#PLOTCON

Scott Berinato, Editor HBR

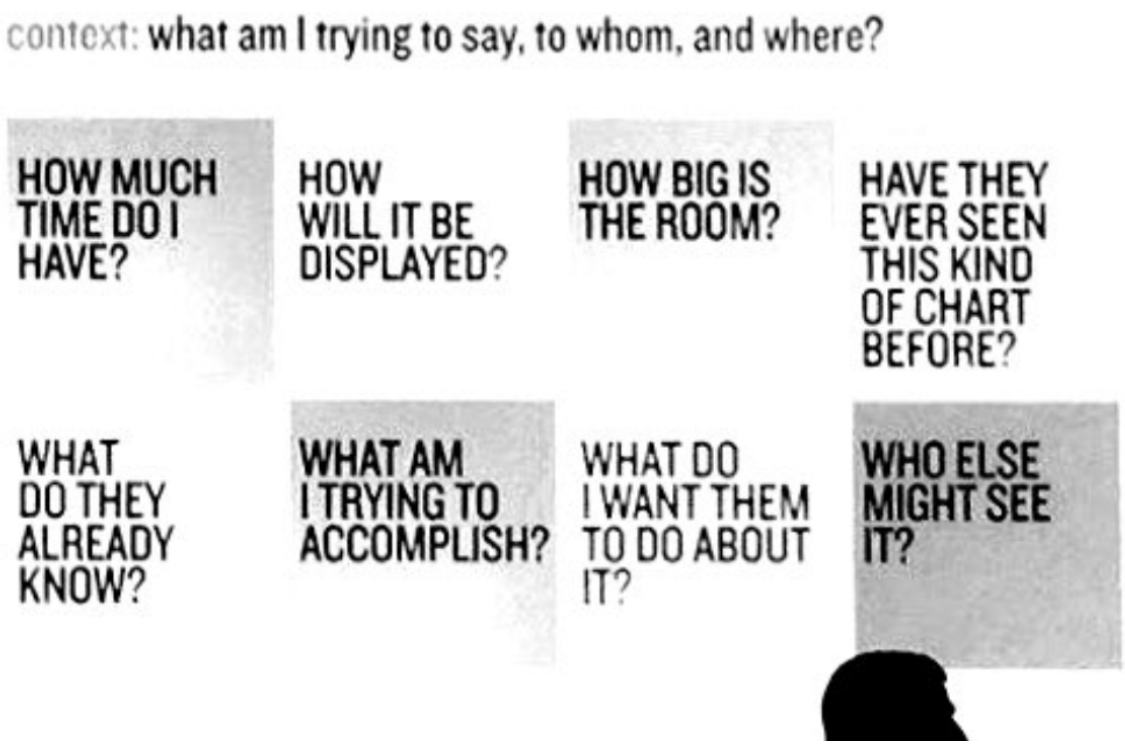
Text walls are dumb AF, don't make people read statistics.

Go read *Good Charts* (https://www.amazon.com/Good-Charts-Smarter-Persuasive-Visualizations/dp/1633690709) published HBR

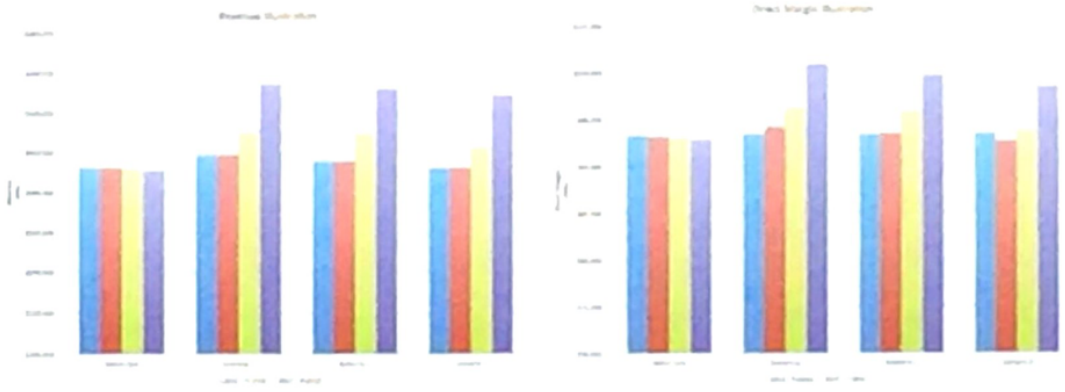
- Can != Should
- Easy != Good

What is Good? Ask: is this actually the right kind of chart?

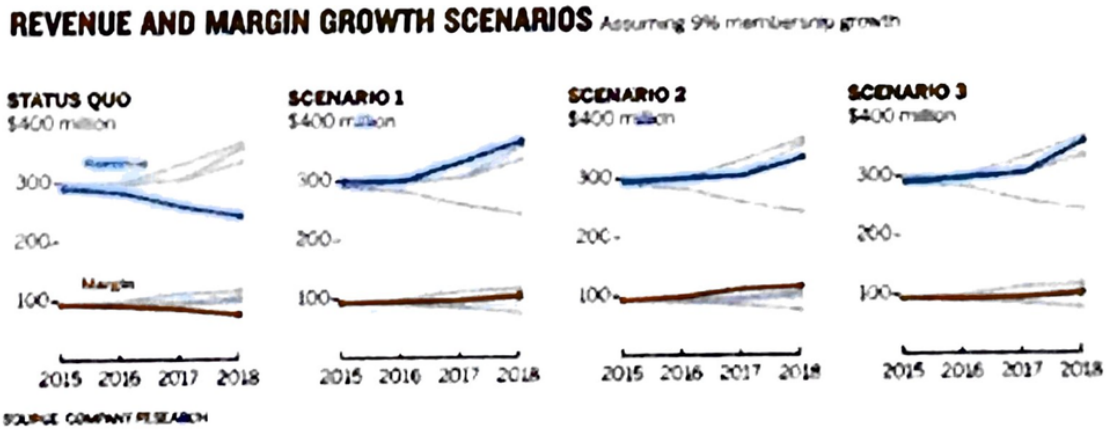
Good Chart Questions:



FOR EXAMPLE



FOR EXAMPLE



Vega-Lite by Dominik Moritz

JSON is going to be THE interchange format and...

Vega/Vega-Lite (https://vega.github.io/vega-lite/) is becoming and likely will be Lingua Franca

Use Altair (https://altair-viz.github.io/) (Vega-lite API hooks) for python (sorta like ggplot2)

Example: Bubble Plot

Showing Fertility, Life Expectancy, and Population in different Regions in 2000

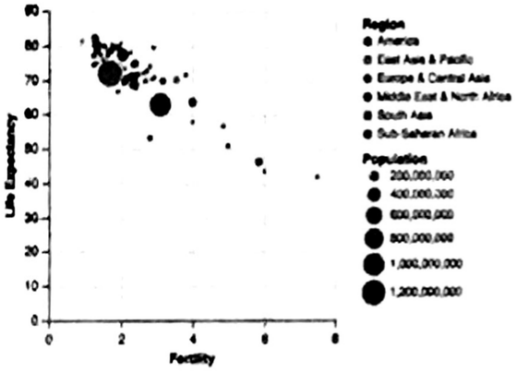
```
{
  data: {url: "data/gapminder.json"},
  transform: {
    filter: {field: "Year", equal: 2000}
  },
  mark: "circle",
  encoding: {
    x:     { field: "Fertility",
             type: "quantitative" },
    y:     { field: "Life Expectancy",
             type: "quantitative" },
    color:{ field: "Region",
             type: "nominal" },
    size:  { field: "Population",
             type: "quantitative" }
  }
}
```

Data

Transform

Mark

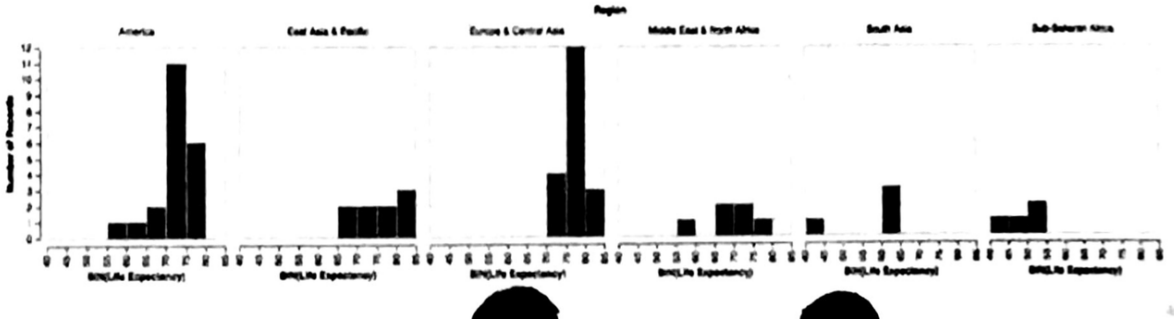
Encoding

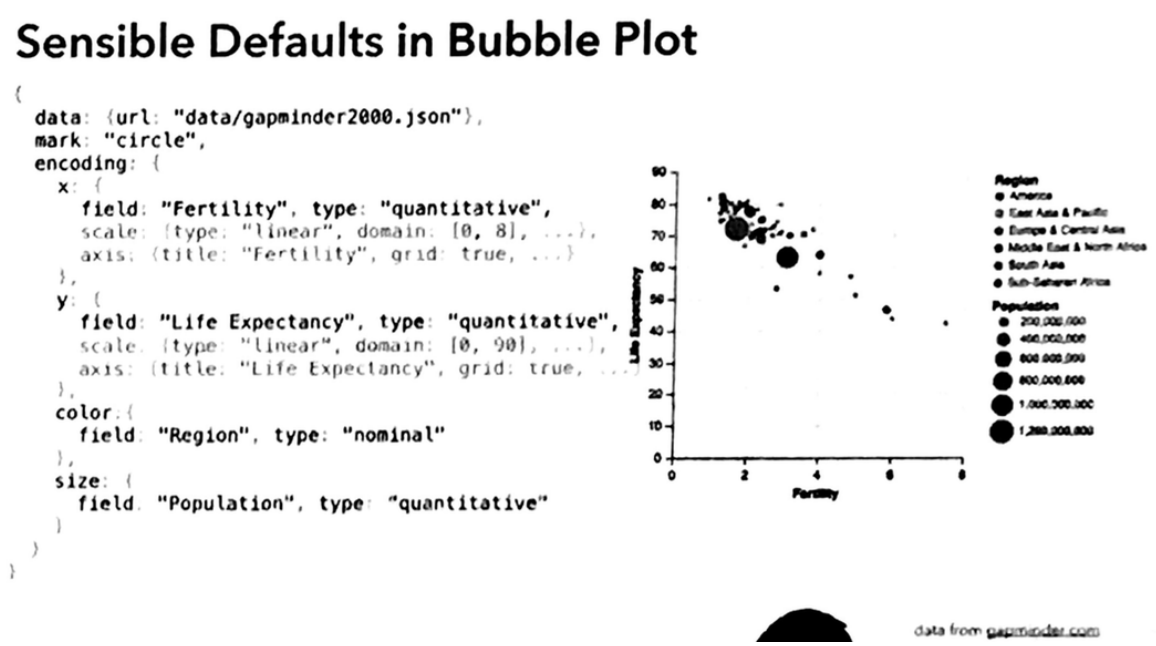


data from gapminder.com

Write-once, reapply with different data.

```
{
  data: {url: "data/gapminder2000.json"},
  mark: "bar",
  encoding: {
    x: {bin: {maxbins: 10}, field: "Life Expectancy", type: "Q"},
    y: {aggregate: "count", field: "*",               type: "Q"},
    column: {               field: "Region",          type: "N"}
  }
}
```





Dealing with Text by Irene Ros

THIS PRESENTATION (https://speakerdeck.com/iros/text-analysis-and-visualization) IS MONEY FOR TEXT ANALYSIS.

Check out NLTK 3.0 (for python) (http://www.nltk.org/)

> Don't forget to stem when dealing with text

`NLTK.pos_tag()` for part of speech tagging (n, adj, verb, ect)

Interesting measures:

- Concordance for keyword context
- n-grams for correlations (see: Google bi-grams)
- Co-occurance for things like X [word] Y
- Significance, use something like TF IDF

Relevant Slides:

CONCORDANCE

Quick Concordance Plots

A Concordance plot shows you where in a set of texts a particular search word appears. Use this quick tool to make your own concordance plots! Drag in your own texts, or try it out now with the included *Alice in Wonderland*. Punctuation is automatically removed and all words are lowercased.

