can’t point to code changes that resulted in a different analysis results.
Code review	A second analyst can’t understand your code.
Modular, tested code	Can you re-use logic in different parts of the analysis?
	You change the logic used in analysis, but only in some places where it’s implemented.
	Your code is not performing as expected, but you don’t notice.
Assertive testing of data,	Your data becomes corrupted, but you don’t notice.
assumptions and results	You use a new dataset that renders your statistical methods invalid, but you don’t notice.
Code review	You make a mistake in your code.
	You use inefficient code.
Version Control (collaborative)	A second analyst wants to contribute code to the analysis, but can’t do so.
	Two analysts want to combine code but cannot.
Issue tracking	You aren’t able to track and communicate known next steps in your analysis.
	Your collaborators can only make requests in email or meetings, and they aren’t incorporated into the project.

OPINIONATED ANALYSIS DEVELOPMENT

- Define the process of technically creating an analysis
- Define opinions based on common process errors
- Push for software that makes it easy to implement opinions
- Will allow you to focus on creativity

- Some "opinions" on how to do data analysis in R will follow...

TIDYVERSE

FOUR FACTS ABOUT TIDY VERSE AND A DEFINITION (HADLEY WICKHAM)

- It exists (~~Hadleyverse~~)
- There's a website, http://tidyverse.org/
- Tidy verse book, "R for Data Science" http://r4ds.had.co.nz/
- There's a package, `library("tidyverse")` will load all the core packages
- See the manifesto
- Definition: "The tidyverse is a collection of R packages that share common philosophies and are designed to work together. "
- If you're new to R, I would start here, learning tidy verse principles will make your life a lot easier

TIDY VERSE PRINCIPLES

- Reuse existing data structures - data frames, tidy data
- Compose simple functions with the pipe
- Embrace functional programming
- Design for humans

REUSE EXISTING DATA STRUCTURES

- Generally, it's better to prefer common existing data structures over custom data structures, even if slightly ill-fitting
- Most R packages nowadays input and ouput rectangular datasets where each column is a variable and each row is a observation. This is the "tidy" data format. See the tidy data paper
- With tidy data, it makes more down-stream processing e.g. plotting with ggplot2 much easier

COMPOSE SIMPLE FUNCTIONS WITH A PIPE

- *"No matter how complex and polished the individual operations are, it is often the quality of the glue that most directly determines the power of the system." — Hal Abelson*
- Functions you write should do **one** thing **well**, then you can use them in pipe chains

PIPES IN A NUTSHELL

- `x %>% f` is equivalent to `f(x)`
- `x %>% f(y, .)` is equivalent to `f(y, x)`
- Makes your code much more readable and intuitive
- See Bob Rudis' Mario themed [presentation on pipes]

```
monthly_originations <-
  all_loans %>%
  filter(ExecutionDate > '2016-01-01') %>%
  group by(ExecutionDate) %>%
  summarise(monthly_origs = sum(Originations))
```

LES RÈGLES DE LA TUYAUTERIE DE hrbrmstr

(hrbrmstr's Rules of Piping)

- The chain should be > 1
- A pipe group should be designed to accomplish a unified task
- It's OK to change object class/type/mode
- Be data-source aware
- Pipe operations should be "atomic"
- Pipe (briefly) in pipes
- Don't be reticent to create new verbs
- Keep them **short**

EMBRACE FUNCTIONAL PROGRAMMING

- R is a functional programming language - meaning it provides tools for creating and manipulating functions
- Should favor tools that abstract over for-loops
  - apply
  - map functions in purrr
- See Functional Programming chapter in Advanced R

PURRR IN A NUTSHELL

- https://github.com/hadley/purrr
- Charlotte Wickham's presentation: Happy R Users Purrr
- map is similar to apply but will always return the same type

```
library(purrr)
mtcars %>%
  split(.$cyl) %>%
  map(~ lm(mpg ~ wt, data = .)) %>%
  map(summary) %>%
  map_dbl("r.squared")
```

DESIGN FOR HUMANS

- Computer efficiency is a secondary concern because the bottleneck in most data analysis is thinking time, not computing time.

