Speeding up Dashboards & Case Studies

March 15, 2021

DATA 531 - Lecture 7

20 minutes

Part 1: Maintaining Dashboards

Source: 512pixels, Apple WWDC 2009

For your projects…

1. Implement known bug-fixes and work-arounds.
2. Implement feedback you receive from peers during feedback session.
3. Clean-up and refactor code to make it more readable.
4. Reformat your code and add docstrings (PEP8 style).
5. If needed, reorganize and restructure your files/directory structure to separate different structures (see the [Dash docs - Structuring a Multi-Page App) for suggested organizations).
6. Apply principles of creating effective dashboards.
7. Implement any new features (if time permits).

Refactor code 1

Let's say we had a dropdown of all the countries.
Do we really need to list every country in a dictionary?

```
dcc.Dropdown(
    options=[
        {'label': 'Afghanistan', 'value':'Afghanistan'},
        {'label': 'Argentina', 'value':'Argentina'},
        {'label': 'Aruba', 'value':'Aruba'},
        {'label': 'Australia', 'value':'Australia'},
        {'label': 'Austria', 'value':'Austria'},
        {'label': 'Bahamas', 'value':'Bahamas'},
        {'label': 'Bangladesh', 'value':'Bangladesh'},
        {'label': 'Barbados', 'value':'Barbados'},
        {'label': 'Belgium', 'value':'Belgium'},
        {'label': 'Bolivia', 'value':'Bolivia'},
        {'label': 'Brazil', 'value':'Brazil'},
        {'label': 'Canada', 'value':'Canada'},
        {'label': 'Chile', 'value':'Chile'},
        {'label': 'China', 'value':'China'},
        {'label': 'Colombia', 'value':'Colombia'},
        {'label': 'Costa Rica', 'value':'Costa Rica'},
        {'label': 'Croatia', 'value':'Croatia'},
        {'label': 'Cuba', 'value':'Cuba'},
        {'label': 'Dominican Republic', 'value':'Dominican Republic'},
        {'label': 'Ecuador', 'value':'Ecuador'},
        {'label': 'Egypt', 'value':'Egypt'},
        {'label': 'El Salvador', 'value':'El Salvador'},
        {'label': 'Finland', 'value':'Finland'},
        {'label': 'France', 'value':'France'},
        {'label': 'Georgia', 'value':'Georgia'},
        {'label': 'Germany', 'value':'Germany'},
```

Refactor code 1

Let's say we had a dropdown of all the countries.
Do we really need to list every country in a dictionary?

```
dcc.Dropdown(
    options=[
        {'label': 'Afghanistan', 'value':'Afghanistan'},
        {'label': 'Argentina', 'value':'Argentina'},
        {'label': 'Aruba', 'value':'Aruba'},
        {'label': 'Australia', 'value':'Australia'},
        {'label': 'Austria', 'value':'Austria'},
        {'label': 'Bahamas', 'value':'Bahamas'},
        {'label': 'Bangladesh', 'value':'Bangladesh'},
        {'label': 'Barbados', 'value':'Barbados'},
        {'label': 'Belgium', 'value':'Belgium'},
        {'label': 'Bolivia', 'value':'Bolivia'},
        {'label': 'Brazil', 'value':'Brazil'},
        {'label': 'Canada', 'value':'Canada'},
        {'label': 'Chile', 'value':'Chile'},
        {'label': 'China', 'value':'China'},
        {'label': 'Colombia', 'value':'Colombia'},
        {'label': 'Costa Rica', 'value':'Costa Rica'},
        {'label': 'Croatia', 'value':'Croatia'},
        {'label': 'Cuba', 'value':'Cuba'},
        {'label': 'Dominican Republic', 'value':'Dominican Republic'},
        {'label': 'Ecuador', 'value':'Ecuador'},
        {'label': 'Egypt', 'value':'Egypt'},
        {'label': 'El Salvador', 'value':'El Salvador'},
        {'label': 'Finland', 'value':'Finland'},
        {'label': 'France', 'value':'France'},
        {'label': 'Georgia', 'value':'Georgia'},
        {'label': 'Germany', 'value':'Germany'},
```

There is a better way!

Python List Comprehension

```
country_options = [{'label' : ctry, 'value' : ctry} for ctry in countries]
country_options
```

```
[{'label': 'Afghanistan', 'value': 'Afghanistan'},
 {'label': 'Argentina', 'value': 'Argentina'},
 {'label': 'Aruba', 'value': 'Aruba'},
 {'label': 'Australia', 'value': 'Australia'},
 {'label': 'Austria', 'value': 'Austria'},
```

