THE OHIO STATE UNIVERSITY
FISHER COLLEGE OF BUSINESS

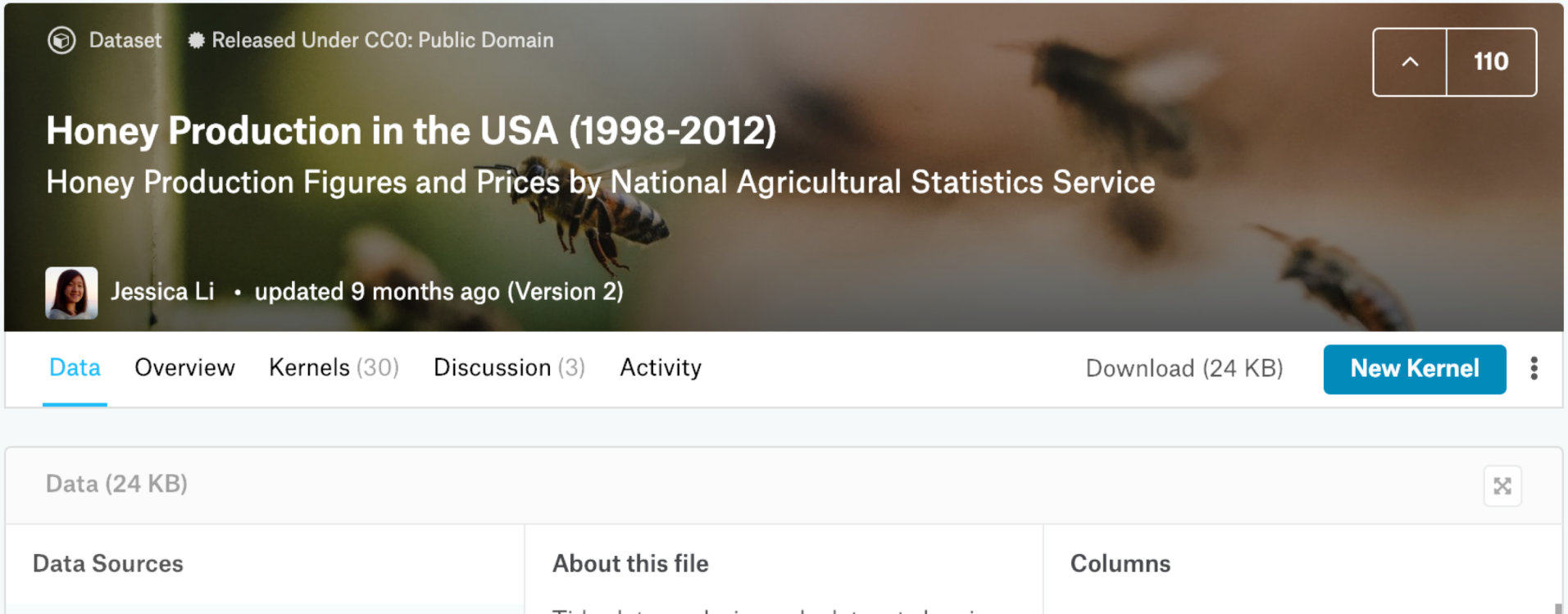
BUSMGT 7331: Descriptive Analytics and Visualization
Week 6

Interactivity and Dashboard Design

Hyunwoo Park
Fisher College of Business
The Ohio State University

Data set for this week

- Honey Production in the USA
  https://www.kaggle.com/jessicali9530/honey-production



Packages for this week

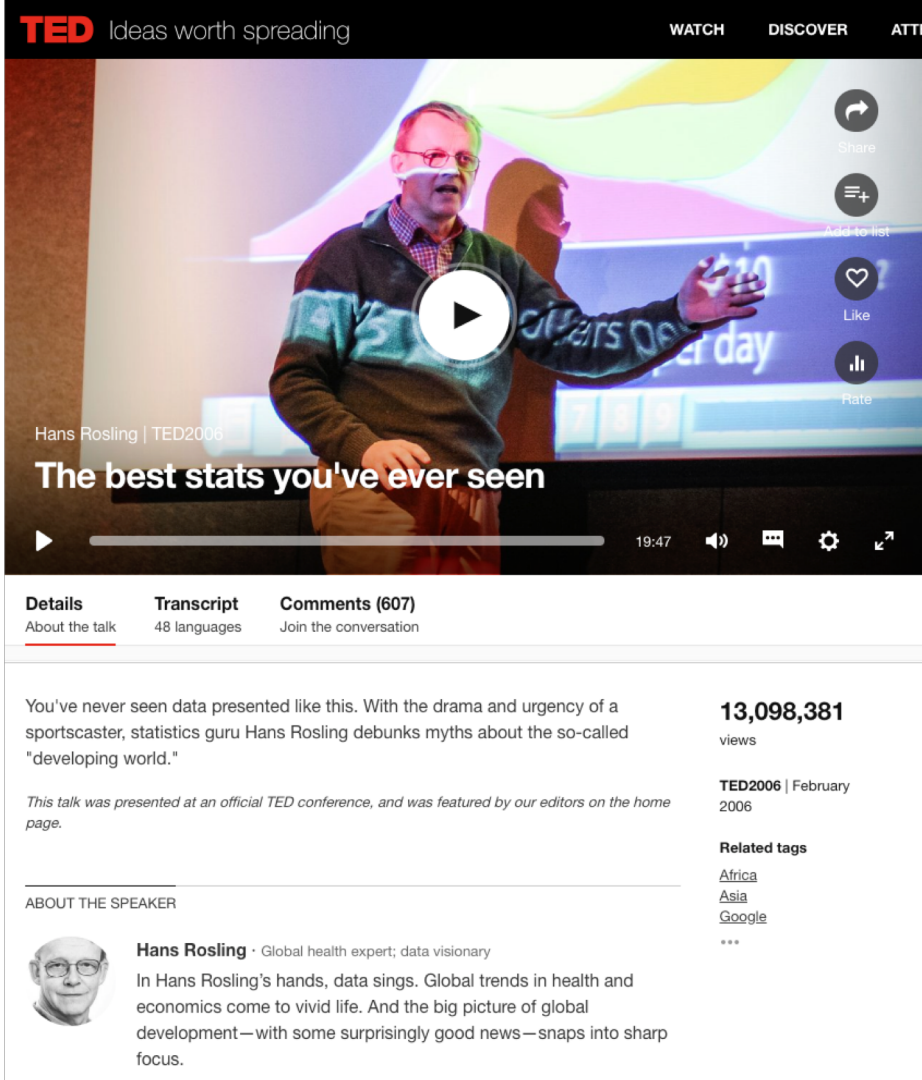
```
install.packages("gganimate")
install.packages("gapminder")
install.packages("gifski")
install.packages("plotly")
install.packages("leaflet")
install.packages("tigris")
install.packages("visNetwork")
install.packages("flexdashboard")
install.packages("shinydashboard")
install.packages("shiny")
```