> *Programs are meant to be read by humans and only incidentally for computers to execute.*
>
> —Knuth

EXTENSIONS TO THE "TIDYVERSE"

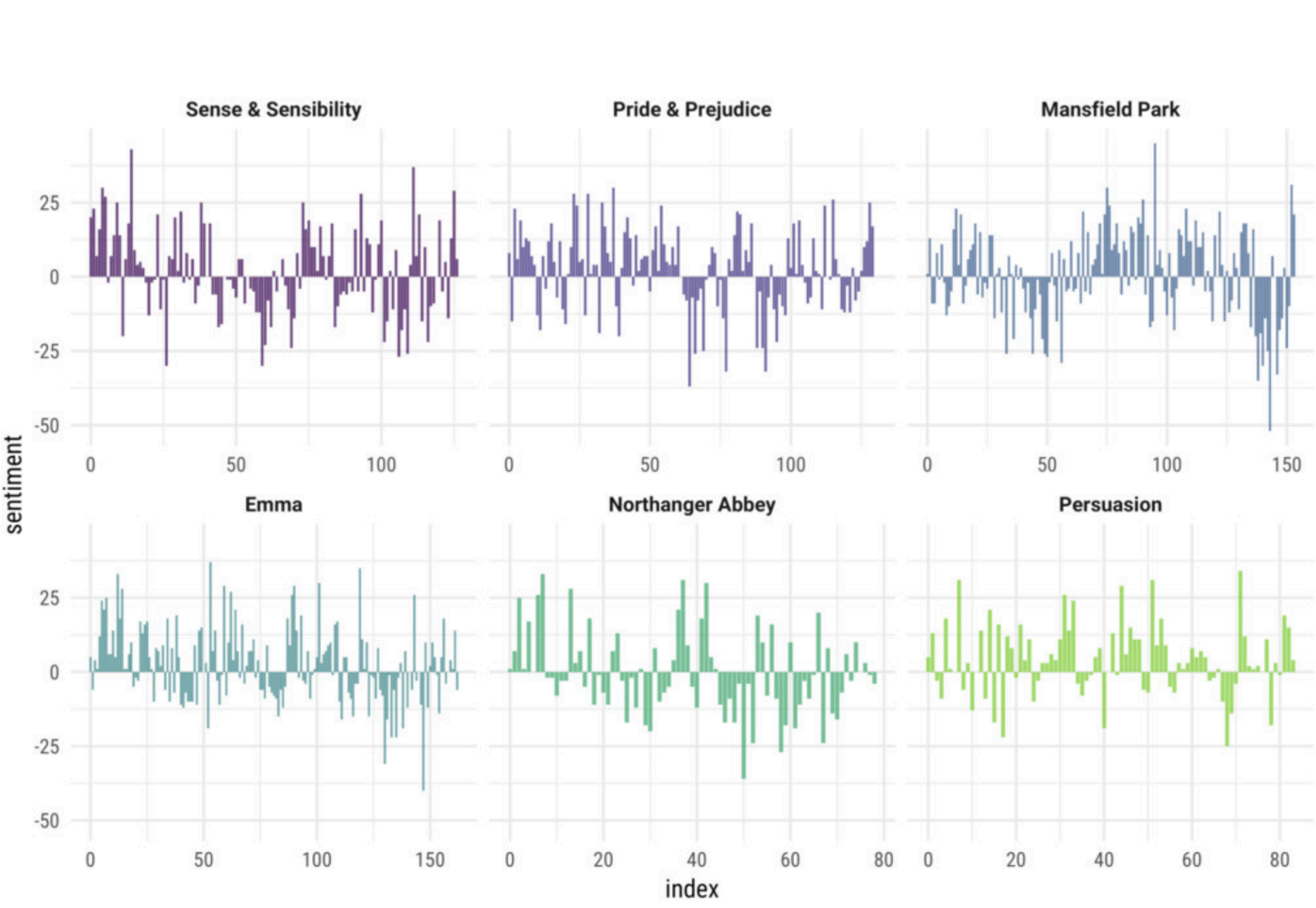
- Many R package developers make their packages with the tidy verse in mind
- Some examples of packages
  - tidytext - "Text Mining the Tidy Way"
  - corrr - "Exploring correlations in a tidy R framework"
  - broom - "tidying" model output

Tidying the works of Jane Austen

```
> tidy_books <- original_books %>%
    unnest_tokens(word, text)
>
> tidy_books
# A tibble: 725,054 × 4
                   book linenumber chapter         word
                 <fctr>      <int>   <int>        <chr>
1  Sense & Sensibility          1       0        sense
2  Sense & Sensibility          1       0          and
3  Sense & Sensibility          1       0  sensibility
4  Sense & Sensibility          3       0           by
5  Sense & Sensibility          3       0         jane
6  Sense & Sensibility          3       0       austen
7  Sense & Sensibility          5       0         1811
8  Sense & Sensibility         10       1      chapter
9  Sense & Sensibility         10       1            1
10 Sense & Sensibility         13       1          the
# ... with 725,044 more rows
```