Milestone 3 - Refactor code 1

Let's say we had a dropdown of all the countries.
Do we really need to list every country in a dictionary?

```
dcc.Dropdown(
    options=[
        {'label': 'Afghanistan', 'value':'Afghanistan'},
        {'label': 'Argentina', 'value':'Argentina'},
        {'label': 'Aruba', 'value':'Aruba'},
        {'label': 'Australia', 'value':'Australia'},
        {'label': 'Austria', 'value':'Austria'},
        {'label': 'Bahamas', 'value':'Bahamas'},
        {'label': 'Bangladesh', 'value':'Bangladesh'},
        {'label': 'Barbados', 'value':'Barbados'},
        {'label': 'Belgium', 'value':'Belgium'},
        {'label': 'Bolivia', 'value':'Bolivia'},
        {'label': 'Brazil', 'value':'Brazil'},
        {'label': 'Canada', 'value':'Canada'},
        {'label': 'Chile', 'value':'Chile'},
        {'label': 'China', 'value':'China'},
        {'label': 'Colombia', 'value':'Colombia'},
        {'label': 'Costa Rica', 'value':'Costa Rica'},
        {'label': 'Croatia', 'value':'Croatia'},
        {'label': 'Cuba', 'value':'Cuba'},
        {'label': 'Dominican Republic', 'value':'Dominican Republic'},
        {'label': 'Ecuador', 'value':'Ecuador'},
        {'label': 'Egypt', 'value':'Egypt'},
        {'label': 'El Salvador', 'value':'El Salvador'},
        {'label': 'Finland', 'value':'Finland'},
        {'label': 'France', 'value':'France'},
        {'label': 'Georgia', 'value':'Georgia'},
        {'label': 'Germany', 'value':'Germany'},
```

There is a better way!

Dictionary inside list comprehension

```
country_options = [{'label' : ctry, 'value' : ctry} for ctry in countries]
country_options
```

```
[{'label': 'Afghanistan', 'value': 'Afghanistan'},
 {'label': 'Argentina', 'value': 'Argentina'},
 {'label': 'Aruba', 'value': 'Aruba'},
 {'label': 'Australia', 'value': 'Australia'},
 {'label': 'Austria', 'value': 'Austria'},
```

Refactor code 1

Let's say we had a dropdown of all the countries.
Do we really need to list every country in a dictionary?

```
dcc.Dropdown(
    options=[
        {'label': 'Afghanistan', 'value':'Afghanistan'},
        {'label': 'Argentina', 'value':'Argentina'},
        {'label': 'Aruba', 'value':'Aruba'},
        {'label': 'Australia', 'value':'Australia'},
        {'label': 'Austria', 'value':'Austria'},
        {'label': 'Bahamas', 'value':'Bahamas'},
        {'label': 'Bangladesh', 'value':'Bangladesh'},
        {'label': 'Barbados', 'value':'Barbados'},
        {'label': 'Belgium', 'value':'Belgium'},
        {'label': 'Bolivia', 'value':'Bolivia'},
        {'label': 'Brazil', 'value':'Brazil'},
        {'label': 'Canada', 'value':'Canada'},
        {'label': 'Chile', 'value':'Chile'},
        {'label': 'China', 'value':'China'},
        {'label': 'Colombia', 'value':'Colombia'},
        {'label': 'Costa Rica', 'value':'Costa Rica'},
        {'label': 'Croatia', 'value':'Croatia'},
        {'label': 'Cuba', 'value':'Cuba'},
        {'label': 'Dominican Republic', 'value':'Dominican Republic'},
        {'label': 'Ecuador', 'value':'Ecuador'},
        {'label': 'Egypt', 'value':'Egypt'},
        {'label': 'El Salvador', 'value':'El Salvador'},
        {'label': 'Finland', 'value':'Finland'},
        {'label': 'France', 'value':'France'},
        {'label': 'Georgia', 'value':'Georgia'},
        {'label': 'Germany', 'value':'Germany'},
```

There is a better way!

Dictionary inside list comprehension

```
country_options = [{'label' : ctry, 'value' : ctry} for ctry in countries]
country_options
```

```
[{'label': 'Afghanistan', 'value': 'Afghanistan'},
 {'label': 'Argentina', 'value': 'Argentina'},
 {'label': 'Aruba', 'value': 'Aruba'},
 {'label': 'Australia', 'value': 'Australia'},
 {'label': 'Austria', 'value': 'Austria'},
```

List comprehension with `zip()`

```
country_labels = [c.upper() for c in countries]
country_options = [{'label' : lab,
                    'value' : ctry} for ctry,lab in zip(countries,
                                                        country_labels)]
country_options
```

```
[{'label': 'AFGHANISTAN', 'value': 'Afghanistan'},
 {'label': 'ARGENTINA', 'value': 'Argentina'},
 {'label': 'ARUBA', 'value': 'Aruba'},
 {'label': 'AUSTRALIA', 'value': 'Australia'},
 {'label': 'AUSTRIA', 'value': 'Austria'},
 {'label': 'BAHAMAS', 'value': 'Bahamas'},
 {'label': 'BANGLADESH', 'value': 'Bangladesh'},
```

Python: List, Dict, Set Comprehensions
R: Apply functions

Refactor code 2

Current

```
- app.py (big file)
- data
    |- dat.csv
    |- README
- ProcFile
- requirements.txt
- README
```

Structuring a Multi-page app in Python and R

Option 1

```
- app.py
- index.py
- apps
    |-- __init__.py
    |-- app1.py
    |-- app2.py
- data
    |-- dat.csv
    |-- README
- ProcFile
- requirements.txt
- README
```

Structuring a Multi-page app in Python and R

Refactor code 2

Option 1

```
- app.py
- index.py
- apps
    |-- __init__.py
    |-- app1.py
    |-- app2.py
- data
    |-- dat.csv
    |-- README
- ProcFile
- requirements.txt
- README
```

Structuring a Multi-page app in Python and R

app.py

```python
import dash

external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = dash.Dash(__name__, external_stylesheets=external_stylesheets)
server = app.server
app.config.suppress_callback_exceptions = True
```