Animated Visualization

Why animation?

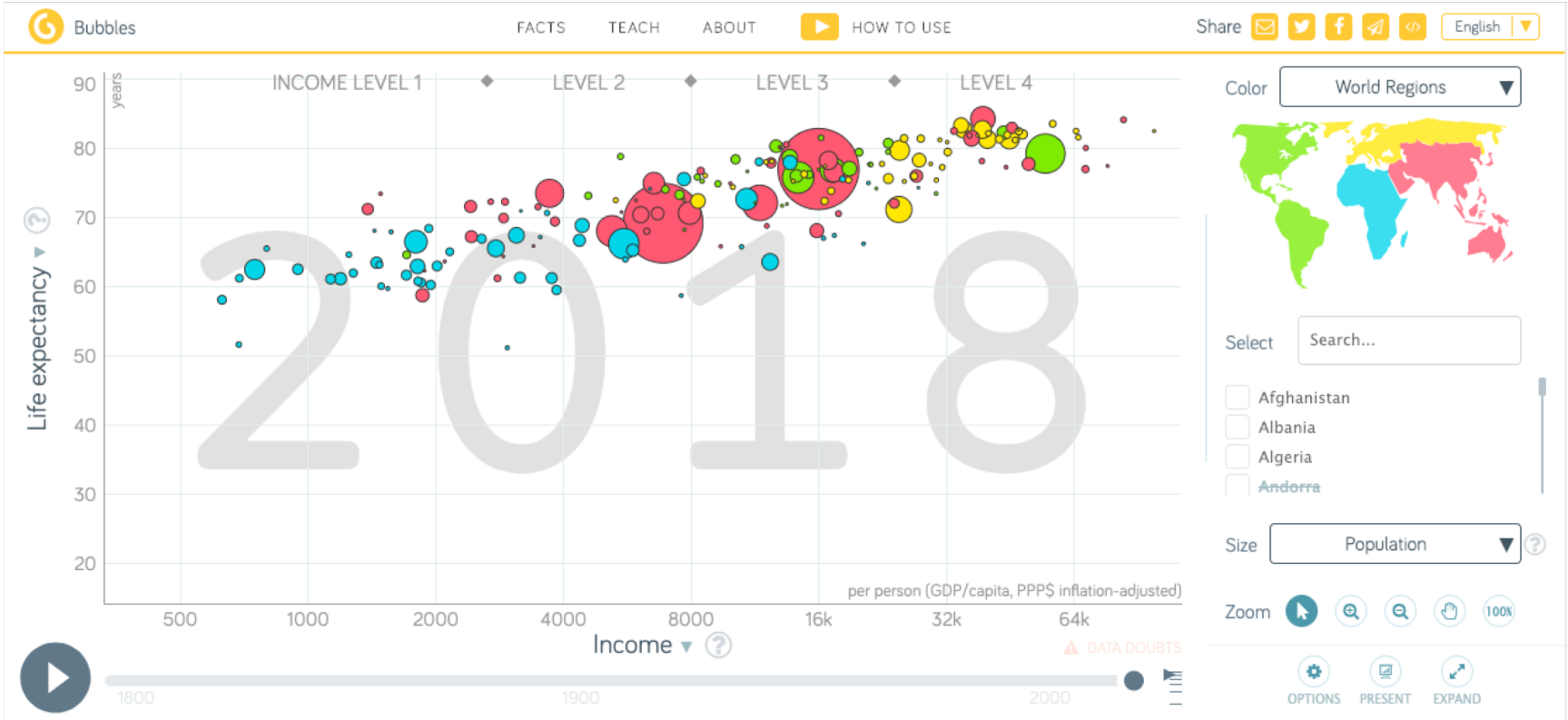
- This video shows the power of animated visualization and how interactivity can empower analytics consumers to investigate the data themselves.

- https://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen or https://youtu.be/hVimVzgtD6w



An example of animated visualization

- Gapminder https://www.gapminder.org/tools/



Why animation?

- Screen replaces paper, to some degree.
- More options to present “moving images” become available.
  - Gif images
  - YouTube
- Most important, the time dimension is now at our disposal when we use animation.
- Aesthetics are scarce resources when it comes to creating a visualization.
  Thus, having another aesthetical dimension is a big help.

gganimate

- Package developed by Thomas Lin Pedersen
- Readings and Tutorials
  - Overview of gganimate: https://gganimate.com/
  - Getting started from gganimate: https://gganimate.com/articles/gganimate.html
  - Github front page of gganimate: https://github.com/thomasp85/gganimate
- Basic Syntax
  - transition_*
    - transition_states()
    - transition_time()
  - ease_aes: linear, quadratic, cubic, sine, elastic, back, bounce
  - enter_* and exit_*

Replicating the GapMinder example

- Let's start with a static visualization that we know.

```
1   library(gapminder)
2
3   ggplot(gapminder, aes(gdpPercap, lifeExp, size=pop, fill=continent)) +
4     geom_point(alpha=.5, shape=21, color=gray(0.3)) +
5     scale_fill_manual(values=c("#33D9EB", "#90ED37", "#FF7185", "#FFE938",
    "#FF7185")) +
6     scale_size(range=c(1,16)) +
7     scale_x_log10(breaks=c(500,1000,2000,4000,8000,16000,32000,64000)) +
8     scale_y_continuous(breaks=seq(20,90,10)) +
9     labs(x="Income", y="Life expectancy") +
10    theme(panel.grid.minor = element_blank())
```

Does it look similar?