Search Word: King

Search Word: Alice

lory and an eagle! and several other curious creatures alice led th

Search Word: caterpillar

you knew i don t know said the caterpillar alice said nothing she h

N-GRAMS (COLLOCATIONS)

Visualizing Google's Bi-Gram Data

N-GRAMS (COLLOCATIONS)

A set of words that occur together more often then chance.

```
from nltk.collocations import BigramCollocationFinder
finder = BigramCollocationFinder.from_words(filtered_tokens)

# built in bigram metrics are in here
bigram_measures = nltk.collocations.BigramAssocMeasures()

# we call score_ngrams on the finder to produce a sorted list
# of bigrams. Each comes with its score from the metric, which
# is how they are sorted.
finder.score_ngrams(bigram_measures.raw_freq)
```

```
[(('said', 'the'), 0.007674512539933169),
 (('of', 'the'), 0.004700179928762898),
 (('said', 'alice'), 0.004259538060441376),
 (('in', 'a'), 0.0035618551022656335),
 (('and', 'the'), 0.002900892299783351),...]
```

CO-OCCURENCE

```
my_text = nltk.Text(tokens)
my_text.findall('<.*><of><.*>')
```

```
tired of sitting; and of having; use of a; pleasure of making; trouble
of getting; out of the; out of it; plenty of time; sides of the; ans
of the; fear of killing; one of the; nothing of tumbling; top of the;
centre of the; things of this; name of the; saucer of milk; sort of
way; heap of sticks; row of lamps; made of solid; one of the; doors of
the; any of them; out of that; beds of bright; be of very; book of
rules; neck of the; sort of mixed; flavour of cherry-tart; flame of a;
one of the; legs of the; game of croquet; fond of pretending; enough
of me; top of her; way of expecting; Pool of Tears; out of sight; pair
of boots; roof of the; ashamed of yourself; gallons of tears;
pattering of feet; pair of white; help of any; were of the; any of
them; sorts of things; capital of Paris; capital of Rome; waters of
the; burst of tears; tired of being; one of the; cause of this; number
of bathing; row of lodging; pool of tears; be of any; out of this;
tired of swimming; way of speaking; -- of a; one of its; knowledge of
history; out of the; end of his; subject of conversation; -- of --;
```

The occurrence of "cat" in an article in **New York Times**

↓

Significant

The occurrence of "cat" in an article in **Cat Weekly Magazine**

↓

Not Significant

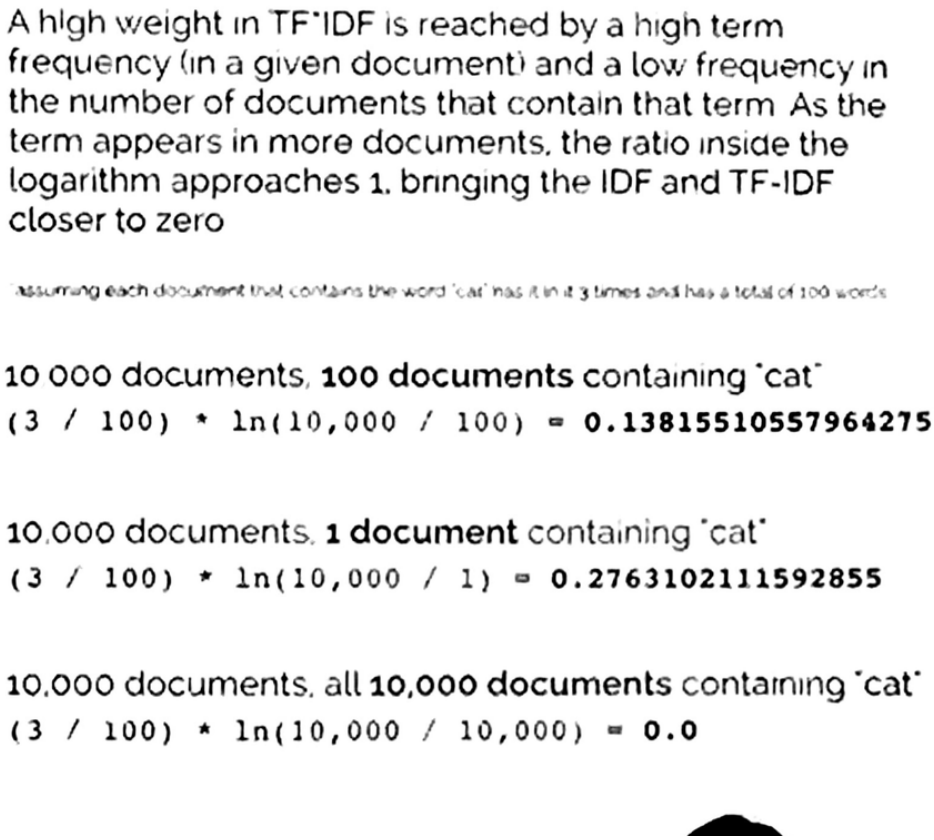
TF

Term Frequency (TF) =

$$\frac{\text{Number of times a term appears in a document}}{\text{Total number of terms in a document}}$$

1 document
100 words
3 "cat"

3 / 100 = 0.03



Mattew Conlen of fivethirtyeight

What went wrong, and other musings

If there's a 30% chance of rain... and it rains... was I wrong?

"Uncertainty narrows"

How do we communicate uncertainty better? Because people still don't get it.

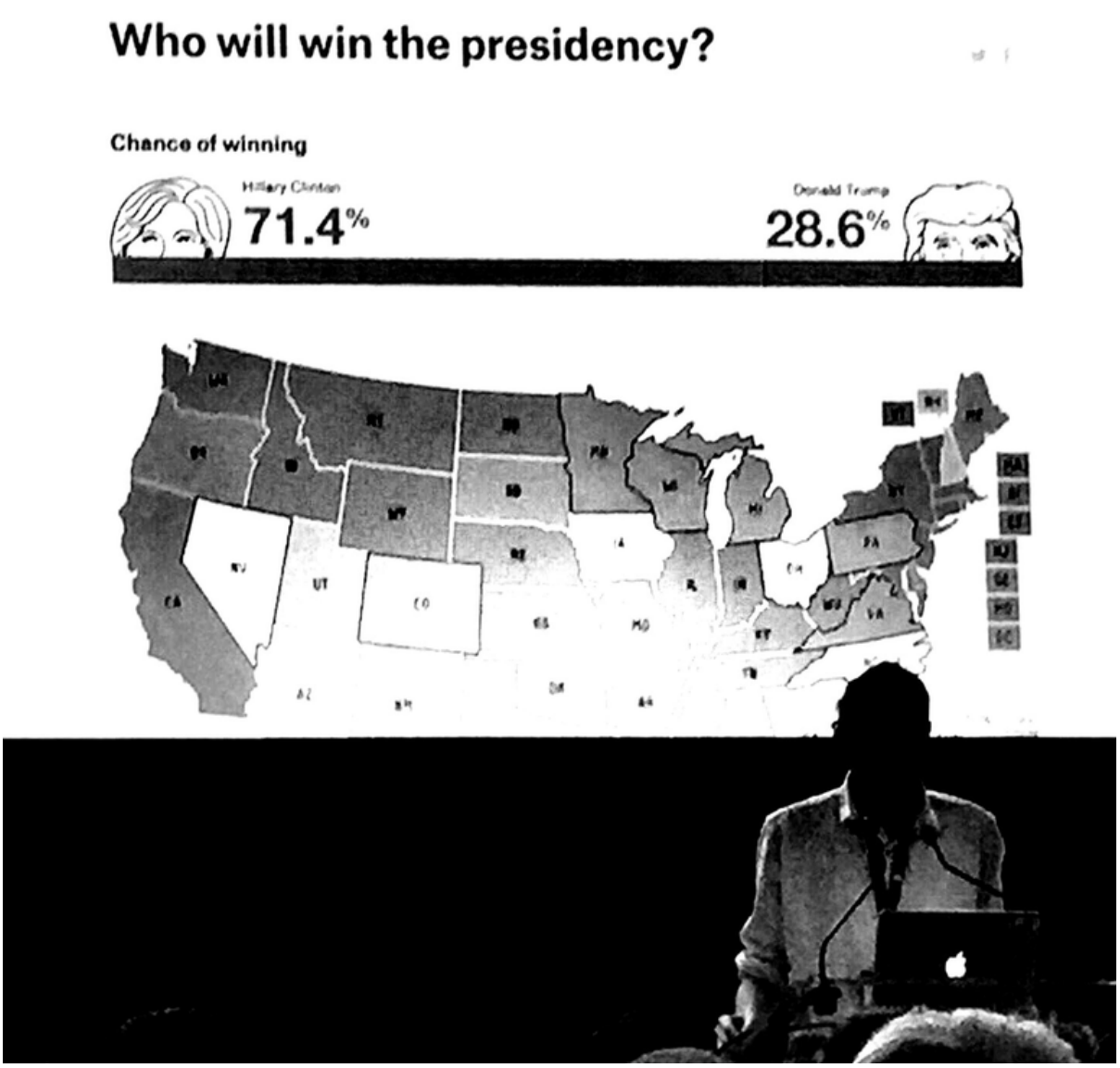
Possible answers?

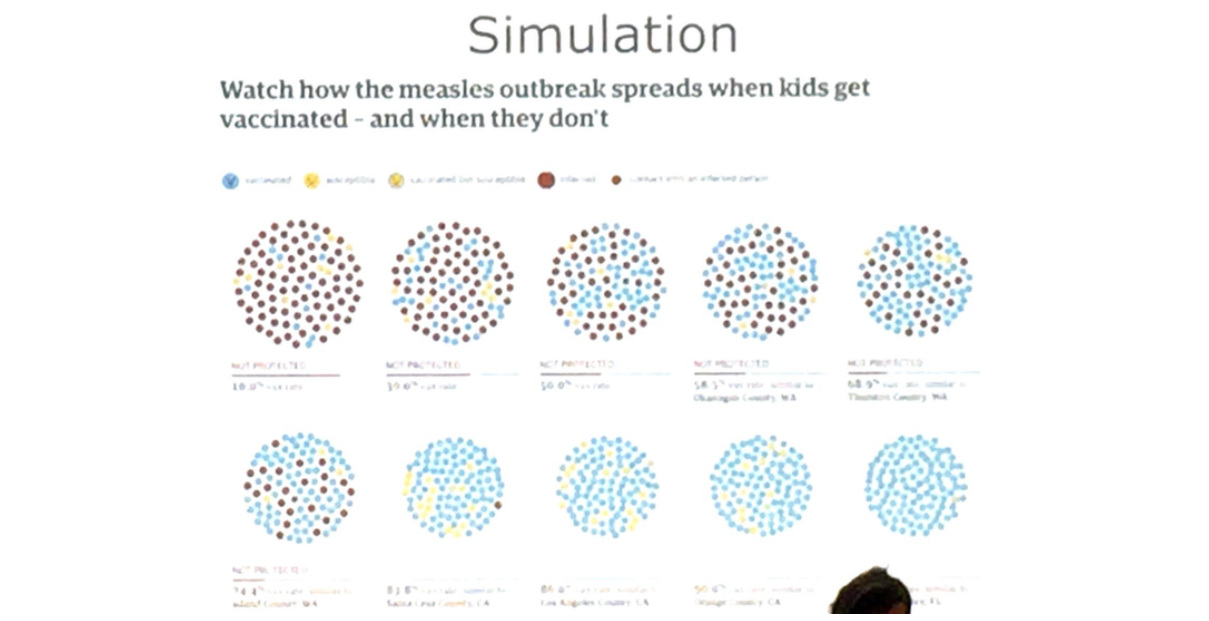
- Hypothetical Outcome Plots
- Simulations! Patterns are better than equations

Think about boutique data sources

Check out:

- BINDER for live hosting Jupyter Notebooks mybinder.org (mybinder.org)
- Terra Pattern for GEO SEARCH
- IDL at U of Wash


Who will win the presidency?
Chance of winning
Hillary Clinton
71.4%
Donald Trump
28.6%


Simulation
Watch how the measles outbreak spreads when kids get vaccinated - and when they don't