Refactor code 2

Option 1

```
- app.py
- index.py
- apps
    |-- __init__.py
    |-- app1.py
    |-- app2.py
- data
    |-- dat.csv
    |-- README
- ProcFile
- requirements.txt
- README
```

Structuring a Multi-page app in Python and R

app.py

```
import dash

external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = dash.Dash(__name__, external_stylesheets=external_stylesheets)
server = app.server
app.config.suppress_callback_exceptions = True
```

apps/app1.py

```
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output

from app import app

layout = html.Div([
    html.H3('App 1'),
    dcc.Dropdown(
```

Key: import app.py as a module

Refactor code 2

Option 1

```
- app.py
- index.py
- apps
    |-- __init__.py
    |-- app1.py
    |-- app2.py
- data
    |-- dat.csv
    |-- README
- ProcFile
- requirements.txt
- README
```

Structuring a Multi-page app in Python and R

app.py

```
import dash

external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = dash.Dash(__name__, external_stylesheets=external_stylesheets)
server = app.server
app.config.suppress_callback_exceptions = True
```

apps/app1.py

```
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output

from app import app

layout = html.Div([
    html.H3('App 1'),
    dcc.Dropdown(
```

Key: import app.py as a module

index.py

index.py loads different apps on different urls

Refactor code 3

PEP8 for Python and Tidyverse for R

Use docstrings and comments to improve code

Python

```
def complex(real=0.0, imag=0.0):
    """Form a complex number.

    Keyword arguments:
    real -- the real part (default 0.0)
    imag -- the imaginary part (default 0.0)
    """
    if imag == 0.0 and real == 0.0:
        return complex_zero
    ...
```

R

7.4 Documenting parameters

For most tags, like `@param`, `@seealso` and `@return`, the text should be a sentence, starting with a capital letter and ending with a full stop.

```
#' @param key The bare (unquoted) name of the column whose values will be used
#'   as column headings.
```

If some functions share parameters, you can use `@inheritParams` to avoid duplication of content in multiple places.

```
#' @inheritParams argument function_to_inherit_from
```

Use GitHub issues for project communication!

Need to implement hot reloading support in Dash for R #125

New issue

Open rpkyle opened this issue on Sep 3 · 4 comments

rpkyle commented on Sep 3 • edited ▾ Member +😀 ···

Dash for Python offers a lovely "hot-reloading" feature, as described in plotly/dash#66, plotly/dash#362, and plotly/dash-renderer#73. Supporting this as in the Python implementation would require

- creating a hash at app initialization, and regenerating it on file change
- adding a route to serve the hash
- front-end hot reloading
- back-end hot reloading (reload the app by restarting the `fiery` session)
- adding parameters for the dev tools UI: `dev_tools_hot_reload`, `dev_tools_hot_reload_interval`, `dev_tools_hot_reload_watch_interval`, `dev_tools_silence_routes_logging`

On the Python side, we use the Flask debug reloader to reset the hash. `fiery` doesn't have native support for something like this feature, but it does allow "Custom Events" which can be manipulated to serve a similar purpose.

State-preserving hot reloading would be really nice to have, as noted in plotly/dash#460, but it has not been implemented in Dash for Python yet either.

@alexcjohnson

👍 1

 **rpkyle** self-assigned this on Sep 3

 **rpkyle** added **enhancement** **Stage: In Progress** labels on Sep 4

Assignees

 **rpkyle**

Labels

enhancement

feature request

Projects

None yet

Milestone

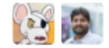
No milestone

Notifications Customize

🔇× Unsubscribe

You're receiving notifications because you commented.

2 participants



Use GitHub issues for project communication!

Integration with geopandas #588 #818

Open **iliatimofeev** wants to merge 24 commits into `altair-viz:master` from `iliatimofeev:it-#588-geopandas`

Conversation 51 | Commits 24 | Checks 0 | Files changed 7 | +323 −19

iliatimofeev commented on May 5, 2018 • edited — Contributor

Integration with geopandas(fix #588)
Now GeoDataFrame is valid Data type for alt.Chart

```
import altair as alt
import geopandas as gpd

alt.renderers.enable('notebook')

world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))

#GeoDataFrame could be passed as usual pd.DataFrame
alt.Chart(world[world.continent!='Antarctica']).mark_geoshape(
).project(
).encode(
    # for GeoDataFrame fields shorthand infer types as for regular pd.DataFrame
    color='pop_est',
    tooltip='id:Q' # GeoDataFrame.index is accessible as id
).properties(
    width=500,
    height=300
)
```

Reviewers
jakevdp
pratapvardhan

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Notifications — Customize
Subscribe
You're not receiving notifications from this thread.