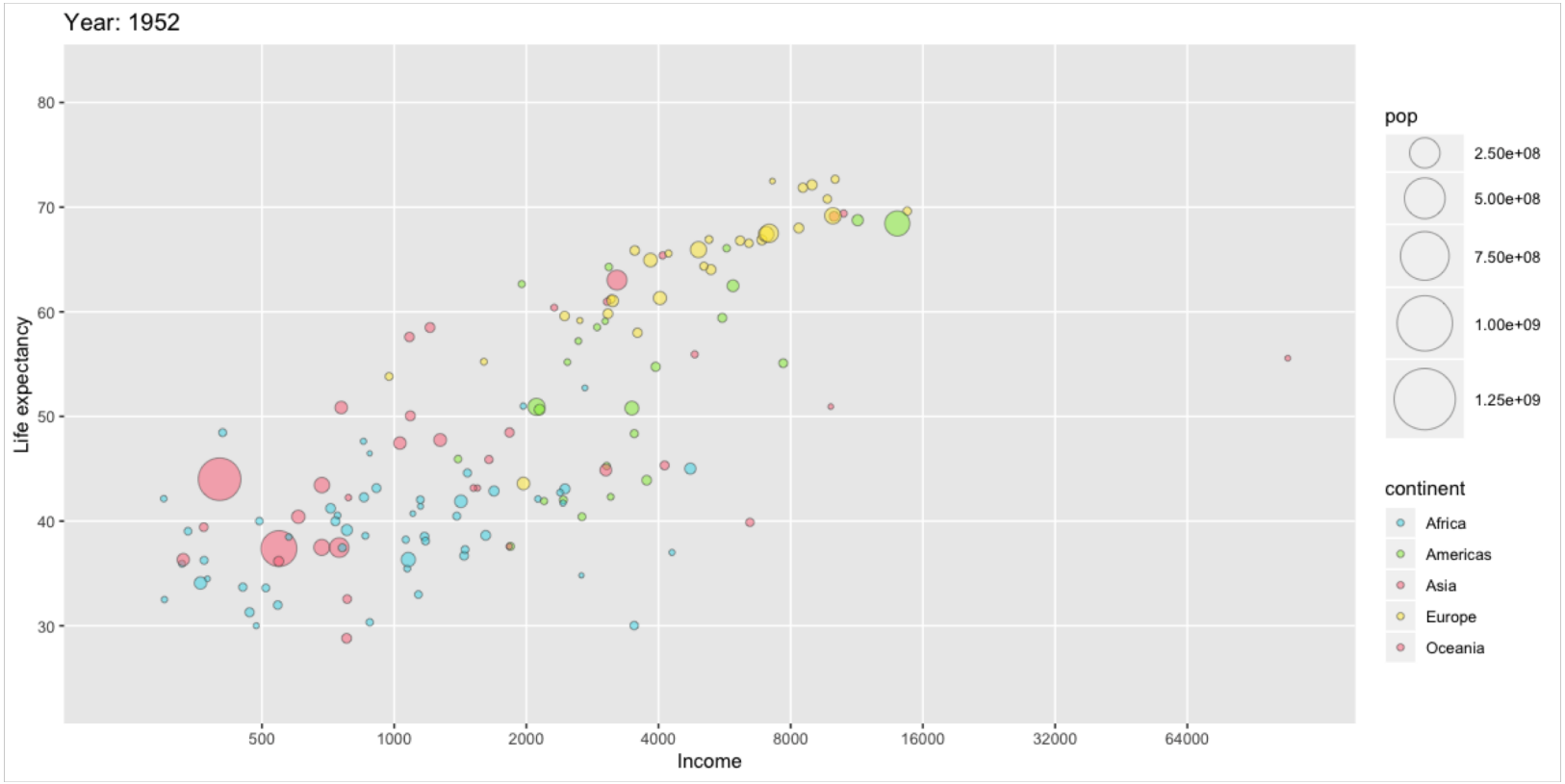
Let's animate this.