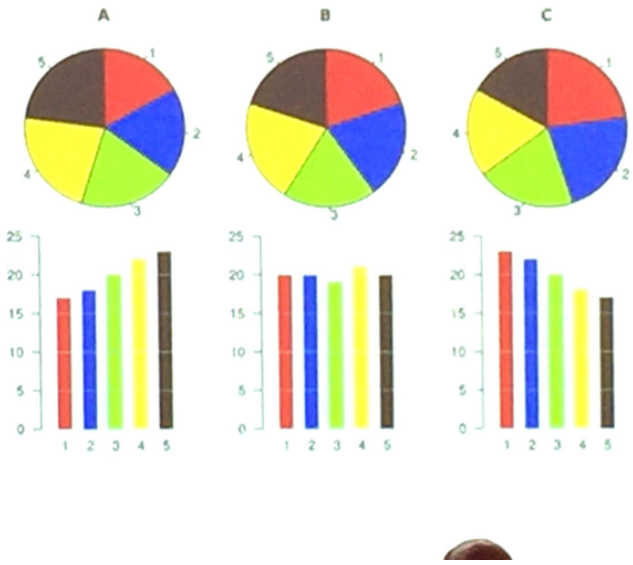
Everything Else from Day I

- "Word clouds are basically pie charts" #MikeWilliams
- "Word clouds are lossy compression" #MW
- "You should know a word by the company it keeps" #MW
- "Git push is going to be the future of publishing" #AndrewOdewahn
- "Jupyter is good for narrated computation" #AO

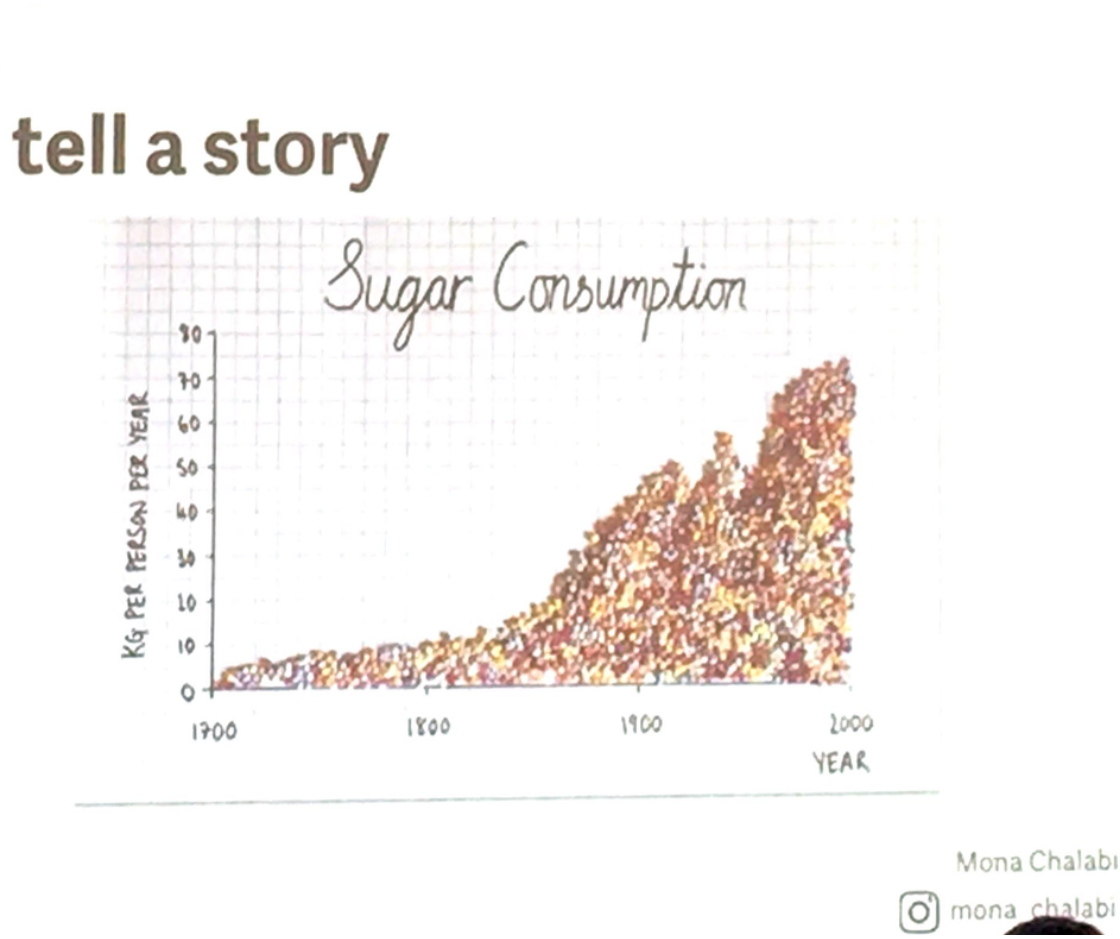
Check out:

- GitLab (better than GitHub for many projects, small teams)
- Project Oriole (in Safari Books)

Pie charts still suck



Analogue charting



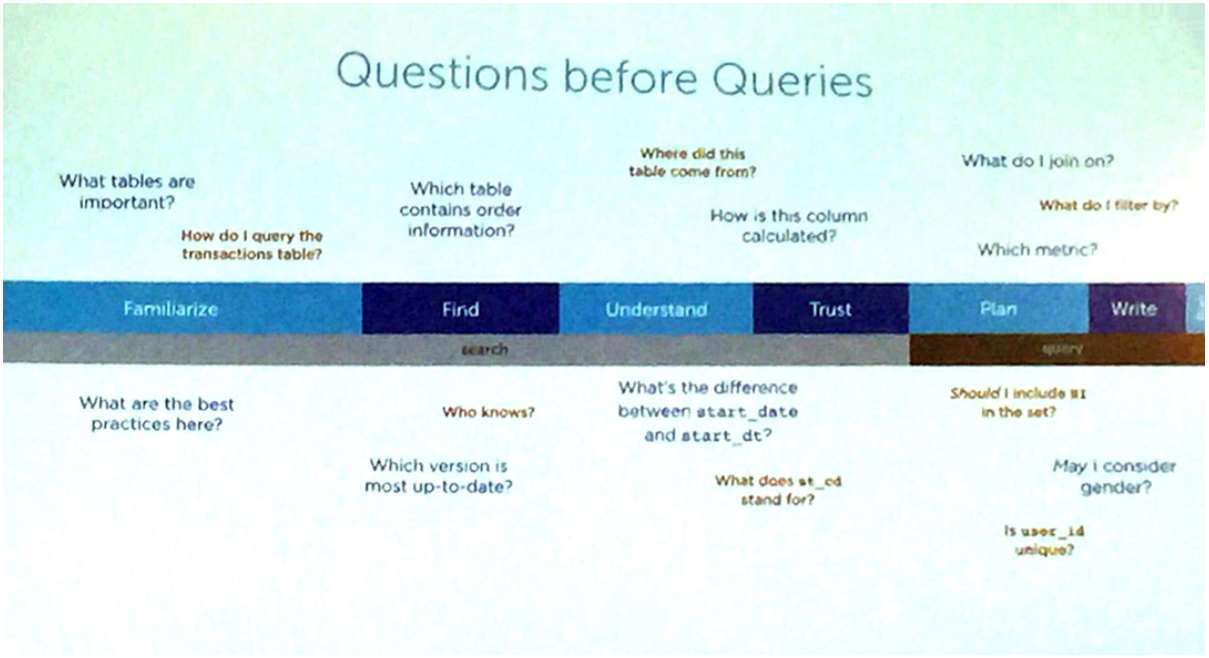
David Crawford of [Alation (https://alation.com/)](https://alation.com/)

"We have questions before queries..."

Reverse-engineer the data. What's important? What's popular?

Look at most frequent joins

Machine Guess, Human Confirm is going to be a big paradigm shift



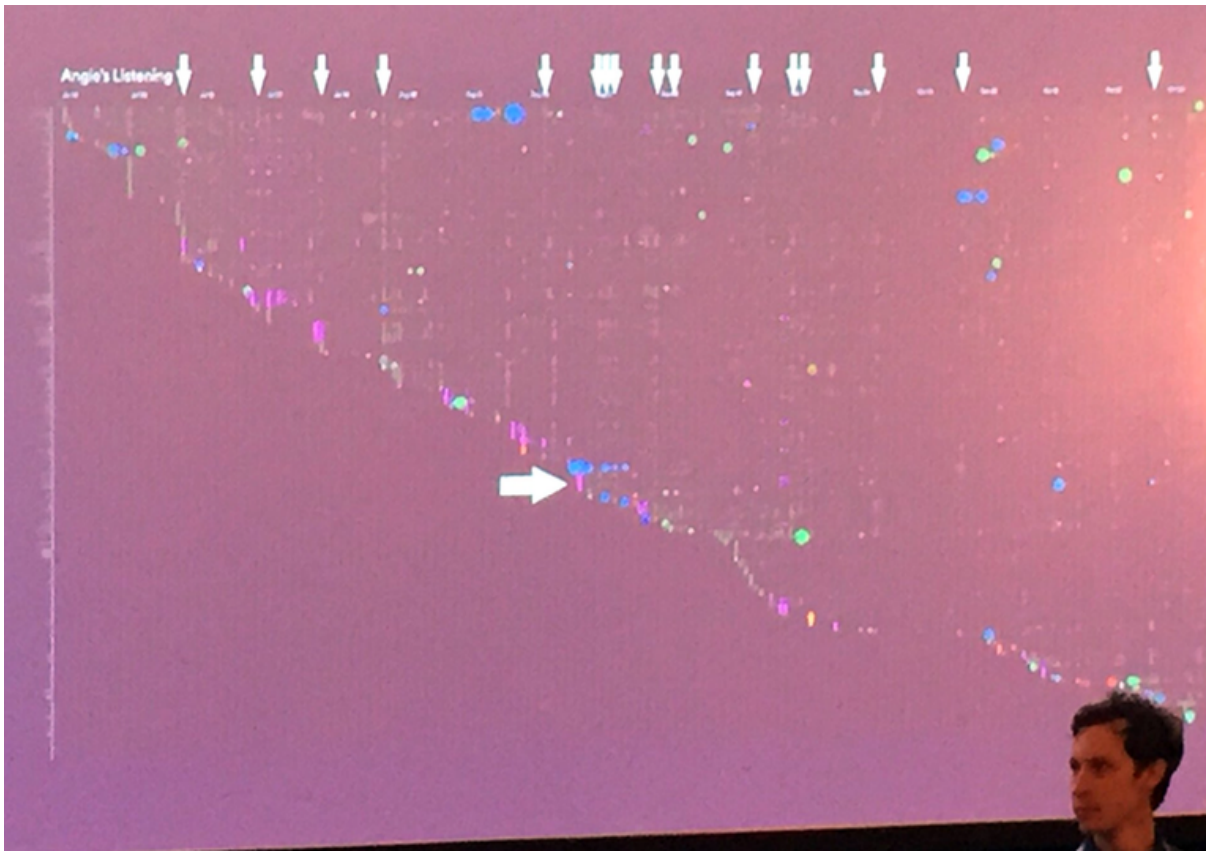
Edward Lee of Spotify (LinkedIn)