4 participants

Source: Altair PR 818

A plea…

- Move all “non-logistics” chatter from Slack to GitHub.com issues
- Abide by the code of conduct
- Get used to commenting publicly
- Think of the history…

20 minutes

Part 2: Speeding up Dashboards

Many ways to speed up dashboards, here are three:

1. Filtering data
2. Strategies for aggregating data
3. Precomputing data

Filtering Data

- Are you using all of your data ?
    - If not, set up a reproducible script to drop unused columns, and rows…
    - If yes, add components (dropdowns/slider) to show less data overall

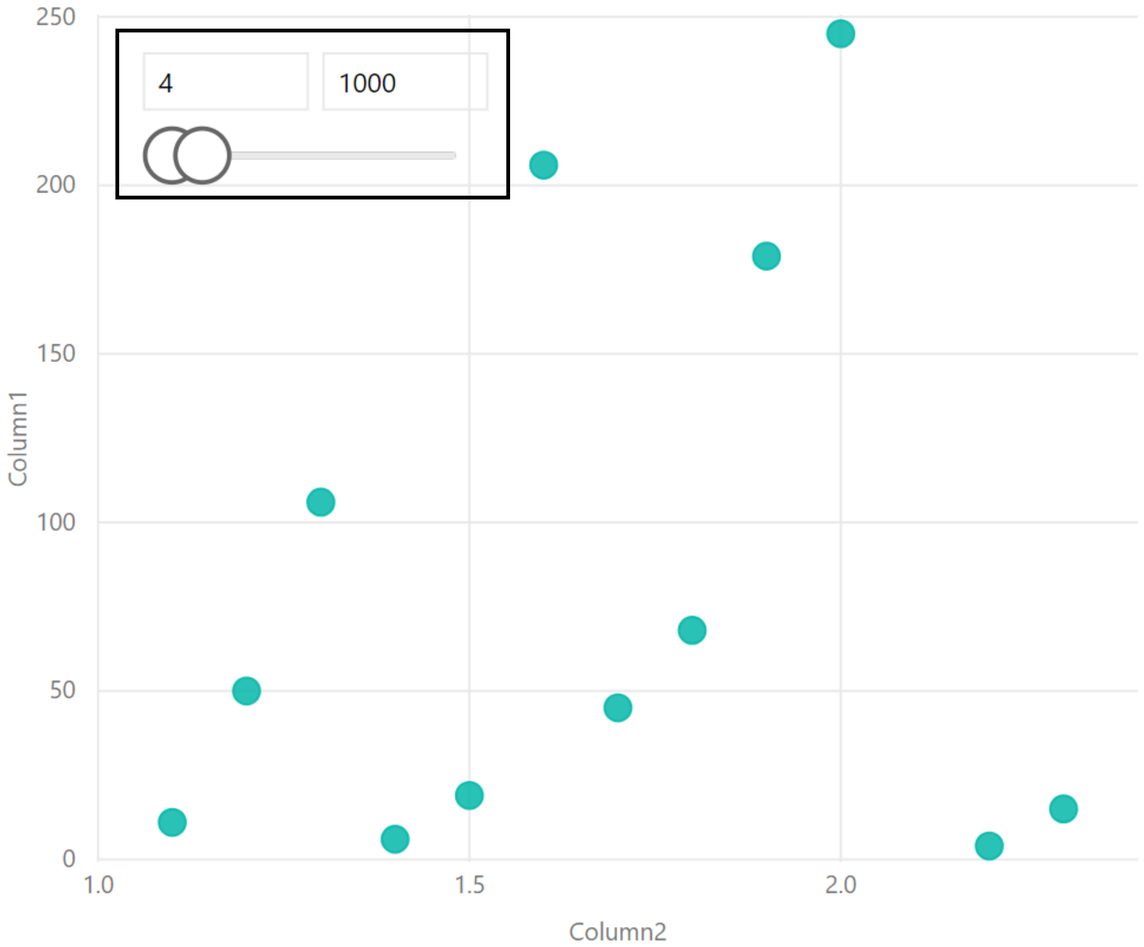
- Remember: purpose-driven dashboards. Is all your data useful to answer your research question?