- In the previous visualization, we showed all years at the same time, which clutters the visualization. Animation can help.

```
p <- ggplot(gapminder, aes(gdpPercap, lifeExp, size=pop, fill=continent)) +
  geom_point(alpha=.5, shape=21, color=gray(0.3)) +
  scale_fill_manual(values=c("#33D9EB", "#90ED37", "#FF7185", "#FFE938",
"#FF7185")) +
  scale_size(range=c(1,16)) +
  scale_x_log10(breaks=c(500,1000,2000,4000,8000,16000,32000,64000)) +
  scale_y_continuous(breaks=seq(20,90,10)) +
  labs(title="Year: {frame_time}", x="Income", y="Life expectancy") +
  theme(panel.grid.minor = element_blank()) +
  transition_time(year) + ease_aes("linear")

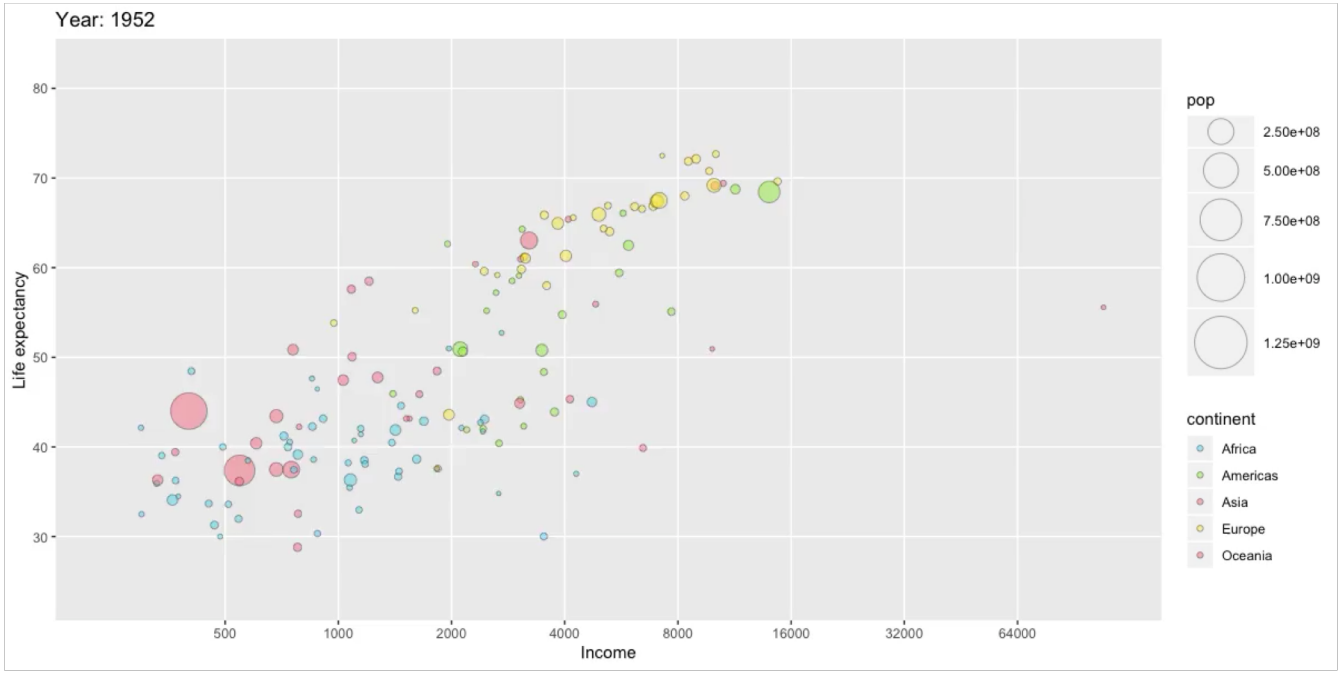
animate(p, width=1200, height=600, res=100)
anim_save("img/gapminder_animation.gif")
```

Animation output: gif



Using other renderers: mp4

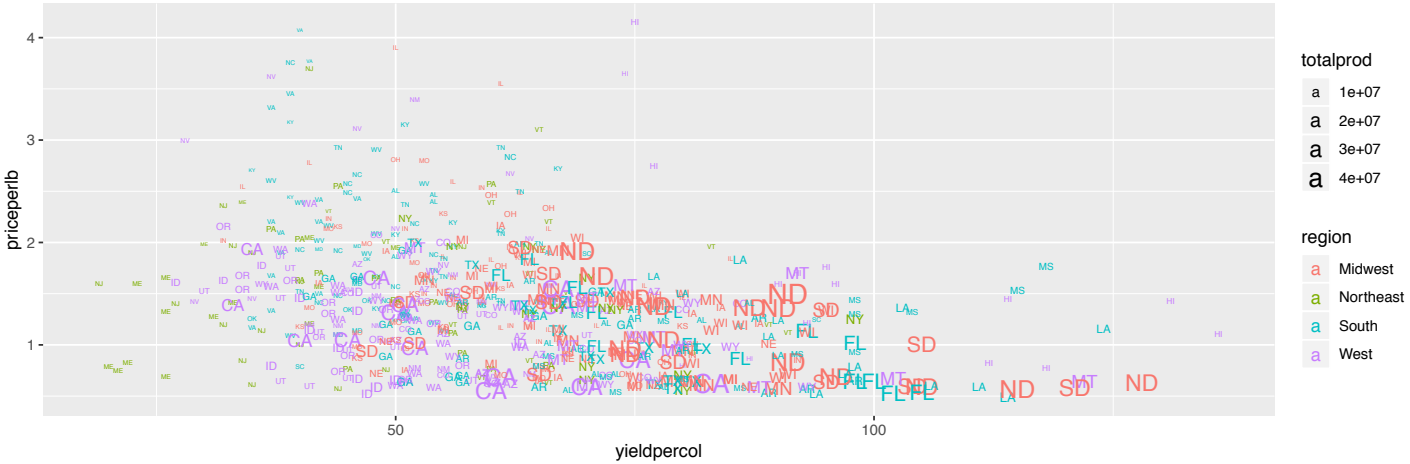
```
1  animate(p, width=1200, height=600, res=100, renderer=ffmpeg_renderer("mp4"))
2  anim_save("img/gapminder_animation.mp4")
```



Load and visualize this week's dataset on honey production.