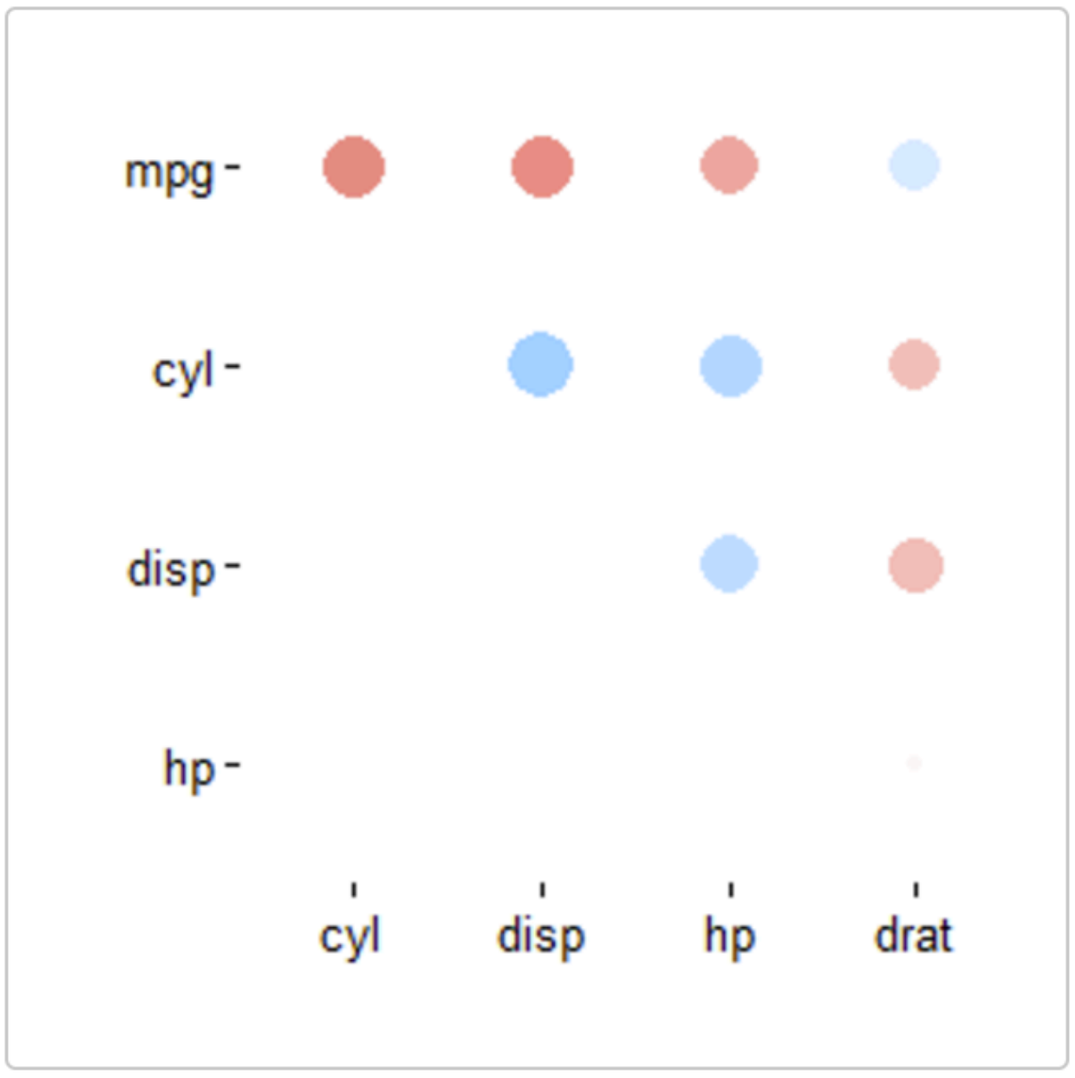
Sentiment analysis

```
> library(tidyr)
>
> janeaustensentiment <- tidy_books %>%
inner_join(get_sentiments("bing")) %>%
count(book, index = linenumber %/% 100, sentiment) %>%
spread(sentiment, n, fill = 0) %>%
mutate(sentiment = positive - negative)
```


Sense & Sensibility
Pride & Prejudice
Mansfield Park
Emma
Northanger Abbey
Persuasion
sentiment
index
25
0
-25
-50
0
50
100
150
20
40
60
80

CORRR PACKAGE

```
d %>%
  focus(mpg:drat, mirror = TRUE) %>%
  shave(upper = FALSE) %>%
  rplot()     # Plot
```



ASIDE: DPLYR VS DATA.TABLE

- data.table is very performant
- dtplyr is a dplyr backend for data.table (not that popular)
- dplyr solves the bottleneck of human thinking while data.table is better for computational speed since it does not make copies of data frames

RMARKDOWN

RMARKDOWN

- blogdown - build a static-based website based on Hugo written in R!
- xaringan - ports the remark.js library for slideshows, looks really neat
- Advanced R Markdown - really in-depth treatment of how knitting works, all the different things you can tweak - *slides were written in R - reproducible slides! (demo in RStudio)*

VISUALIZATION

THREEJS – 3D VISUALIZATION

- https://github.com/bwlewis/rthreejs
- 3D plots that are actually htmlwidgets - Amazing how fast these plots render. Based on the javascript library threejs
  - graphjs: an interactive force directed graph widget
  - scatterplot3js: a 3-d scatterplot widget similar to the scatterplot3d function
  - globejs: a widget that plots data and images on a 3-d globe
- See this blog for examples: http://bwlewis.github.io/rthreejs/ (demo)

gapminder_lifeexp

life expectancy vs. year by country using Gapmin...