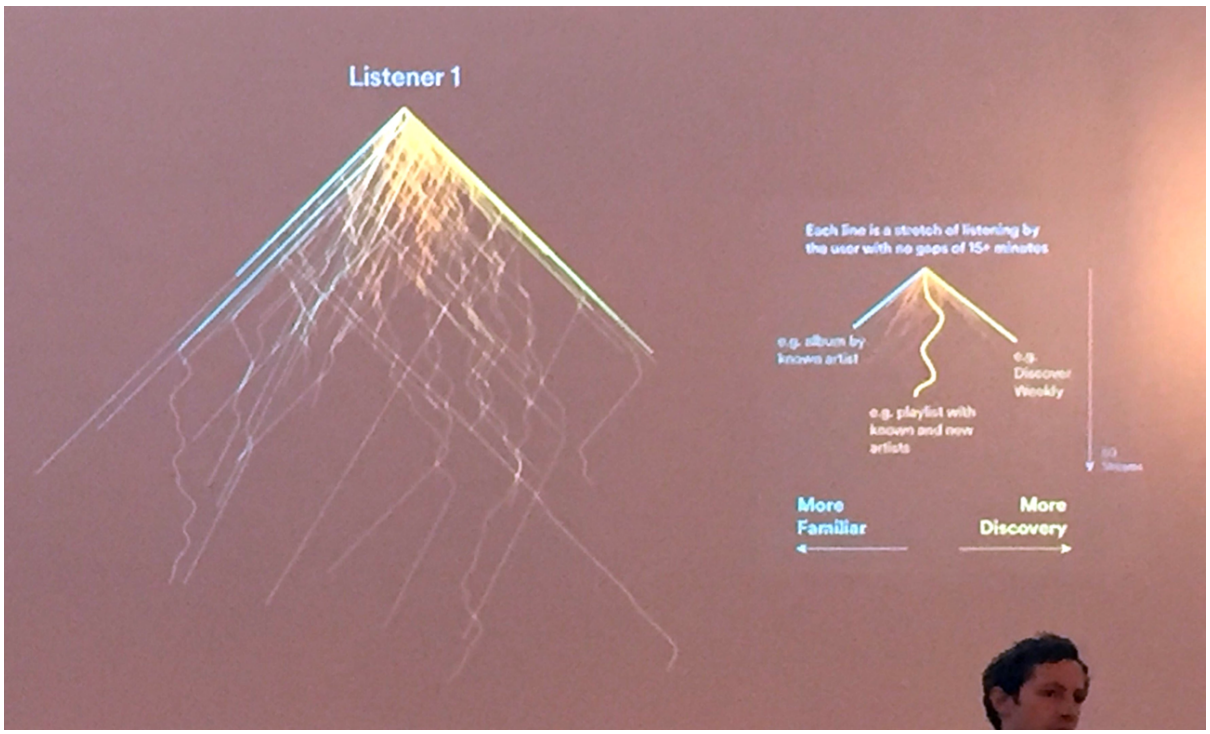
"Let's just see the data". We force our ideas on data... to coerce it into what we want it to show. Try to be more objective at beginning. Find ways to put up super-multi-dimensional data

Spot checking at human level helps uncover outliers, problems

Explore vs. Exploit trade-offs

> Can we do a session metric for intent?


Angie's Listening


Listener 1
Each line is a stretch of listening by the user with no gaps of 15+ minutes
e.g. album by known artist
e.g. playlist with known and new artists
e.g. Discover Weekly
More Familiar
More Discovery

Everything Else from Day II

- "Training peers leads to geometric growth" - Stacey Jones
- "Vis is good for fitting data into Human RAM" - Scott Sanderson
- Animation is good when it's used for building up complexity
- If visualization is complicated, it's okay to use primers
- "Innovation is just a buzz word for solving problems" - Yao Huang

Check out:

- `.table()`, `.chart()`, `.interact()` in Jupyter
- Quantopian (https://www.quantopian.com/home)

Dask (http://dask.pydata.org/en/latest/) by Matthew Rocklin

Dask is the best distributed computing library for python

Look at `%%time magic` in Jupyter

Bokeh is a relaly good stand in for D3.js (interactivity)

Jupyter by Fernando Perez

YOU CAN NOW MIX AND MATCH PYTHON AND R CODE WITH RPY2 (http://rpy2.bitbucket.org/)

Jupyter Lab (https://github.com/jupyterlab/jupyterlab) can split out cells (but still super Alpha)

Get C++ and bash kernels for Jupyter configured

(Follow tutorials for setting .profile & .bash_profile)

Links:

- Yhat R & Python (http://blog.yhat.com/posts/rpy2-combing-the-power-of-r-and-python.html)
- R magic deprecated (https://ipython.org/ipython-doc/2/config/extensions/rmagic.html)
- R/Py Example Notebook (https://github.com/lgautier/jpd-pdapr-slides/blob/gh-pages/notebooks/potholes.ipynb)

Datashader (https://datashader.readthedocs.io/en/latest/) (Bokeh) by Peter Wang

Massive data amplifies the normal issues with data vis

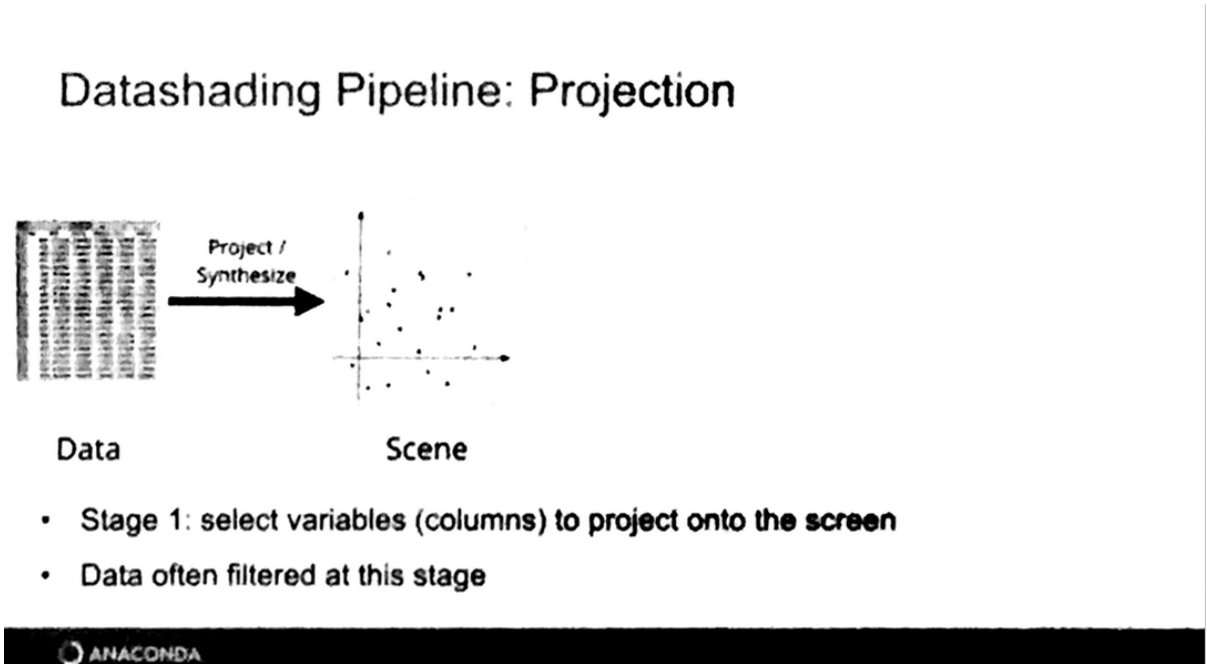
Alpha values are the end all... still have problems. Still hiding outliers, still messing up with collisions, still over/undersaturating

Binning data reduces information density.

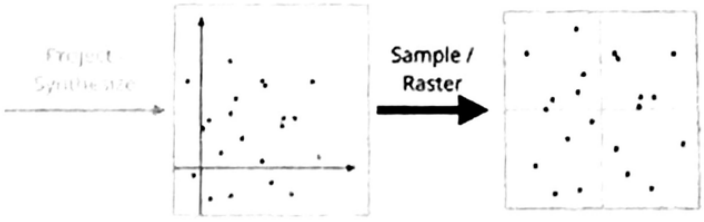
Always be skeptical of binning... it's lying to you!

Datashader [Link] has a live pipeline that lets you hit with statistics

Use with Dask and Numba, (and regular Numpy/pandas)



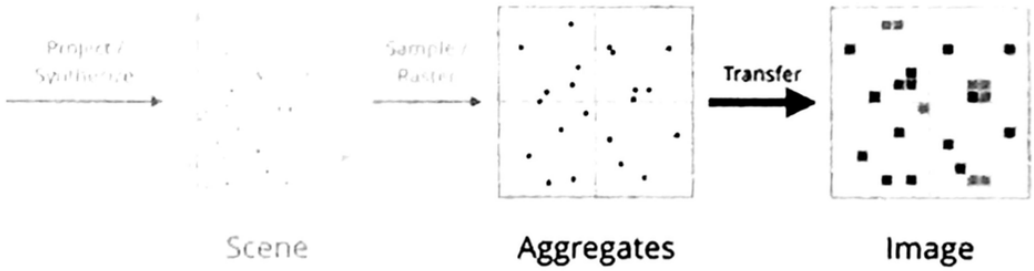
Datashading Pipeline: Aggregation



- Stage 2: Aggregate data into a fixed set of bins
- Each bin yields one or more scalars (total count, mean, stddev, etc.)