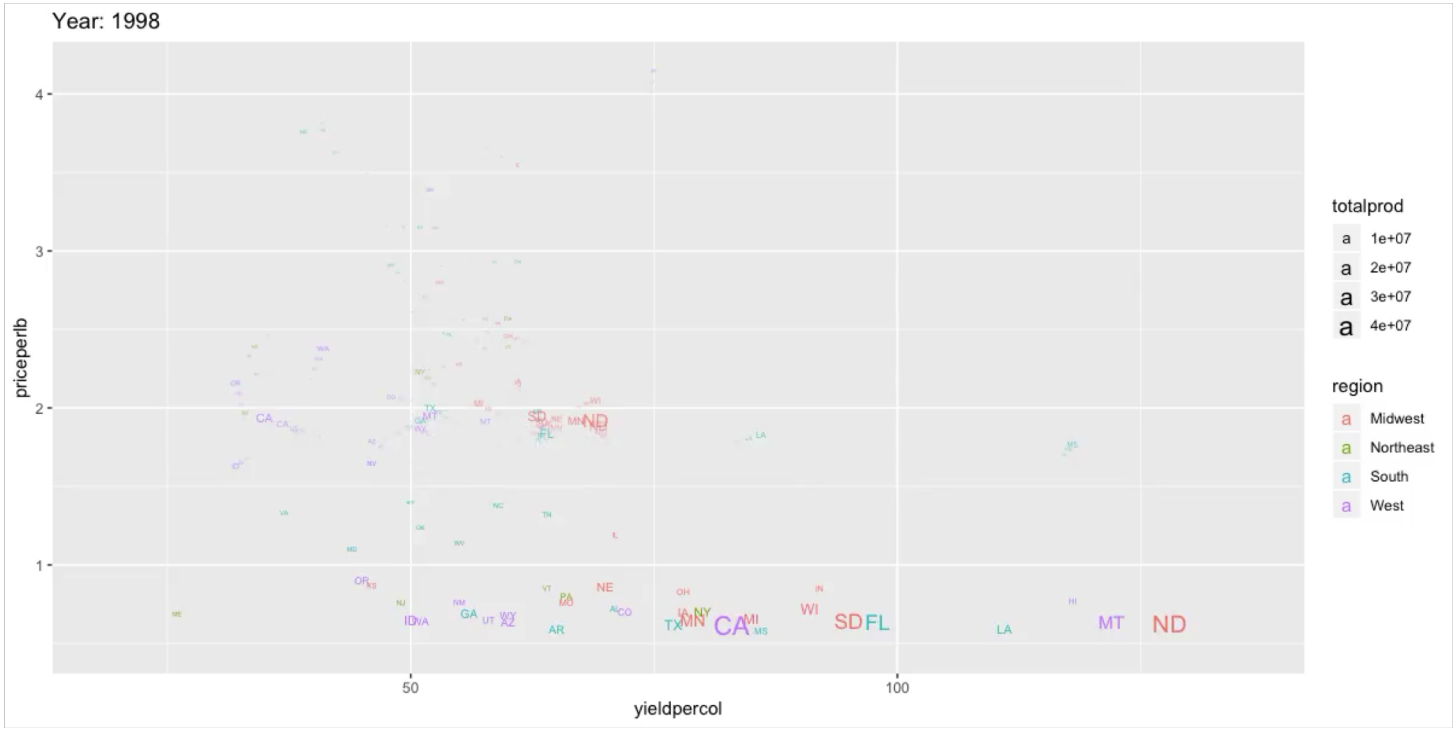
- State regions come from https://github.com/cphalpert/census-regions

```
1  regions <- read_csv("https://raw.githubusercontent.com/cphalpert/census-
2  regions/master/us%20census%20bureau%20regions%20and%20divisions.csv") %>%
     rename(name=State, state=`State Code`, region=Region, division=Division)
3  honey <- read_csv("data/honeyproduction.csv",
   col_types=cols(year=col_integer())) %>% left_join(regions)
4  ggplot(honey, aes(yieldpercol, priceperlb, color=region, label=state,
   size=totalprod)) + geom_text()
```



Let's animate with some trails.

```
1  animate(p, width=1200, height=600, res=100, renderer=ffmpeg_renderer("mp4"))
2  anim_save("img/gapminder_animation.mp4")
```



More fun examples of animated visualizations

- World Cup Goal Animation
  - https://github.com/thomasp85/gganimate/wiki/World-Cup-Goal-Animation
  - https://datascienceplus.com/animating-the-goals-of-the-world-cup-comparing-the-old-vs-new-gganimate-and-tweenr-api/
- Other examples compiled in gganimate wiki on Github.
  - https://github.com/thomasp85/gganimate/wiki
- My selections
  - https://github.com/thomasp85/gganimate/wiki/Optical-Illusion
  - https://github.com/thomasp85/gganimate/wiki/Moving-Hawaii-and-Alaksa
  - https://github.com/thomasp85/gganimate/wiki/Tracking-of-hurricanes-and-typhoons

Interactive Visualization

Motivation for interactivity in visualization

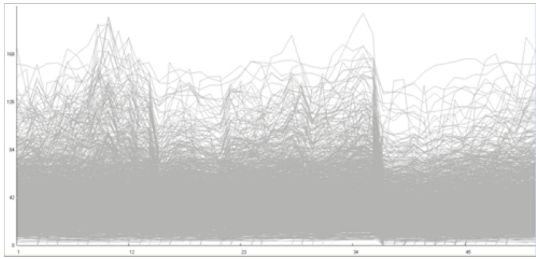
- Animation vs. interaction
  - Animation is a great way to exploit the unused “time” dimension in visualizing data.
  - However, animation may still be considered “static” in the sense that viewers can only see the data in a way that the creator wants them to see.
  - Interactivity turns those viewers into active investigators on the data by enabling them with proper tools.
- Why interactive visualization?
  - Supports exploratory needs of the analytics consumers
  - Facilitates user-initiated investigation of the data
  - Helps in a situation when there are too many details to visualize (i.e., limited real estate)

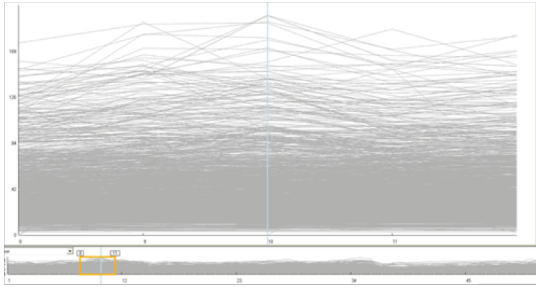
The Visual Information-Seeking Mantra

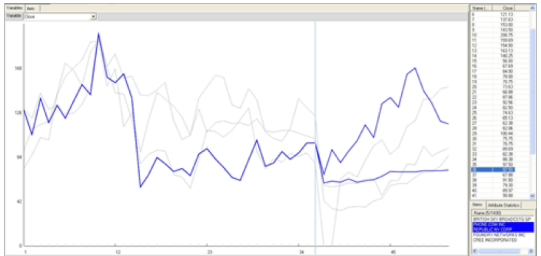
- Readings
  - [Required] Shneiderman, B. (1996). The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations.
    https://ieeexplore.ieee.org/abstract/document/545307
    http://info.ils.indiana.edu/~katy/S637-S11/Shneiderman96.pdf
  - [Optional] Yi, J.S., Kang, Y.A., Stasko, J., & Jacko, J. A. (2007). Toward a Deeper Understanding of the Role of Interaction in Information Visualization.
    https://ieeexplore.ieee.org/abstract/document/4376144
    https://users.cs.duke.edu/~rodger/jflappapers/Stasko2007.pdf
- The Mantra: “Overview first, zoom and filter, and details-on-demand”
  - Overview / Zoom / Filter / Details-on-demand / Relate / History / Extract
- An example of such visual analysis following the mantra
  - Few, S. (2006). The Surest Path to Visual Discovery.
    http://www.b-eye-network.com/view/2674