1 - 12 of 142

Prev Next First Last

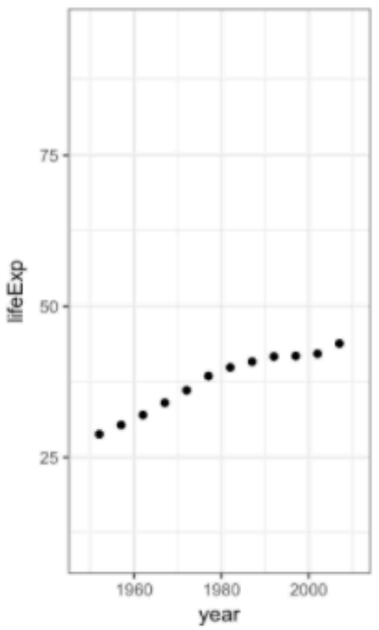
Trelliscope

Grid

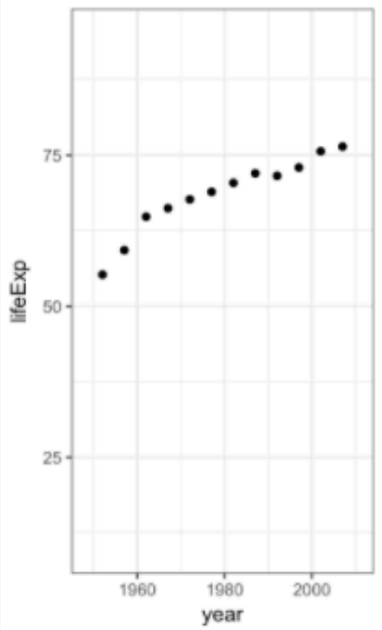
Labels

Filter

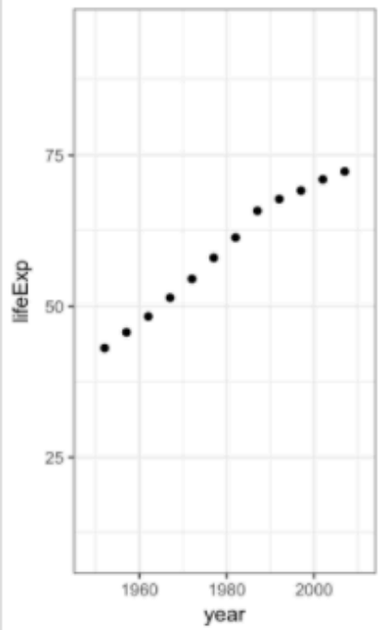
Sort



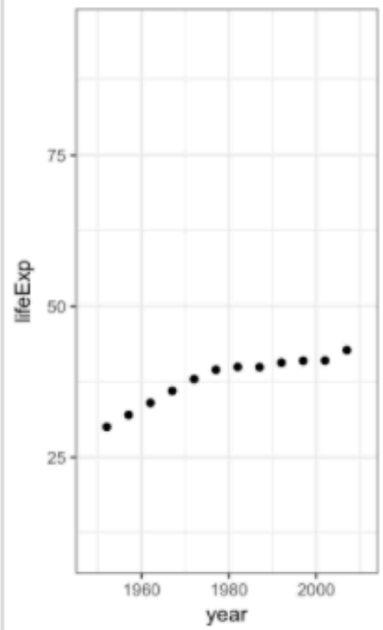
country Afghanistan
continent Asia



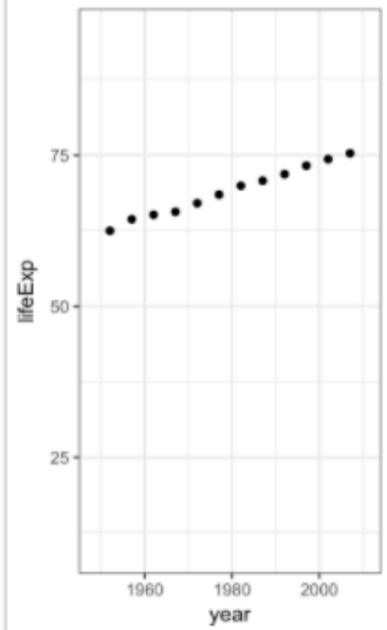
country Albania
continent Europe



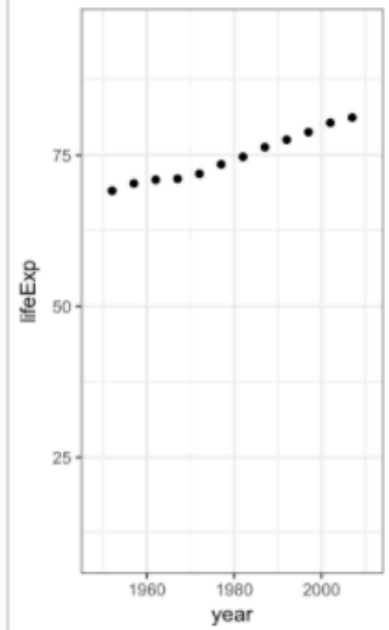
country Algeria
continent Africa



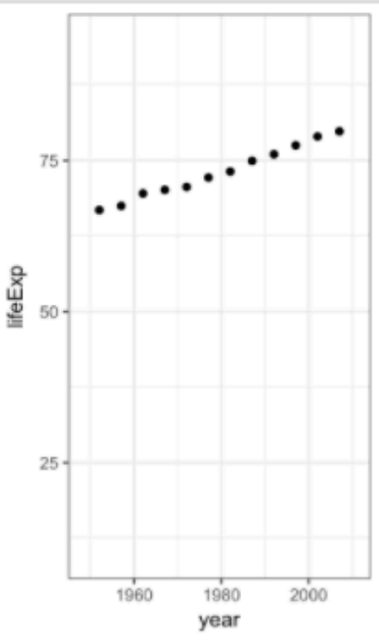
country Angola
continent Africa



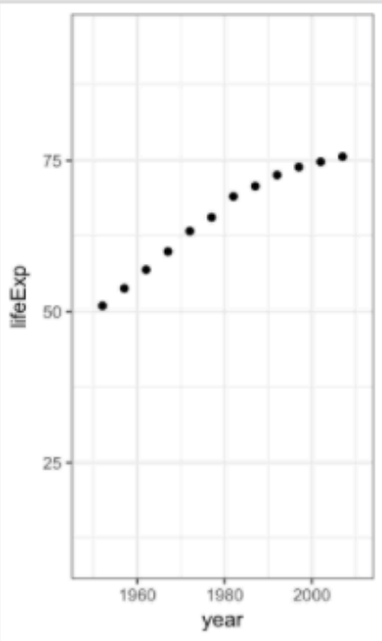
country Argentina
continent Americas



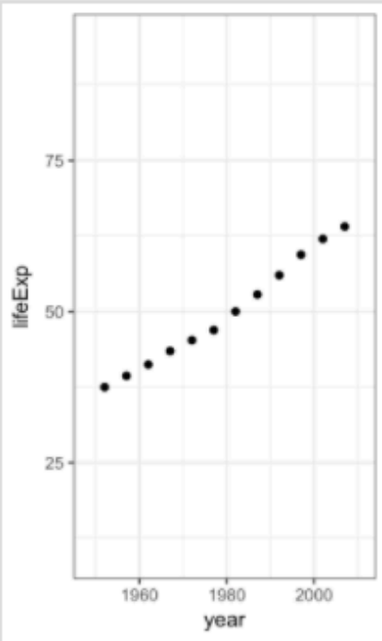
country Australia