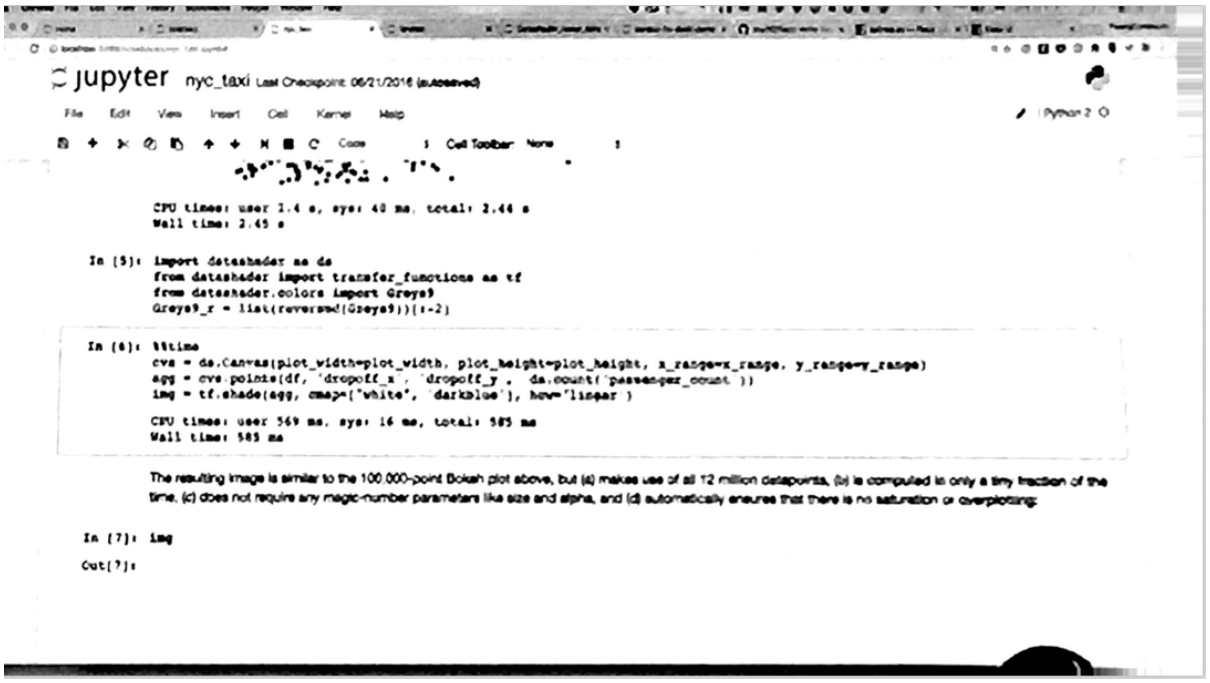
ANACONDA

Datashading Pipeline: Transfer

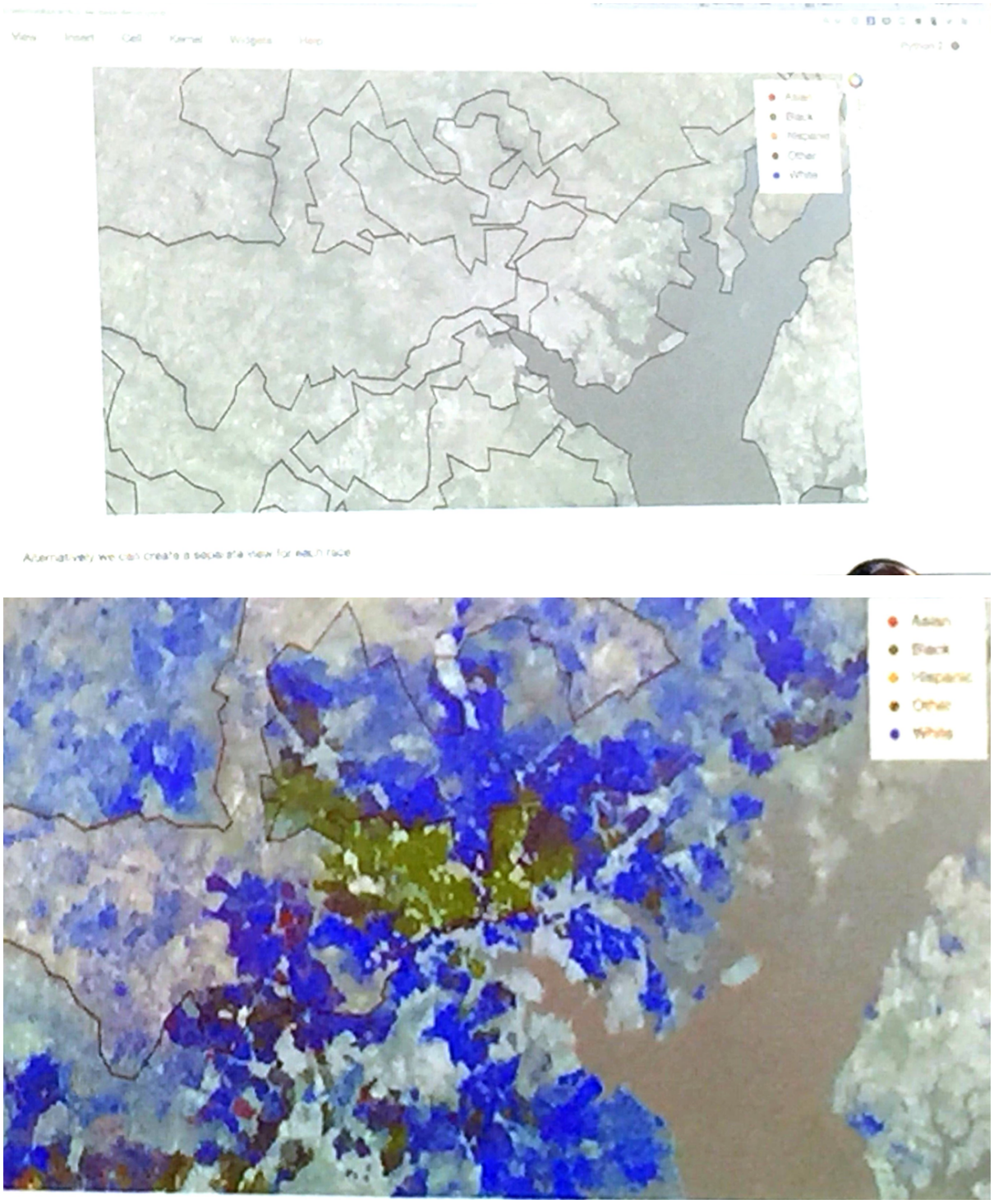


- Stage 3: Transform data using one or more transfer functions, culminating in a function that yields a visible image
- Each stage can be replaced and configured separately

ANACONDA


jupyter nyc_taxi Last Checkpoint: 06/21/2016 (autosaved)
File Edit View Insert Cell Kernel Help
Python 2
CPU times: user 1.4 s, sys: 40 ms, total: 2.44 s
Wall time: 2.45 s
In [5]: import datashader as ds
from datashader import transfer_functions as tf
from datashader.colors import Greys9
Greys9_r = list(reversed(Greys9))[:-2]
In [6]: %%time
cvs = ds.Canvas(plot_width=plot_width, plot_height=plot_height, x_range=x_range, y_range=y_range)
agg = cvs.points(df, 'dropoff_x', 'dropoff_y', ds.count('passenger_count'))
img = tf.shade(agg, cmap=['white', 'darkblue'], how='linear')
CPU times: user 569 ms, sys: 16 ms, total: 585 ms
Wall time: 585 ms
The resulting image is similar to the 100,000-point Bokeh plot above, but (a) makes use of all 12 million datapoints, (b) is computed in only a tiny fraction of the time, (c) does not require any magic-number parameters like size and alpha, and (d) automatically ensures that there is no saturation or overplotting:
In [7]: img
Out[7]:


Edit View Insert Cell Kernel Widgets Help
Python 2
/Users/pwang/anaconda/envs/plotcon/lib/python2.7/site-packages/datashader/transfer_functions.py:263: RuntimeWarning: invalid value encountered in true_divide
b = (data.dot(be)/total).astype(np.uint8)
Out[5]:
dropoff_y
dropoff_x



Kristen Thyng

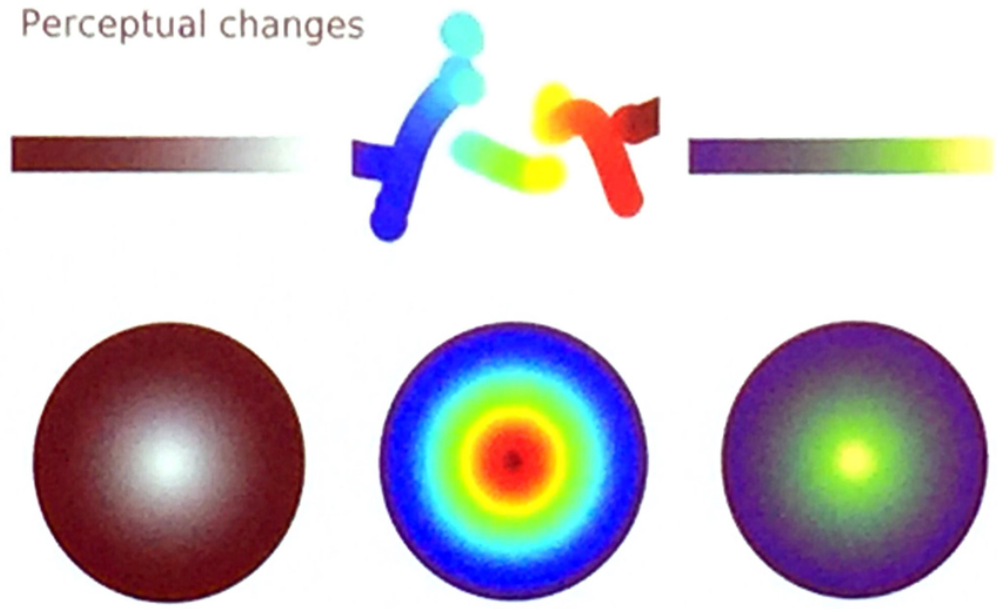
Jet color mapping is garbage.

Use viridis (https://github.com/sjmgarnier/viridis) colour maps.

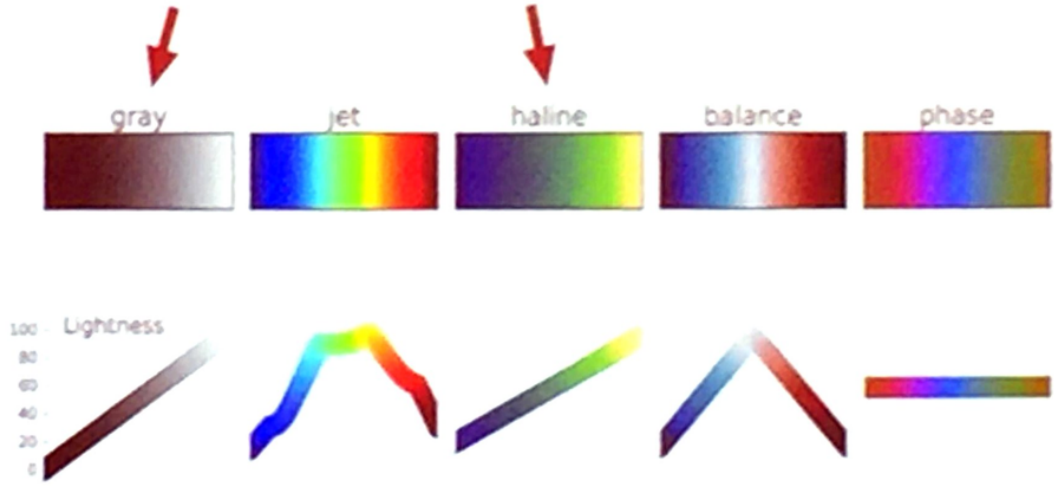
Lightness is what's important in continuous cmaps (not hue).

Use diverging color maps for anomaly detection.

Perceptually uniform colormap doesn't incidentally add or remove information



Importance of lightness



cmocean

beautiful colormaps for oceanography

pip install cmocean

or

conda install -c conda-forge cmocean

matplotlib.org/cmocean

Everything else from Day III

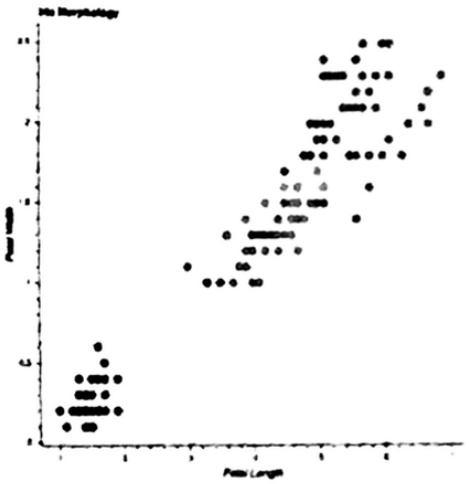
- Foundations of Data Science, data8.org (http://data8.org/fa16/)
- "Before we were data scientists, we were story tellers"
- Follow: nteract.io (https://nteract.io/) project
- Beaker Notebook (http://beakernotebook.com/) might be a viable alternative to Jupyter in future (really interesting autotranslation features... that uses JSON as interchange format)
- Bokeh Server (http://brocktane.net/blog/bokeh-server-example/) is a Shiny for python

Bokeh

bokeh.pydata.org

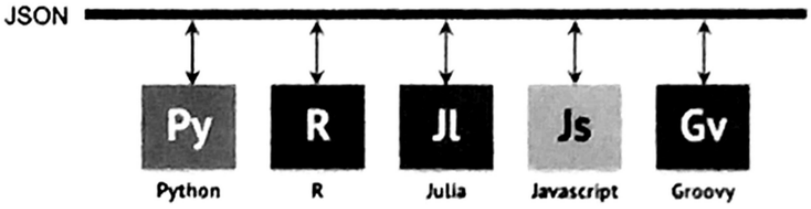
```
# bokeh.plotting

p = figure(title="Iris Morphology")
p.xaxis.axis_label = 'Petal Length'
p.yaxis.axis_label = 'Petal Width'
p.circle(
    x=flowers["petal_length"],
    y=flowers["petal_width"],
    color=flowers["colors"])
show(p)
```



ANACONDA

Autotranslation




```
# set in Python
beaker.x = 10
```

```
// use in Javascript
beaker.x + 1
```

Hadley I

A good package MUST have an opinion

`ggplot2` now has 1:1 translations for secondary y-axes

Use `ggrepel` (https://github.com/slowkow/ggrepel) for labeling data

> "Visualization is the easy bit"

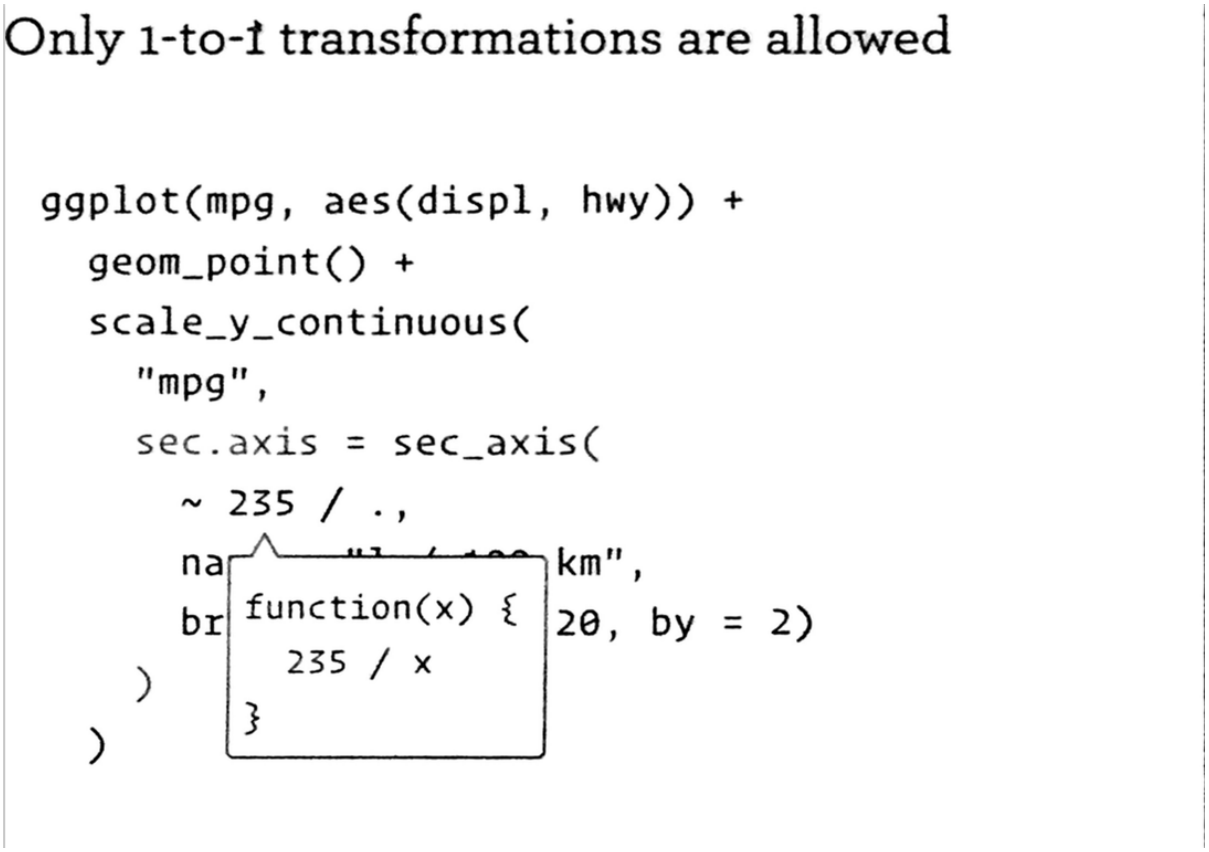
Topic of current consideration: how to visualize missingness?