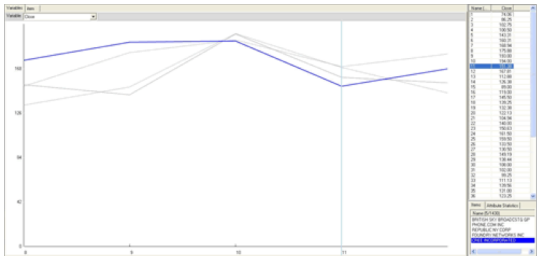
Visual information-seeking process

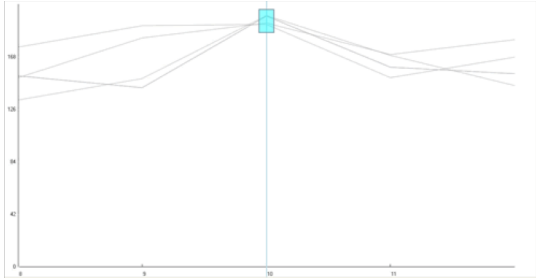
- All bullets below are from Shneiderman (1996) in verbatum. Figures are from Few (2006).
  - Overview: Gain an overview of the entire collection.
  - Zoom: Zoom in on items of interest.
  - Filter: filter out uninteresting items.
  - Details-on-demand: Select an item or group and get details when needed.
  - Relate: View relationships among items.
  - History: Keep a history of actions to support undo, replay, and progressive refinement.
  - Extract: Allow extraction of sub-collections and of the query parameters.











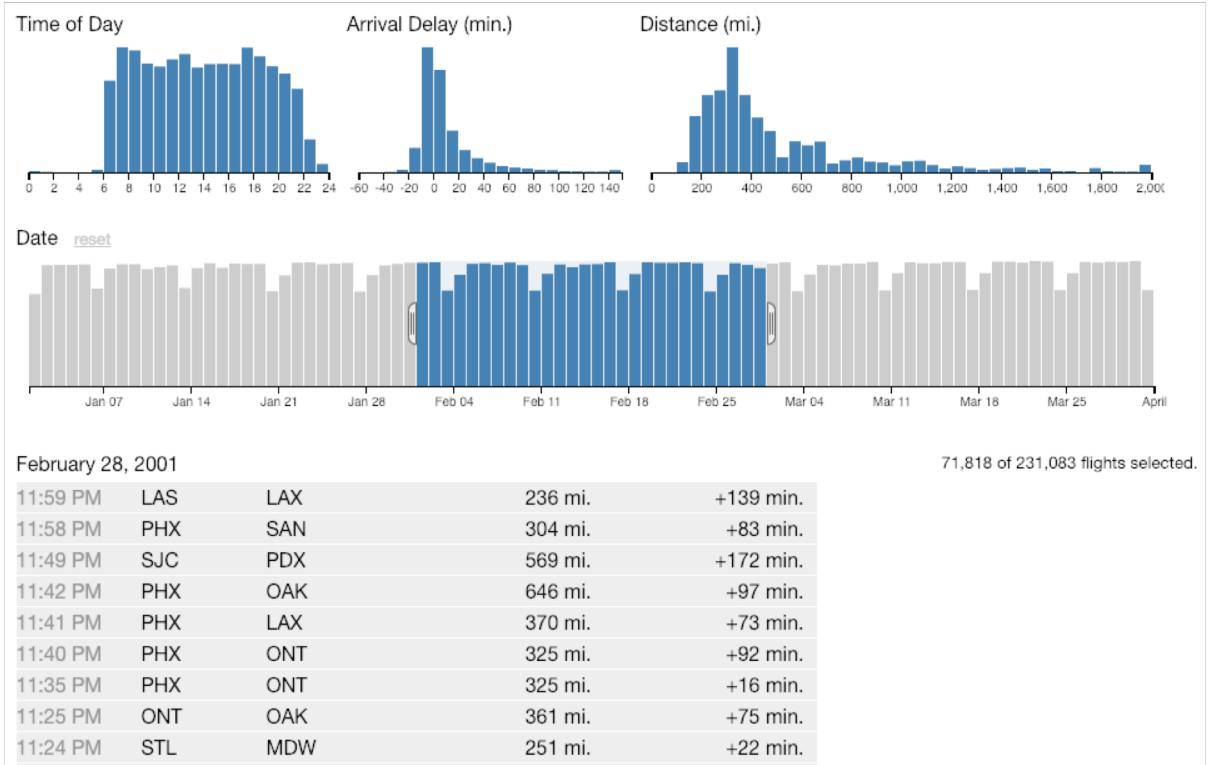
Interaction techniques in information visualization

https://infovis-wiki.net/wiki/Category:Interaction_Techniques

- (Coordinated) multiple views
  - https://infovis-wiki.net/wiki/Multiple_Views
- Dynamic query and filtering
  - https://infovis-wiki.net/wiki/Dynamic_query
  - https://infovis-wiki.net/wiki/Filtering
- Zoom
  - https://infovis-wiki.net/wiki/Zoom
  - https://infovis-wiki.net/wiki/Semantic_Zoom
- Details on demand
  - https://infovis-wiki.net/wiki/Details_on_demand
- Brushing and linking
  - https://infovis-wiki.net/wiki/Brushing
  - https://infovis-wiki.net/wiki/Linking_and_Brushing
- Focus+context
  - https://infovis-wiki.net/wiki/Focus-plus-Context
  - https://infovis-wiki.net/wiki/Fisheye_View
  - https://infovis-wiki.net/wiki/Magic_Lens

Coordinated multiple views & dynamic query and filtering

- http://square.github.io/crossfilter/



Zoom

- Geometric zoom: https://bl.ocks.org/mbostock/3680999
- Semantic zoom: https://bl.ocks.org/mbostock/3680957