continent Oceania



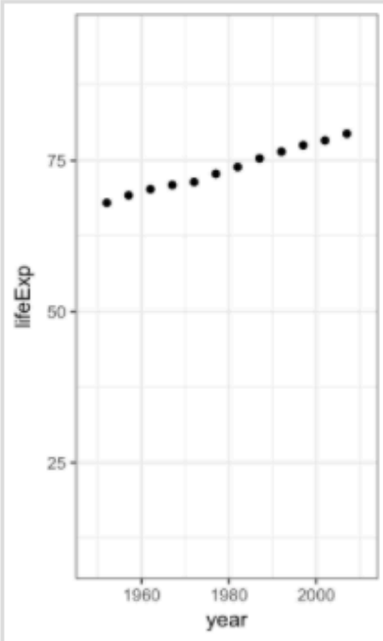
country Austria
continent Europe



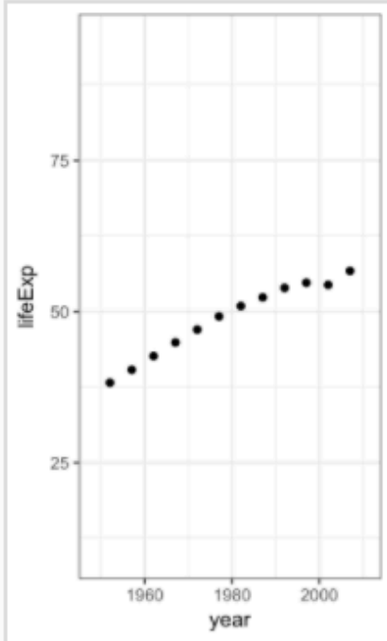
country Bahrain
continent Asia



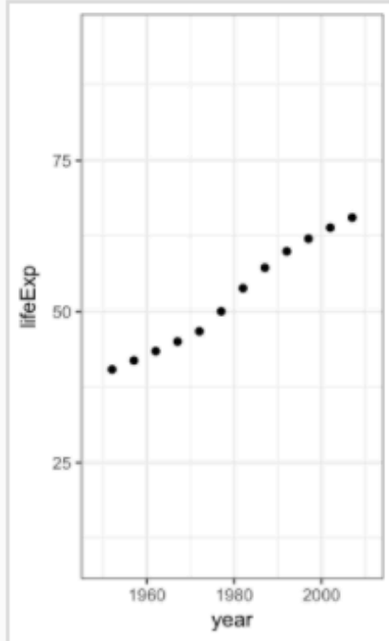
country Bangladesh
continent Asia



country Belgium
continent Europe



country Benin
continent Africa



country Bolivia
continent Americas

Sorting on: country continent

TRELLISCOPEJS DEMO

- Demo with housing data (2008-2016) - bottom of this blog post: http://ryanhafen.com/blog/trelliscopejs

SHINY

THERE'S A LOT...

- Joe Cheng's two-day tutorial: https://github.com/jcheng5/rstudio2017-shiny-workshop