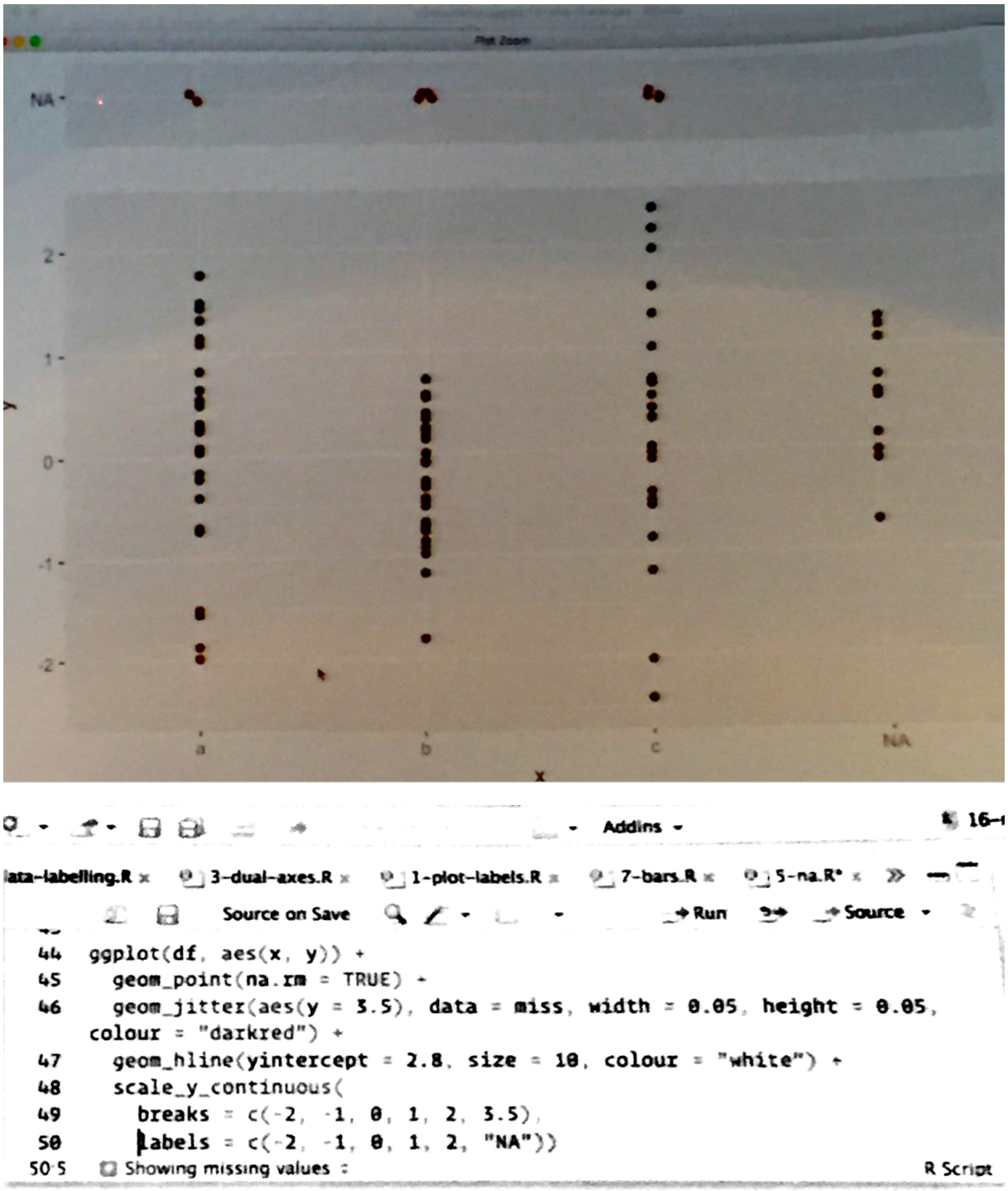
Look into `add_na()`

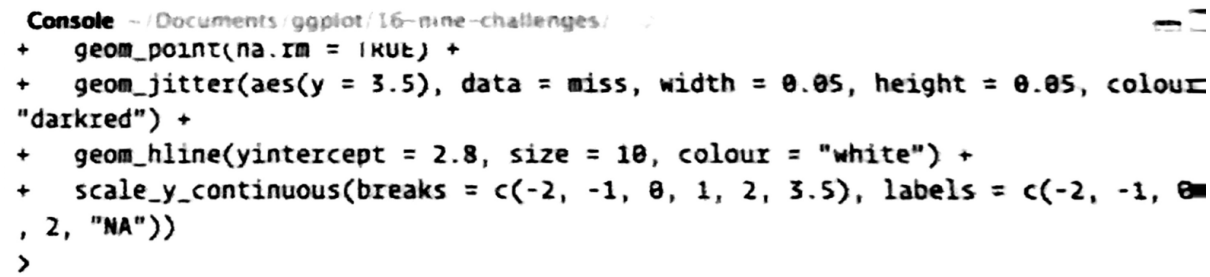
checkout:

- Documentation at: tidyverse.org (http://tidyverse.org/)
- ggplot2-exts.org (http://www.ggplot2-exts.org/)

Get into some pull requests







Data {rect}/{wr}angling by Jenny Bryan

"What happens in a dataframe (should) stay/s in a dataframe"

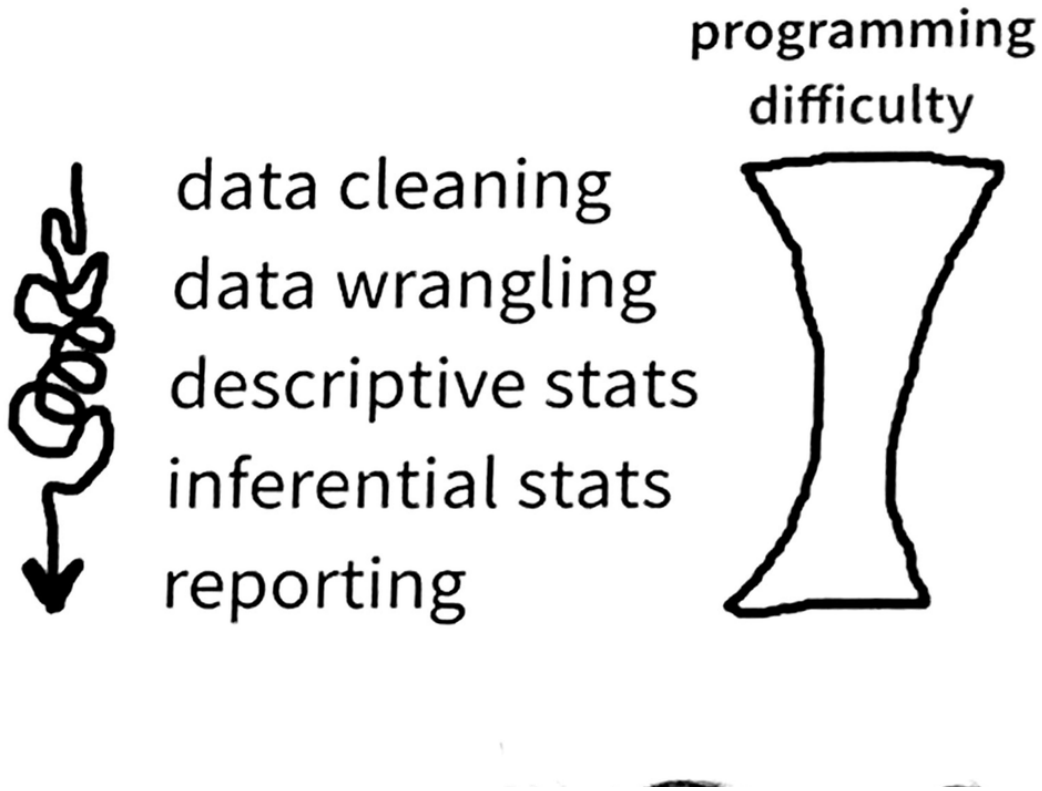
Fuck Matrices

Why should you invite `lists` into your life?

- Working with strings (regex) ... check https://regex101.com/ (https://regex101.com/)
- JSON or XML
- Split-Apply-Combine paradigms

Keep vectors intact and in sync (use list columns in df/tibbles)

`map()` is basically just `lapply()`



Lessons from my fall 2016 teaching:

https://jennybc.github.io/purrr-tutorial/

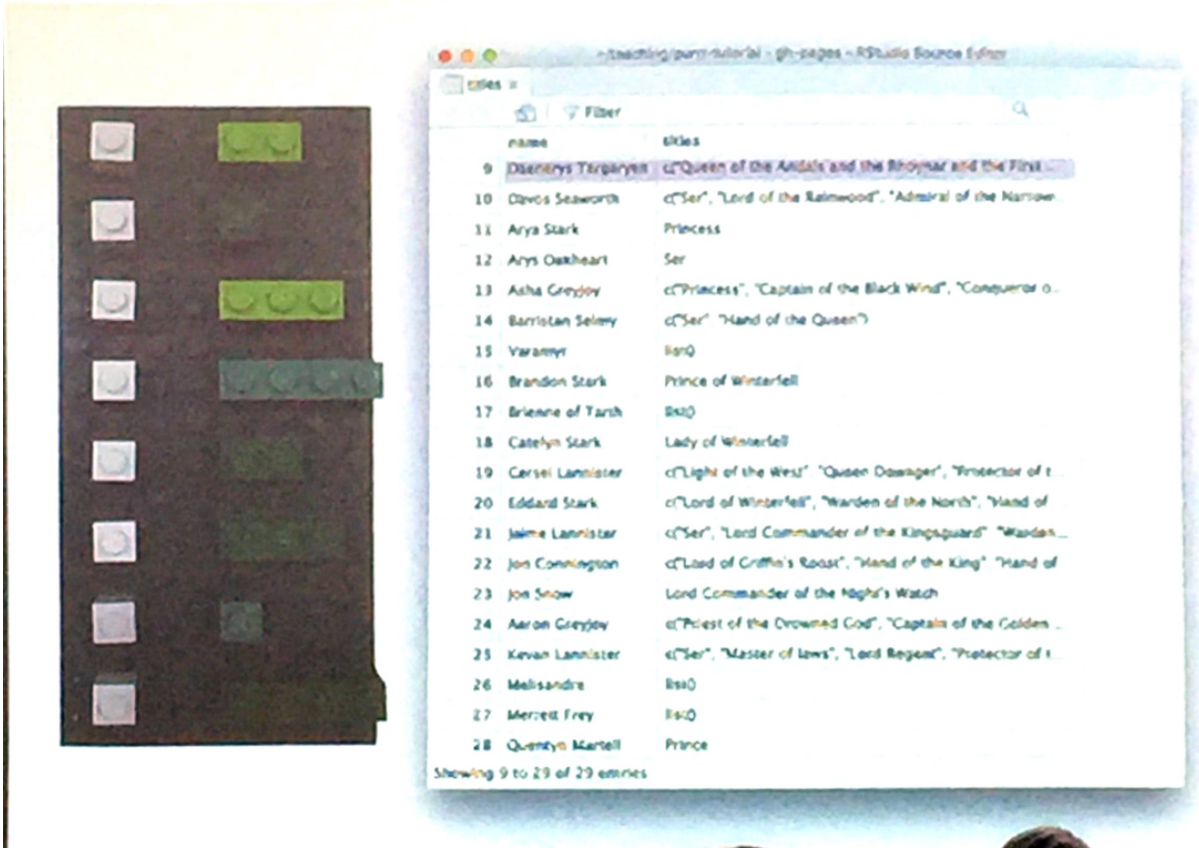
repurrrsive package (non-boring examples):

https://github.com/jennybc/repurrrsive

I am the Annie Leibovitz of lego mini-figures:

https://github.com/jennybc/lego-rstats


```
{
 "url": "http://www.anapioficeandfire.com/api/characters/1303",
 "id": 1303,
 "name": "Daenerys Targaryen",
 "gender": "Female",
 "culture": "Valyrian",
 "born": "In 284 AC, at Dragonstone",
 "died": "",
 "alive": true,
 "titles": [
     "Queen of the Andals and the Rhoynar and the First Men,
      Lord of the Seven Kingdoms",
     "Khaleesi of the Great Grass Sea",
     "Breaker of Shackles/Chains",
     "Queen of Meereen",
     "Princess of Dragonstone"
 ],
 "aliases": [
     "Dany",
     "Daenerys Stormborn"
```