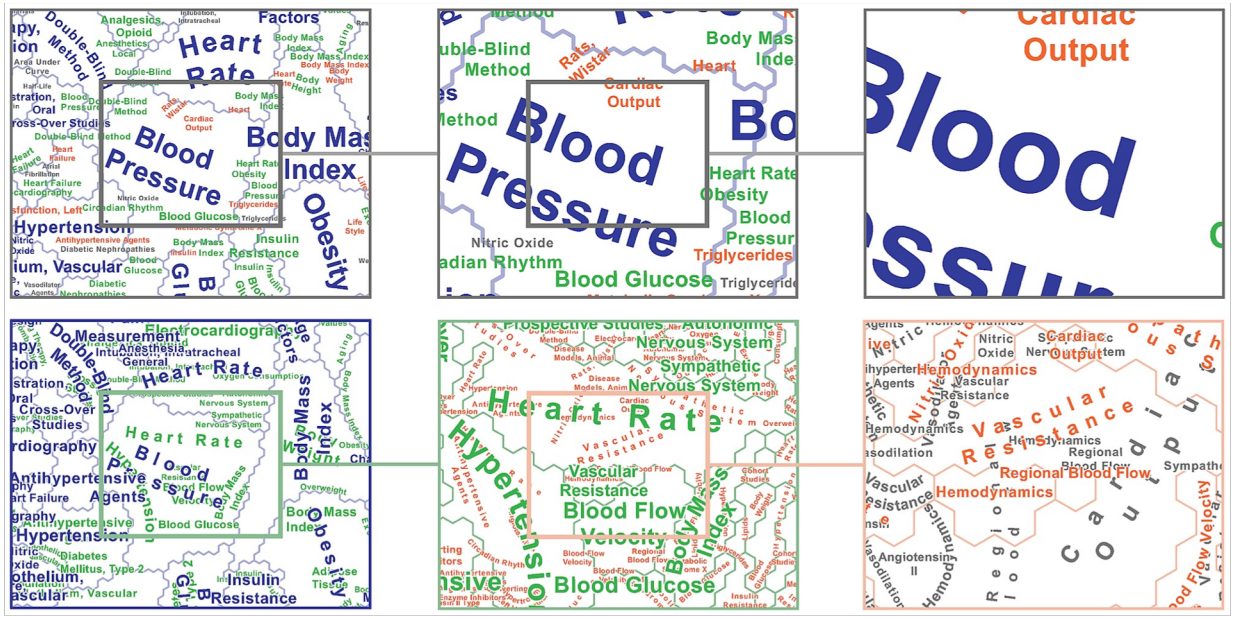
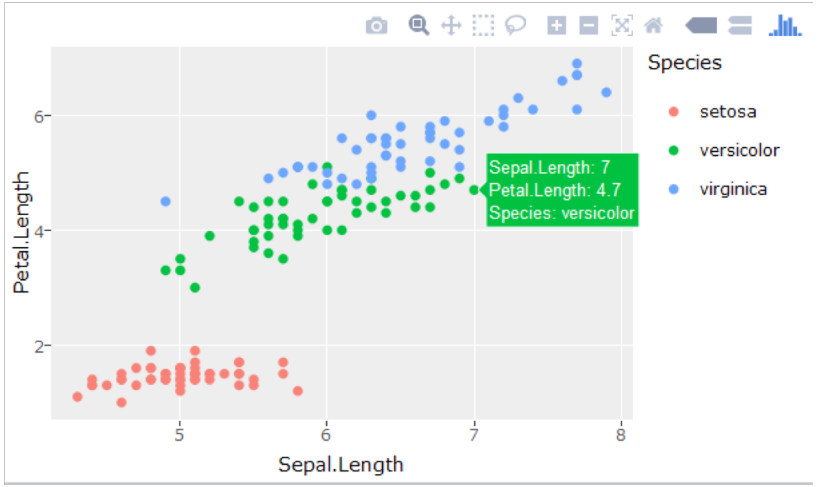
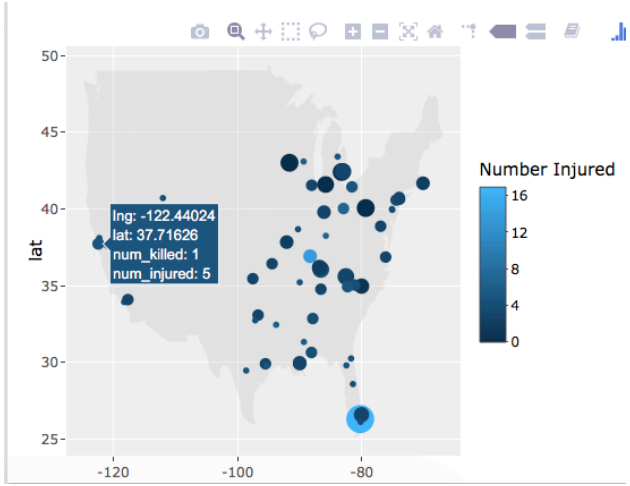


Figure from Skupin, A., R. Biberstine, J. and Börner, K. (2013) 'Geometric zooming versus semantic zooming.'. PLOS ONE. doi: 10.1371/journal.pone.0058779.g008.