- Shiny dashboards: https://github.com/jcheng5/dashtutorial
- Dynamic Shiny interfaces - presentation link
- shinytest - automated testing for Shiny apps

EVERYTHING ELSE

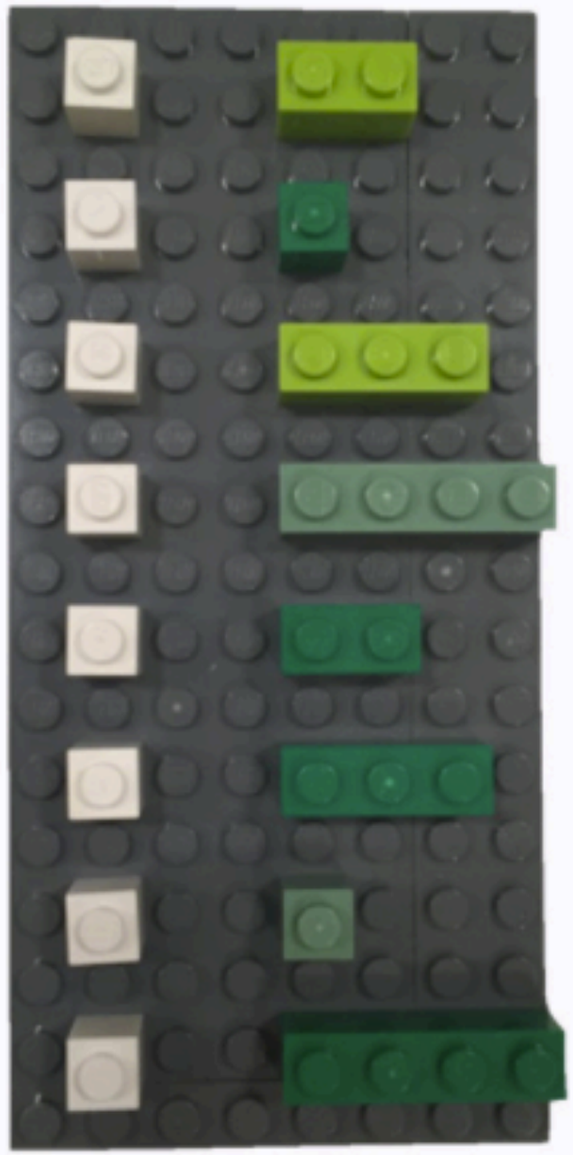
PROFILING R CODE WITH PROFVIS

- https://rstudio.github.io/profvis/
- Profile R code for time and memory usage (instead of wrapping code in system.time)
- Demo in RStudio

JENNY BRYAN'S TALKS

- git tutorial - Happy Git and Github for the useR - if you aren't already using git...
- list-columns - lego themed presentation
- useful if you do a lot of string processing
- JSON/XML data

```
name   stuff
<chr>  <list>
```



this **is** a data frame!