What happens in the

DATA FRAME

Stays in the data frame

```
map_df(minis, `[`,
       c("pants", "torso", "head")
```

simplify

```
> map_df(got_chars, `[`,
         c("name", "culture", "gender", "born"))
#> # A tibble: 29 × 4
#>                    name  culture gender                born
#>                   <chr>    <chr>  <chr>               <chr>
#>  1       Theon Greyjoy Ironborn   Male In 278 AC or 279 AC, at Pyke
#>  2    Tyrion Lannister            Male  In 273 AC, at Casterly Rock
#>  3  Victarion Greyjoy Ironborn    Male In 268 AC or before, at Pyke
#>  4                 Will           Male
#>  5          Areo Hotah Norvoshi   Male In 257 AC or before, at Norvos
#>  6                Chett           Male                At Hag's Mire
#>  7              Cressen           Male In 219 AC or 220 AC, at Norvos
#>  8    Arianne Martell  Dornish Female In 276 AC, at Sunspear
#>  9 Daenerys Targaryen Valyrian Female In 284 AC, at Dragonstone
#> 10      Davos Seaworth Westeros   Male In 2[illegible] King's Landing
#> # ... with 19 more rows
```

simplify

```
got_chars %>% {
  tibble(name = map_chr(., "name"),
         houses = map(., "allegiances"))
} %>%
  filter(lengths(houses) > 1) %>%
  unnest()
```

```
#> # A tibble: 15 × 2
#>                 name
#>               <chr>
#> 1    Davos Seaworth
#> 2    Davos Seaworth
#> 3      Asha Greyjoy
#> 4      Asha Greyjoy
```

```
                houses
                 <chr>
House Baratheon of Dragonstone
House                Cape Wrath
                        f Pyke
                           ker
```

Hadley II

Functional programming is all about:

- Less code
- More ease
- Human readability

But it's basically just for loops (or what you could do with fors)

Nested dataframes (one per row, one per group)

Nest > do work > unnest (explode)

Why would you ever want a "specialized" data strucutre. Get everything into a tibble because you can use all the familiar tools you know and love

Checkout:

- `library(broom); library(modelr)`

Managing many models

November 2016

Hadley Wickham
@hadleywickham
Chief Scientist, RStudio

More convenient to one row per group

Country	Data
Afghanistan	<df>
Albania	<df>
Algeria	<df>
...	...

Year	LifeExp
1952	28.9
1957	30.3
...	...

Year	LifeExp
1952	55.2
1957	59.3
...	...

I call this a **nested data frame**

What data can we extract from a model?

New Zealand

year	lifeEx
1952	69.4
1957	70.3
1962	71.2
1967	71.5
...	...

```
lm(lifeExp ~ year, data = nz)
```

R^2=0.95
glance

tidy

Intercept	-307.7
Slope	0.19

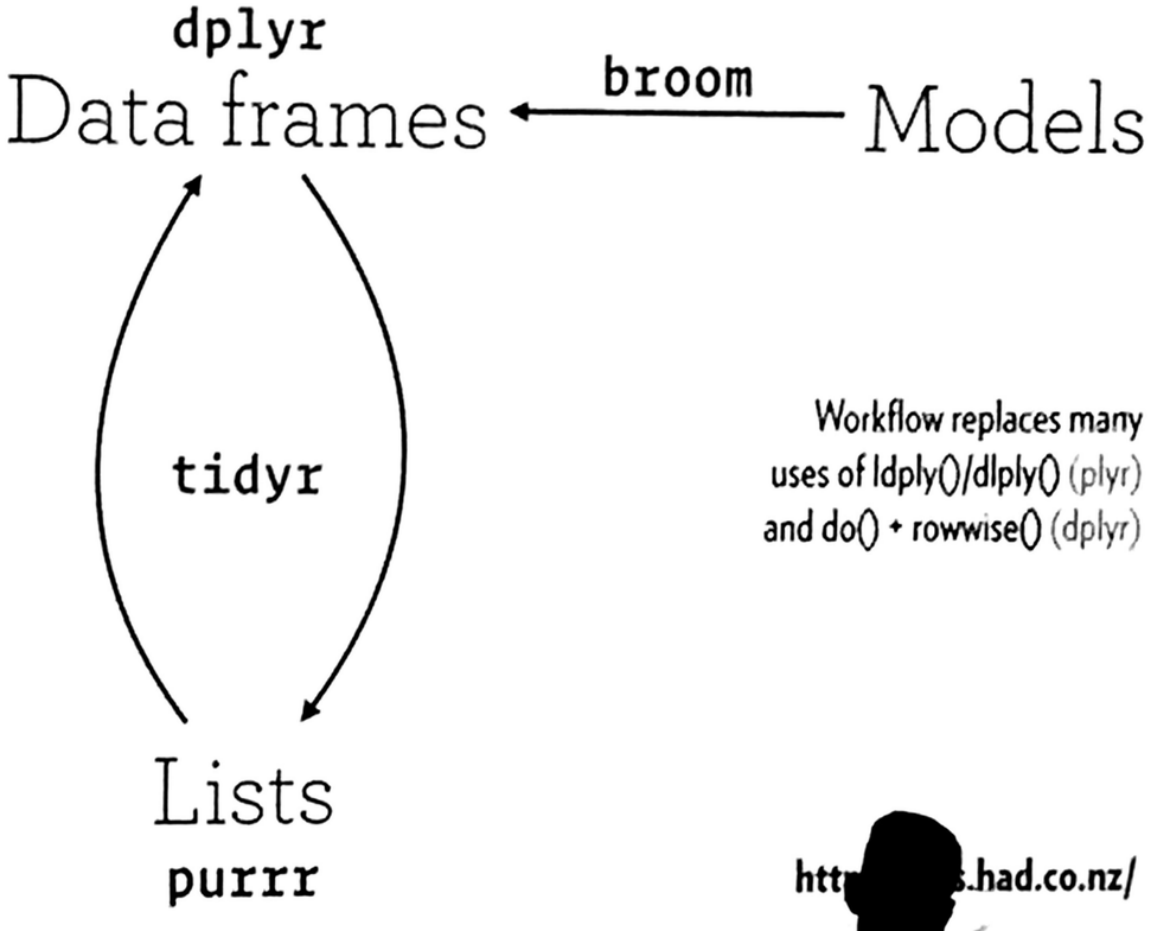
augment

year	resid
1952	0.70
1957	0.61
1962	0.63
1967	-0.05
...	...

```
32
33 - # Broom ------------------------------------------------------------
     --------
34
35   models <- models %>%
36     mutate(
37       glance  = model %>% map(broom::glance),
38       rsq     = glance %>% map_dbl("r.squared"),
39       tidy    = model %>% map(broom::tidy),
40       augment = model %>% map(broom::augment)
38:45  Broom                                                    R Script
```

```
Console ~/Documents/modelling/16-many-models/
3      Africa      Algeria <tibble [12 × 5]> <S3: lm>
4      Africa       Angola <tibble [12 × 5]> <S3: lm>
5    Americas    Argentina <tibble [12 × 5]> <S3: lm>
6     Oceania    Australia <tibble [12 × 5]> <S3: lm>
7      Europe      Austria <tibble [12 × 5]> <S3: lm>
8        Asia      Bahrain <tibble [12 × 5]> <S3: lm>
9        Asia   Bangladesh <tibble [12 × 5]> <S3: lm>
10     Europe      Belgium <tibble [12 × 5]> <S3: lm>
# ... with 132 more rows, and 4 more variables: glance <list>,
#   rsq <dbl>, tidy <list>, augment <list>
```

1. Store related objects in **list-columns**.
2. Learn **FP** so you can focus on verbs, not objects.
3. Use **broom** to convert models to tidy data.



Eduardo Ariño de la Rubia

From Domino Labs

Spine charts are good for duality

Slope charts are good for ranking A and B in time

"Data science is just like 85% histograms"

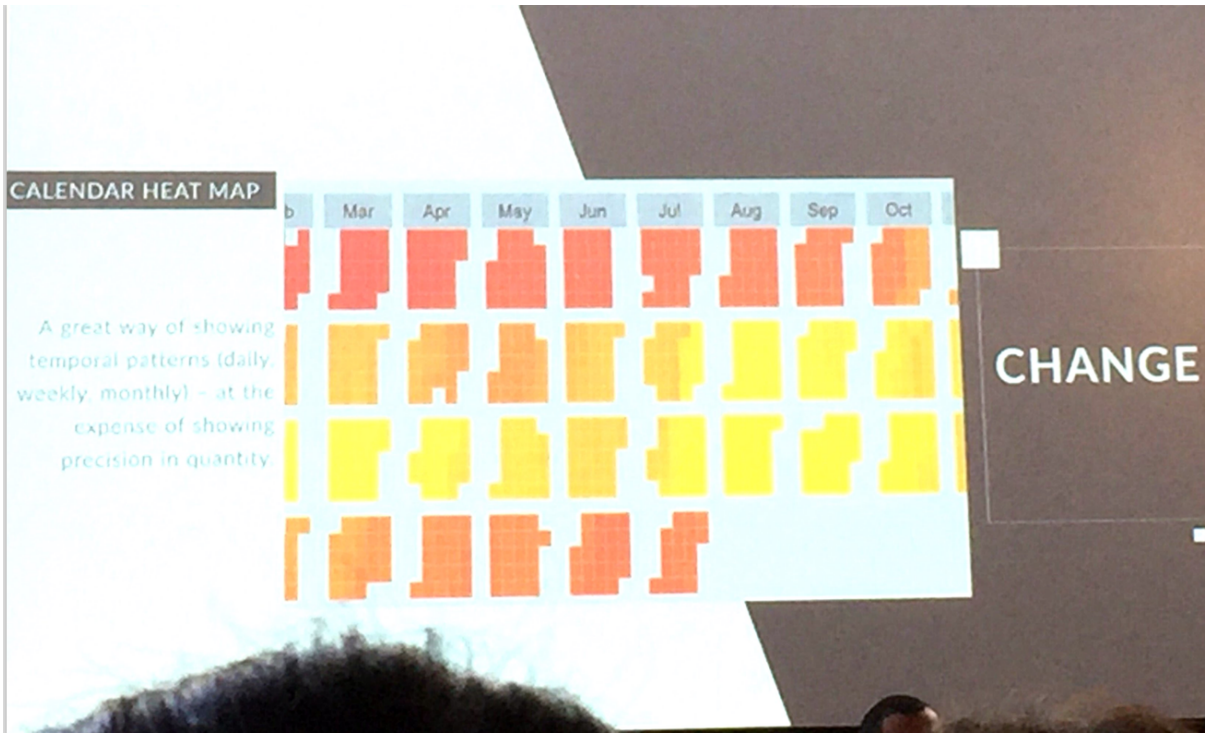
Figure out how to ductape vis together... create the next population pyradmid (basically just two histograms)