Dealing with outliers

Source: DSCI 532 lecture slides from 2019/19 Cydney Nielsen

Dealing with outliers

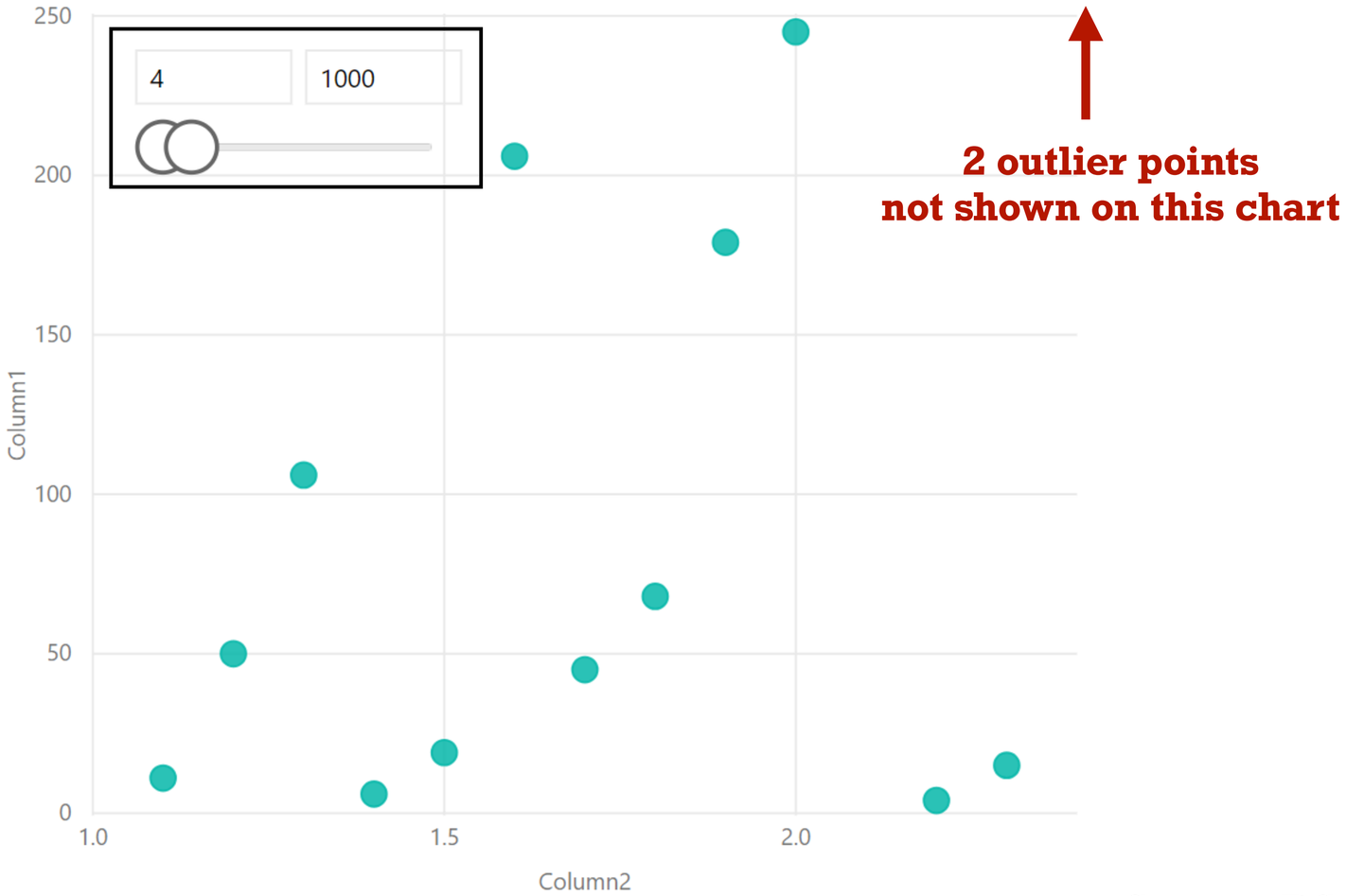
interactive filtering is another solution - *appropriate when want to remove outliers*



Source: DSCI 532 lecture slides from 2019/19
Cydney Nielsen

Dealing with outliers

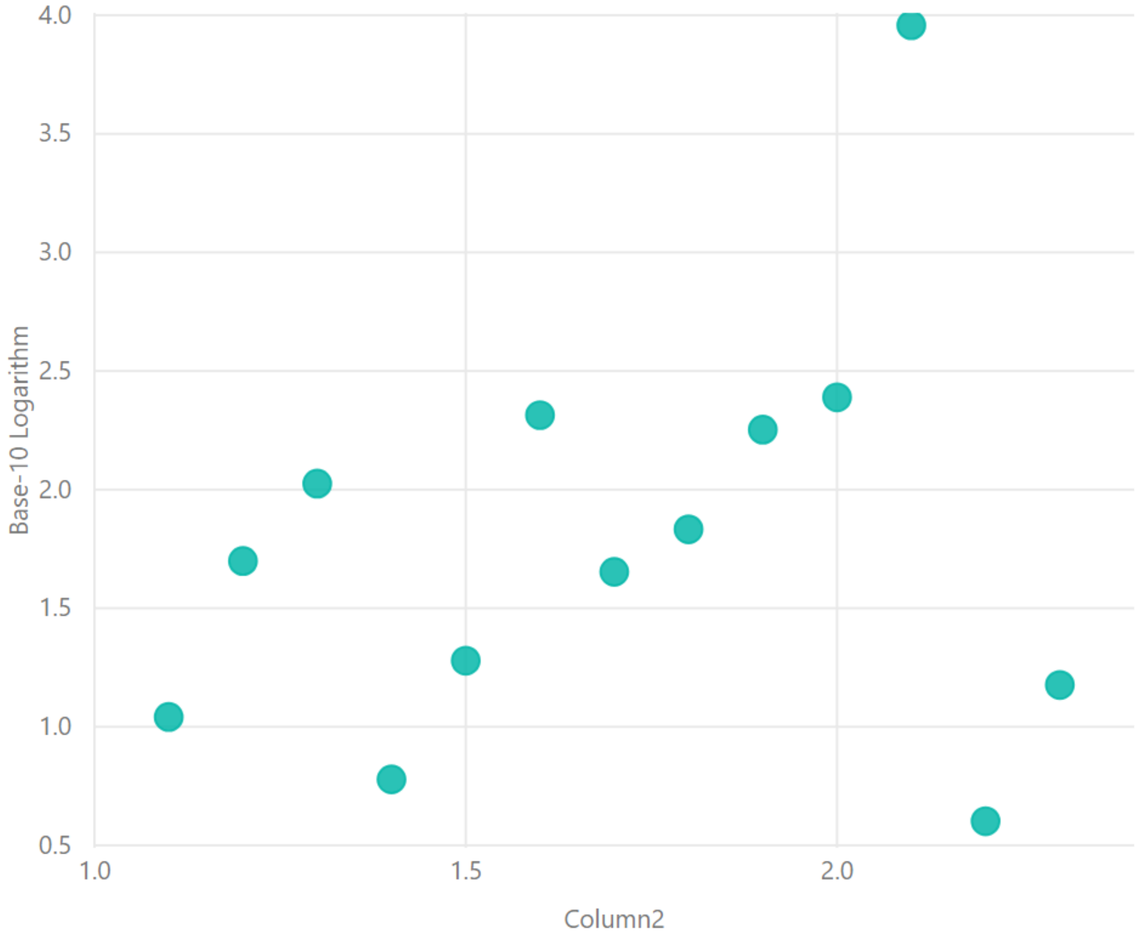
interactive filtering is another solution - *appropriate when want to remove outliers*