a tibble, specifically

https://github.com/timelyportfolio/listviewer

OPINIONS ON THE R LANGUAGE

- During Q&A -- "Limitations of R?"
  - J.J. - "The precursor to R was S which was just an interface to the best algorithms at the time which were written in Fortran"
  - Joe Cheng - "Python is boring but predictable. You can be a lot more expressive in R so it can be surprising. Go has a better balance"
  - Hadley - "One of the best ways to analyze and visualize data in memory"
  - Nowadays - the best algorithms are written in C, Spark, etc. and R will usually have access to these libraries, a R developer just needs to create the interface:
    - See sparklyr - http://spark.rstudio.com/
    - See tensorflow package - https://github.com/rstudio/tensorflow
- I love how expressive and human readable R is - facilitates learning so much faster

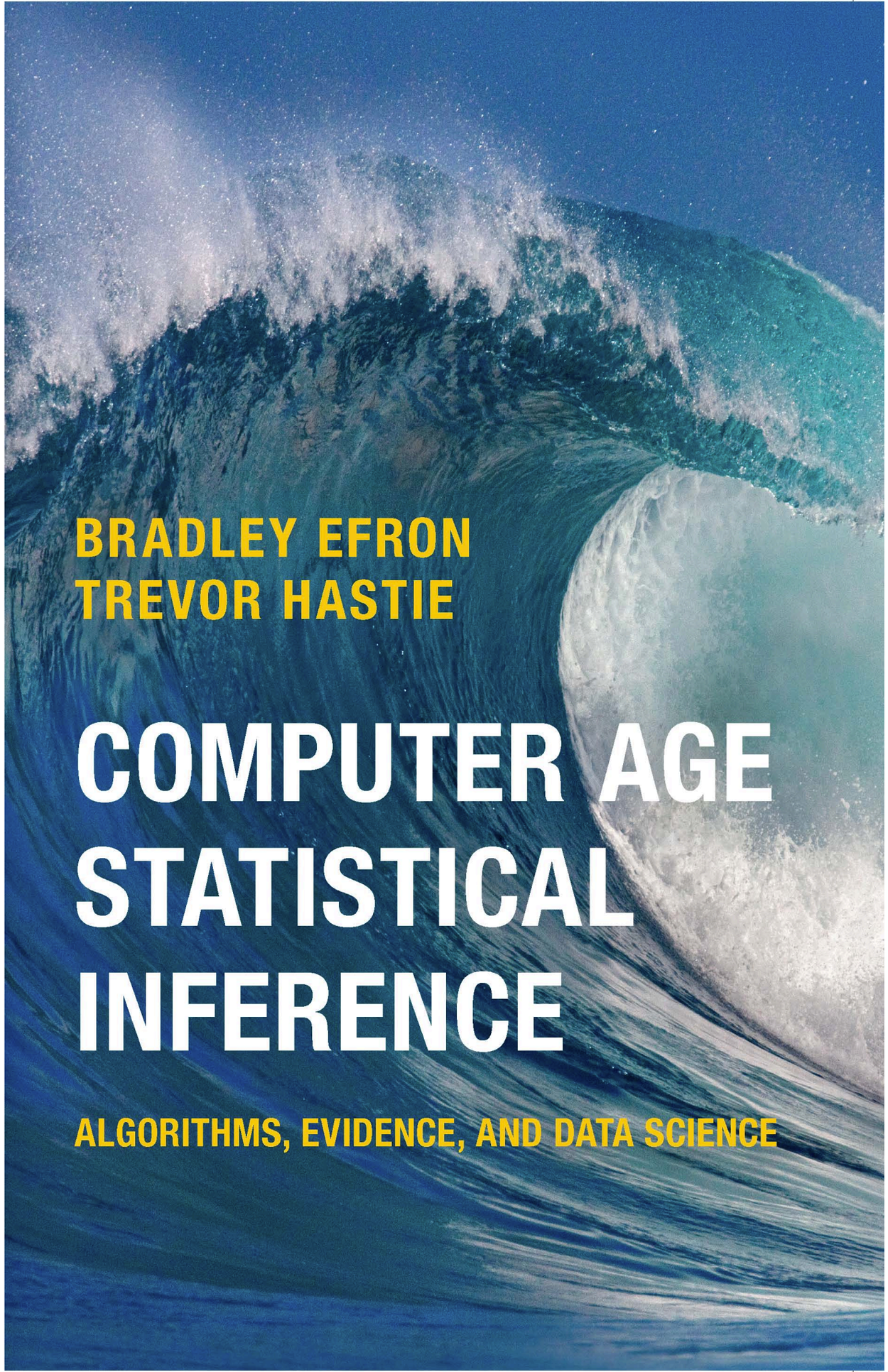
The R Series

Extending R

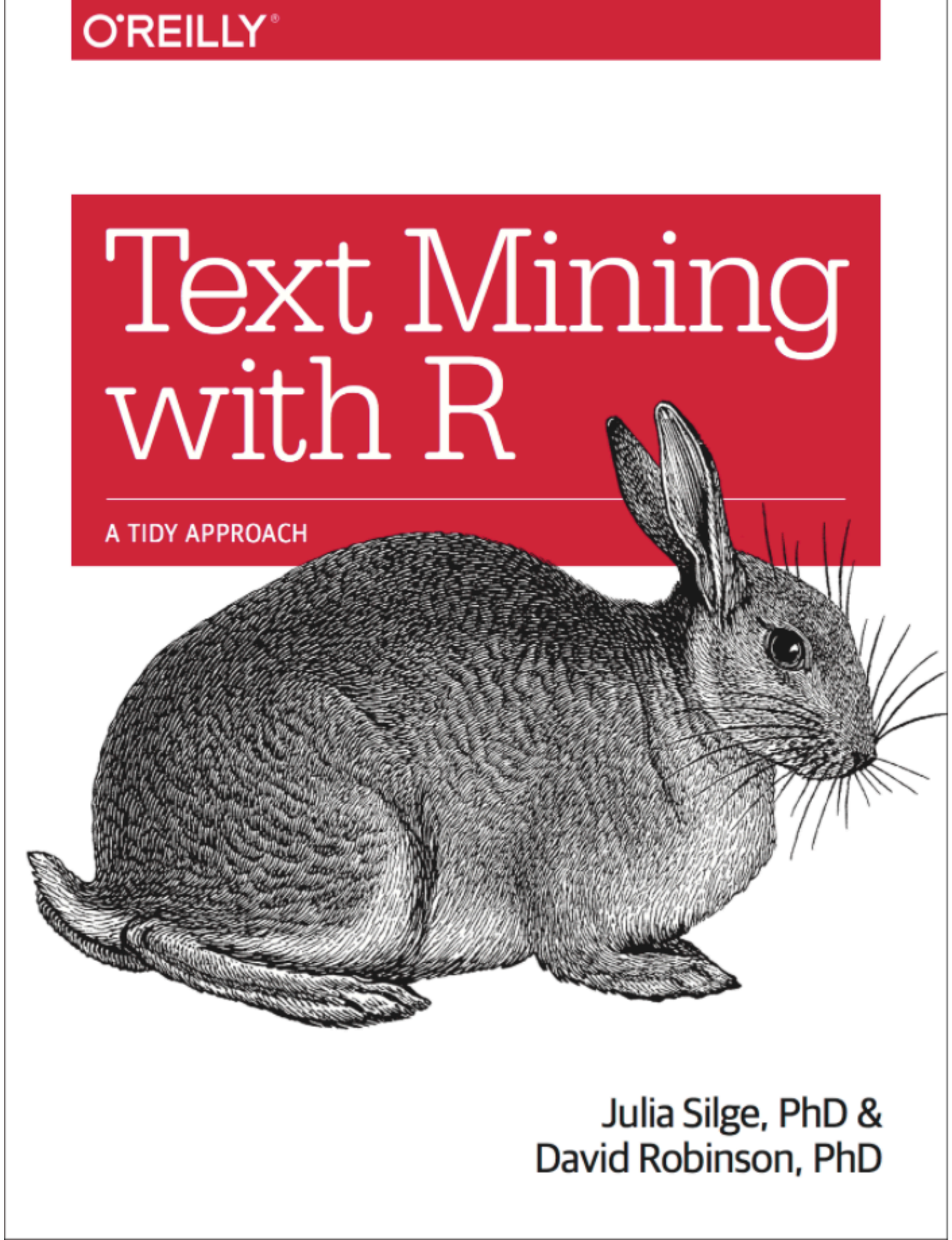
John M. Chambers

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

OTHER BOOKS



This book takes us on a journey through the revolution in data analysis following the introduction of electronic computation in the 1950s. Beginning with classical inferential theories – Bayesian, frequentist, Fisherian – individual chapters take up a series of influential topics: survival analysis, logistic regression, empirical Bayes, the jackknife and bootstrap, random forests, neural networks, Markov chain Monte Carlo, inference after model selection, and dozens more.



Free: http://tidytextmining.com/

#RSTATS RESOLUTIONS

- Important not to be overwhelmed
- What are yours?
- TODOs
  - Read Efron/Hastie's new book
  - Start a blog with blogdown package
  - Explore trelliscopejs

RSTUDIO CONF MATERIALS

- Karl Broman's repo has the best list of links to slides: https://github.com/kbroman/RStudioConf2017Slides
- RStudio IDE shortcuts learned at rstudio::conf