- Think about overlayed bars
- Think about heat contours
- Think about animating hypothetical outcome plots

Fan charts... are good for plausible deniability

Check out: `tilegramsR` for 538 hex geos

Don't just pick the "prettiest" chart



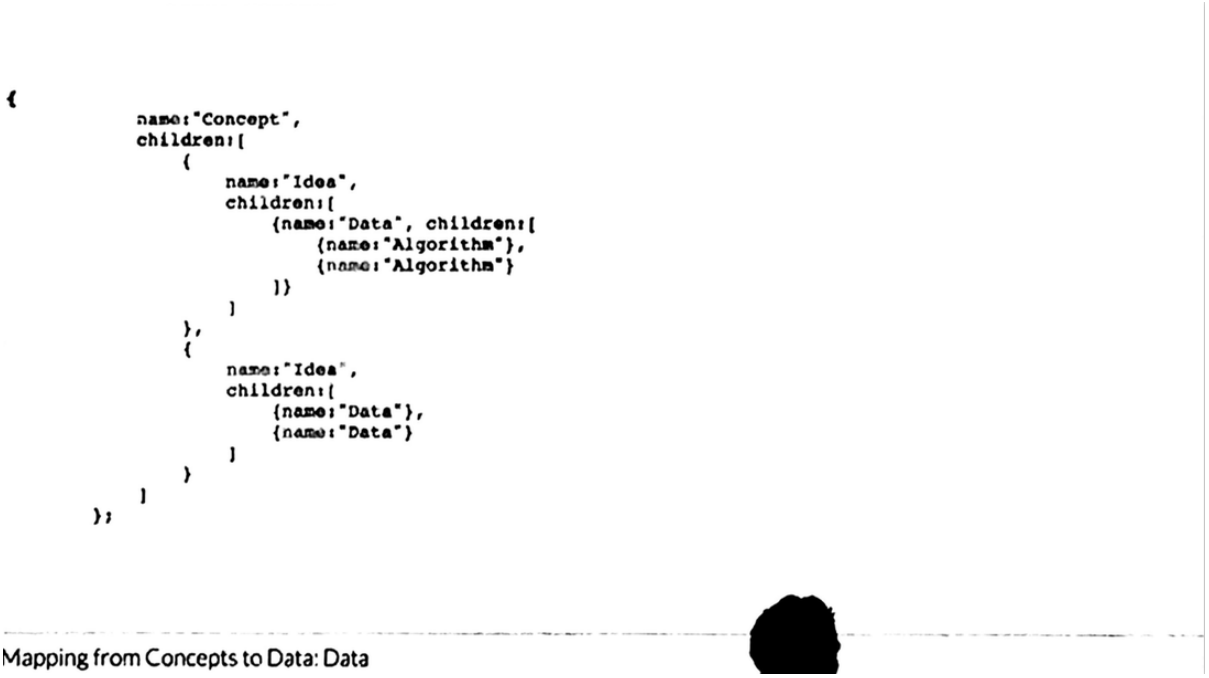
Michael Freeman

U of Washington (IDL ?)

(check out his github/twitter for tuts)

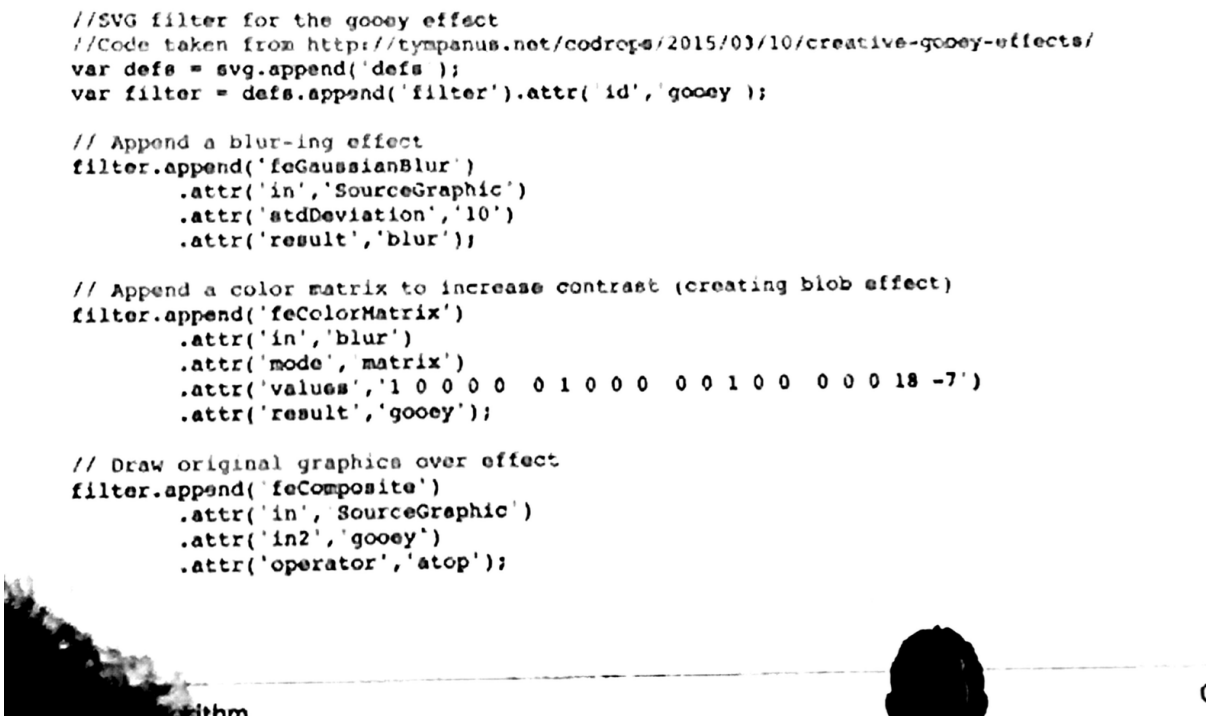
How do we vis a concept? (like division) ... D3?

"Magic" is just data and algorithms



```
{
        name:"Concept",
        children:[
            {
                name:"Idea",
                children:[
                    {name:"Data", children:[
                        {name:"Algorithm"},
                        {name:"Algorithm"}
                    ]}
                ]
            },
            {
                name:"Idea",
                children:[
                    {name:"Data"},
                    {name:"Data"}
                ]
            }
        ]
};
```

Mapping from Concepts to Data: Data



```
//SVG filter for the gooey effect
//Code taken from http://tympanus.net/codrops/2015/03/10/creative-gooey-effects/
var defs = svg.append('defs');
var filter = defs.append('filter').attr('id','gooey');

// Append a blur-ing effect
filter.append('feGaussianBlur')
        .attr('in','SourceGraphic')
        .attr('stdDeviation','10')
        .attr('result','blur');

// Append a color matrix to increase contrast (creating blob effect)
filter.append('feColorMatrix')
        .attr('in','blur')
        .attr('mode','matrix')
        .attr('values','1 0 0 0 0  0 1 0 0 0  0 0 1 0 0  0 0 0 18 -7')
        .attr('result','gooey');

// Draw original graphics over effect
filter.append('feComposite')
        .attr('in','SourceGraphic')
        .attr('in2','gooey')
        .attr('operator','atop');
```

David Robinson

`gganimate()`

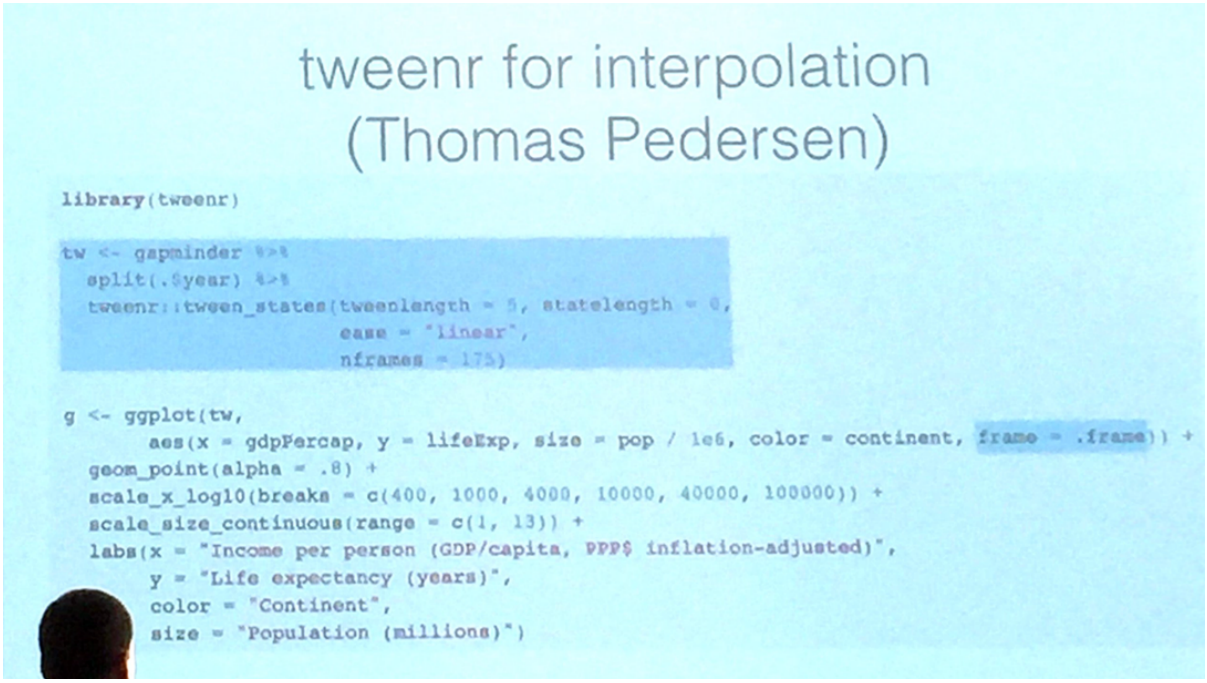
Mapping time to `aes(frame= )`

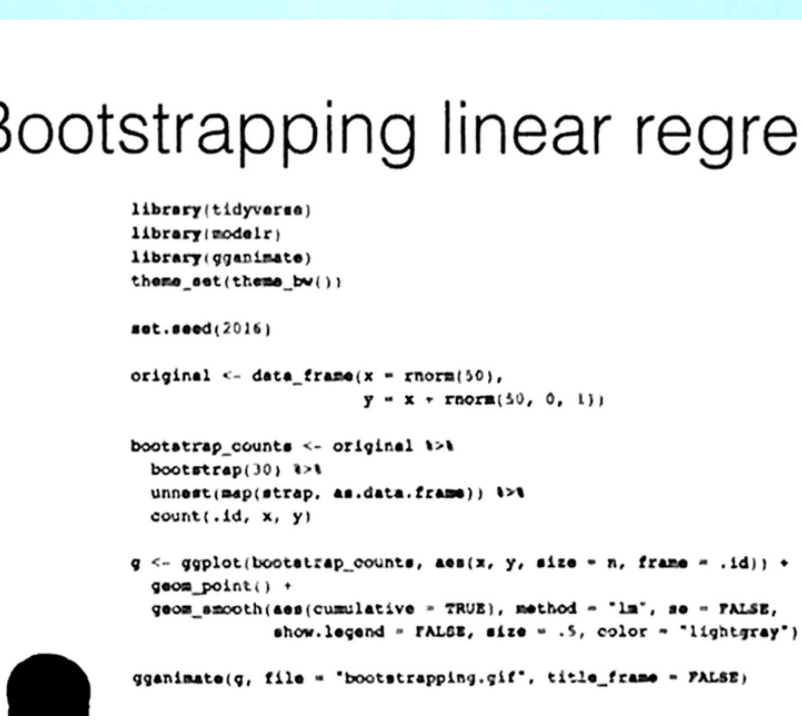
Checkout `tweenr` for smoothing animations

When you see something build (like election forecasts over time)... you develop a visceral connection to it... you feel the ups and downs

Grab locally weighted regression examples

Animations are "transparent" ... equations are opaque





Everything Else from Day IV

- Think about repurposing candlesticks for non financy-things (mood?)
- Look at `ggproto` and `productplots` for creating extensible geoms_
- Check out: QuantEcon from the Julia guy (Spencer Lyon)