Source: DSCI 532 lecture slides from 2019/19
Cydney Nielsen

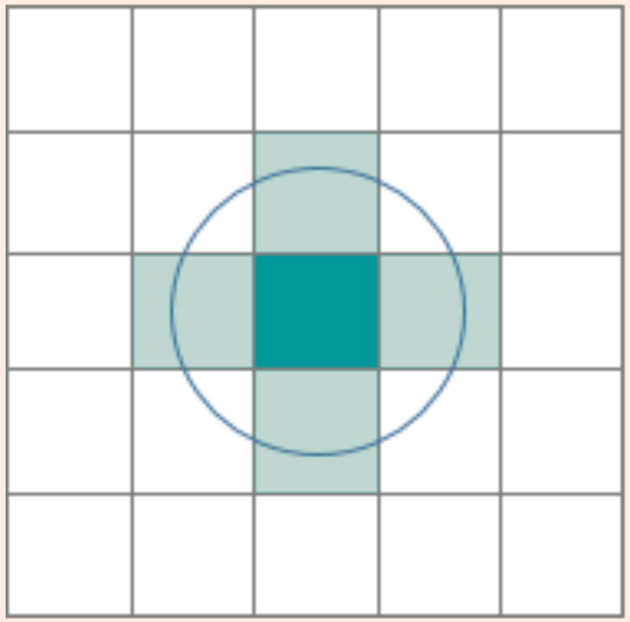
Dealing with outliers

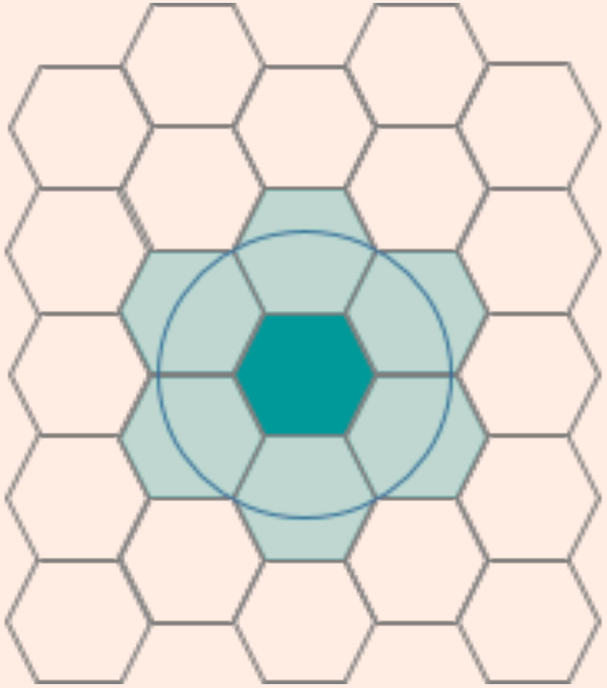
log scale helps - *appropriate if you want to keep all points*



Source: DSCI 532 lecture slides from 2019/19
Cydney Nielsen

Why Hexagons?





Hexagons:

- Reduce sampling bias

- Curves in data more naturally shown

- Grids draw our eyes to straight, unbroken parallel lines

Source: Many more arguments presented here ArcGIS Pro

Pre-computing Data

- Use vectorized functions and avoid loops at all costs!

- Setup a script/notebook to do any heavy wrangling and processing outside your app

- Load data at app start; **re-structure** your data to serve queries!

Many ways to speed up dashboards, here are three:

1. Filtering data
2. Strategies for aggregating data
3. Precomputing data (vectorize functions)
4. Compute additional columns in advance (separate from plotting)

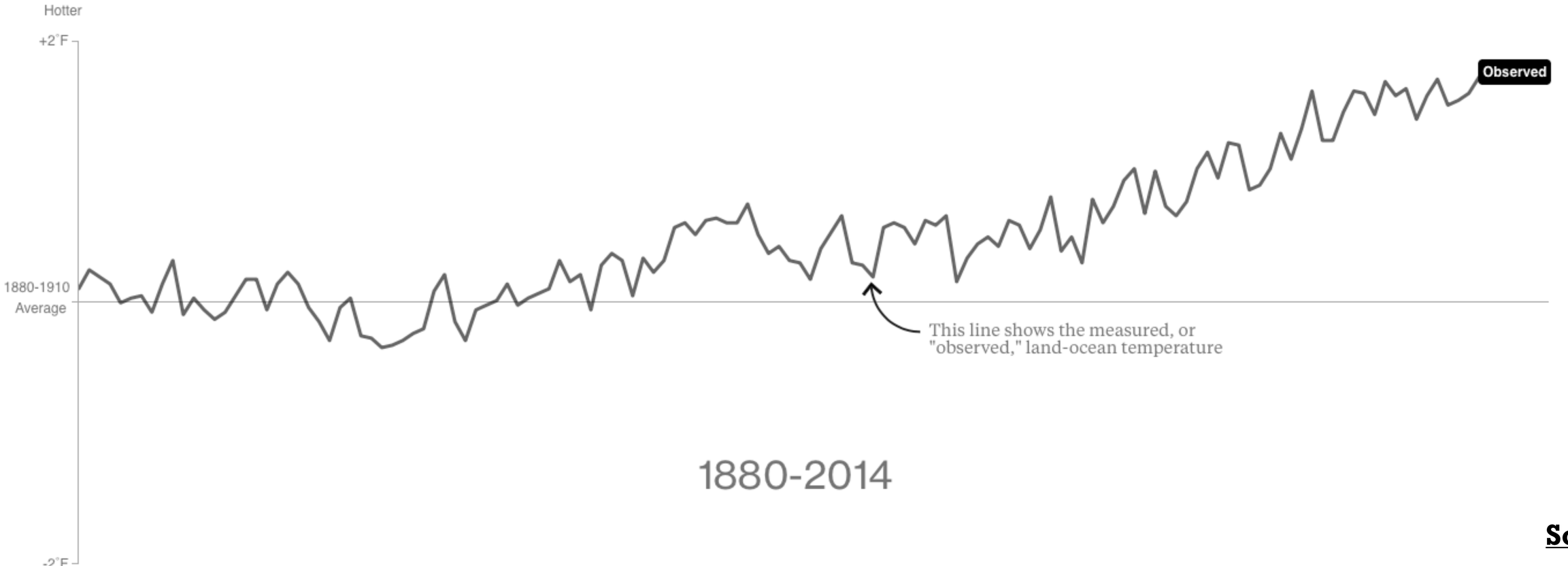
20 minutes

Part 3: Dashboards case study

What's Really Warming the World?

By Eric Roston and Blacki Migliozzi | June 24, 2015

Skeptics of manmade climate change offer various natural causes to explain why the Earth has warmed 1.4 degrees Fahrenheit since 1880. But can these account for the planet's rising temperature? Scroll down to see how much different factors, both natural and industrial, contribute to global warming, based on findings from NASA's Goddard Institute for Space Studies.



Source

PARABLE OF THE POLYGONS

A PLAYABLE POST ON THE SHAPE OF SOCIETY

by vi hart + nicky case

español | deutsch | français | português | 日本語 | 中文 | polski
italiano | magyar | nederlands | हिन्दी | čeština | Русский | العربيّة | Українська

This is a story of how harmless choices can make a harmful world.

These little cuties are 50% Triangles, 50% Squares, and 100% slightly shapist.
But only slightly! In fact, every polygon *prefers* being in a diverse crowd:

Source

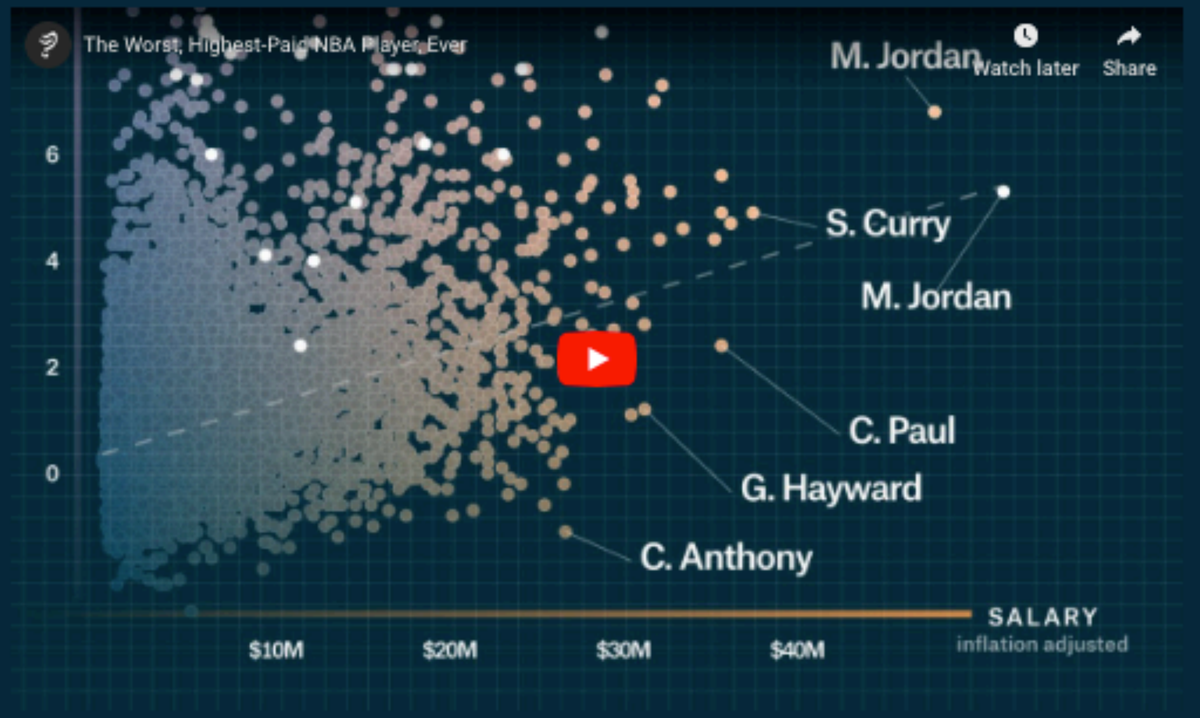
The Pudding

Finding the Worst, Highest-Paid NBA Player, Ever

Using advanced NBA stats to rank player performance against pay.

by Matt Daniels

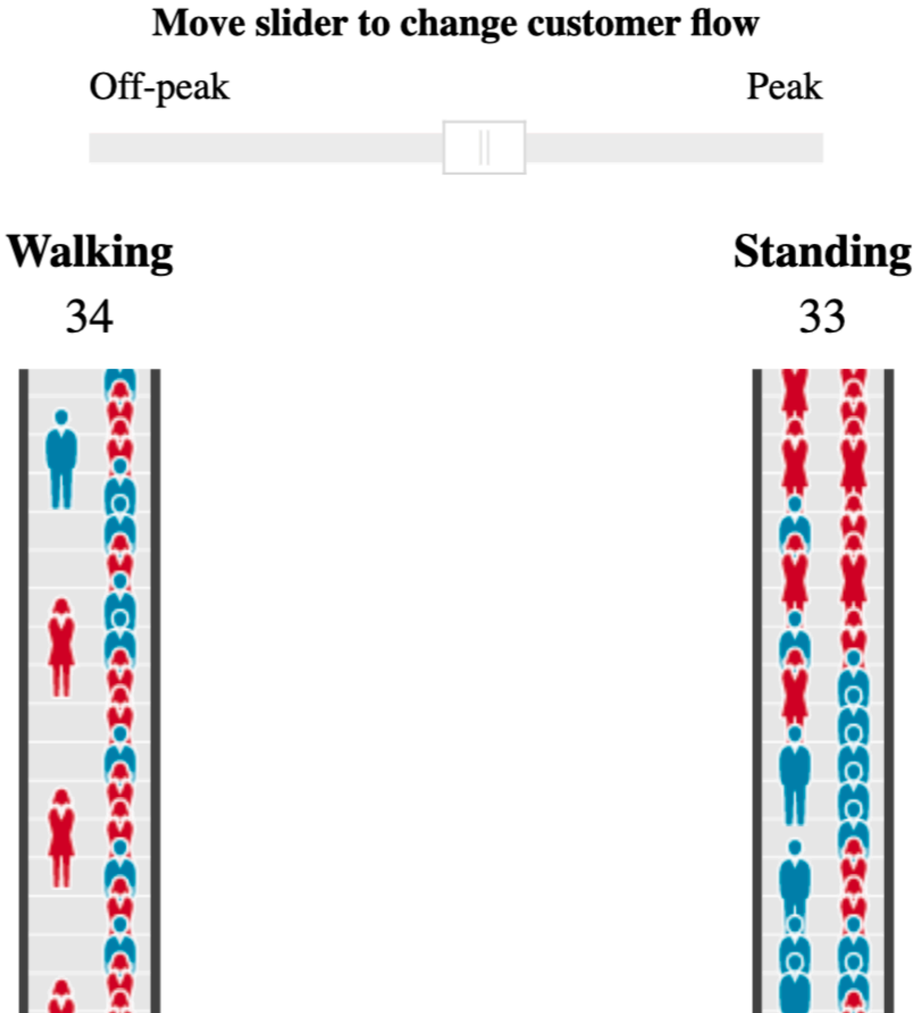
Watch the Video



Source

Does standing on both sides of the escalator work? Test it with our interactive simulator

The interactive below shows how the trial helped ease congestion - showing the number of people reaching the top of each escalator, and based on figures provided by TfL.



Source

You Draw It: How Family Income Predicts Children's College Chances

By **GREGOR AISCH, AMANDA COX and KEVIN QUEALY** MAY 28, 2015

How likely is it that children who grow up in very poor families go to college? How about children who grow up in very rich families?

We'd like you to **draw your guess** for every income level on the chart below.

If you think the chances of enrolling in college (or vocational school) are about the same for everyone, you should draw something like this: —— . If you think the odds are especially harsh for children from the poorest families, but higher for middle- and higher-income children, your drawing would instead look like this: ⁄ . Or here is one for a situation in which chances level off after a certain income threshold: ⁀ . Or for one that spikes ⁀ or dips ⁀ for the very richest.

Source

Can you form a stable government?

Combine parties as best you can to form a workable government. You need 323 votes, probably, to survive a confidence vote, but you may find that some parties get along together better than others

Choose your parties

Start by dragging either Labour or Conservatives

Ukip 3

SNP 52

PC 3

SDLP 3

Green 1

You have

582 seats

Reset

DUP 9

It's a bad match because ...

Source

Great list of fantastic interactive dashboards!

Rock 'n Poll

The power of Explorable Explanations

Maarten Lambrechts
@maartenzam

Mediafin

DataHarvest 2016

1

Dataharvest I: Rock 'n Poll

4 years ago 1 5,583

maartenzam PRO ★

maartenzam

Source

Summary

- Speed-up dashboards by precomputing, filtering, and aggregating data

- Use vectorized functions and packages like **purrr** and **dplyr**

- Practicing Dash callbacks in R

- Examples of story-telling with excellent dashboards