Details-on-demand

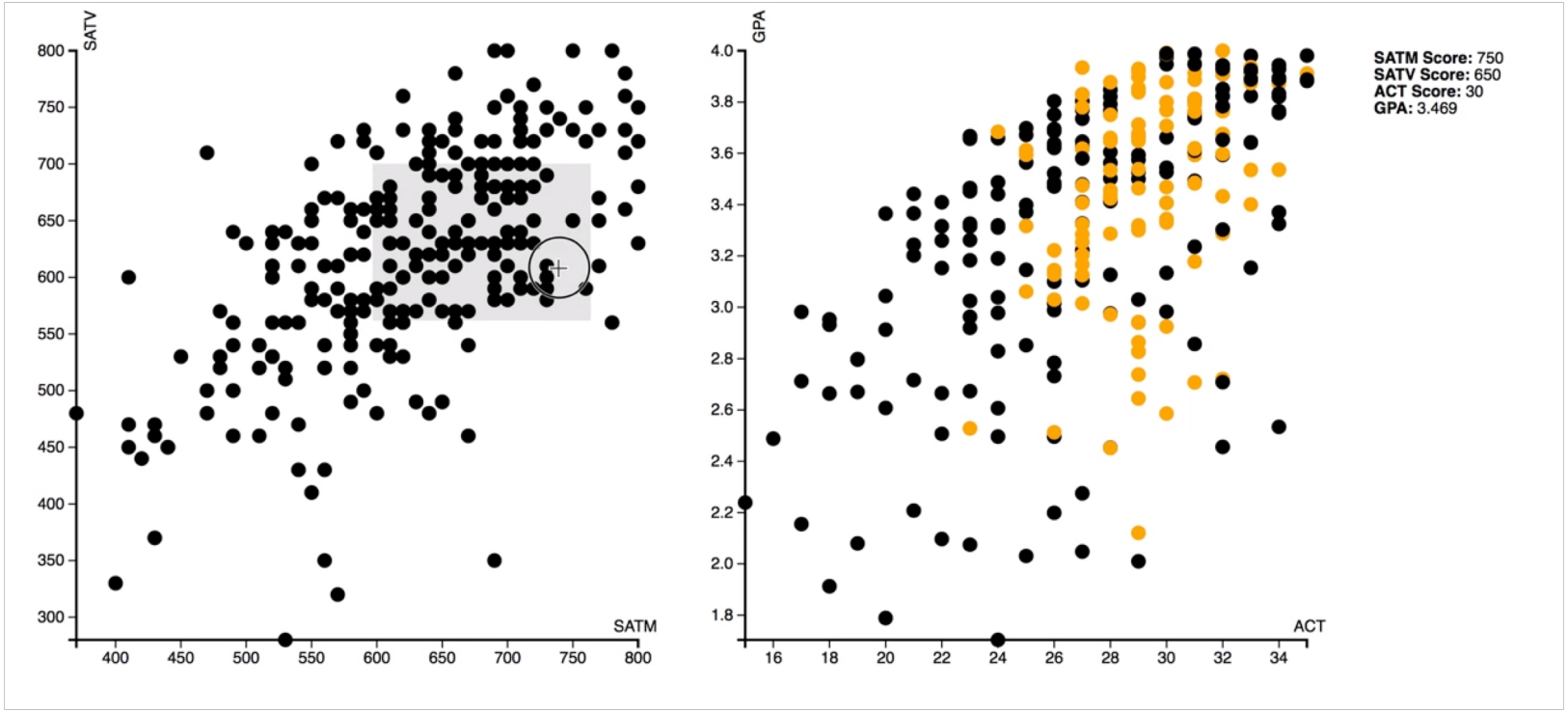
- Images from:
  - https://stackoverflow.com/questions/37465285/interactive-scatter-plots-in-r-overlay-hover-summary-tooltip-as-user-supplied-p
  - https://stackoverflow.com/questions/50222764/how-do-i-format-the-names-of-the-variables-in-the-r-plotly-tooltip





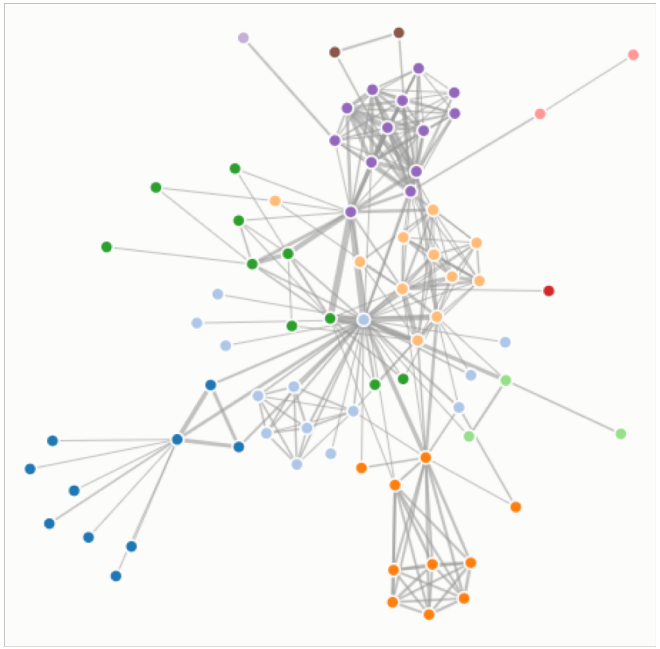
Brushing and linking

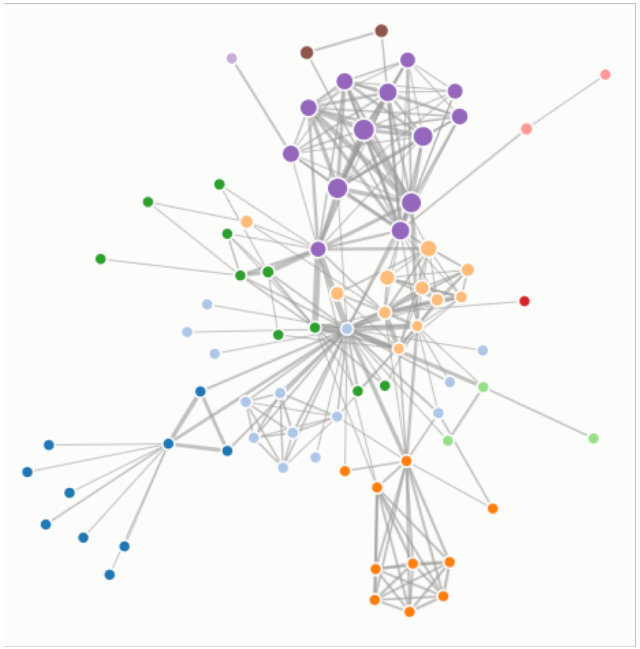
- https://www.youtube.com/watch?v=FYUj-kjl2lw



Focus+context technique: fisheye distortion

- https://bost.ocks.org/mike/fisheye/





Necessity of a user interface

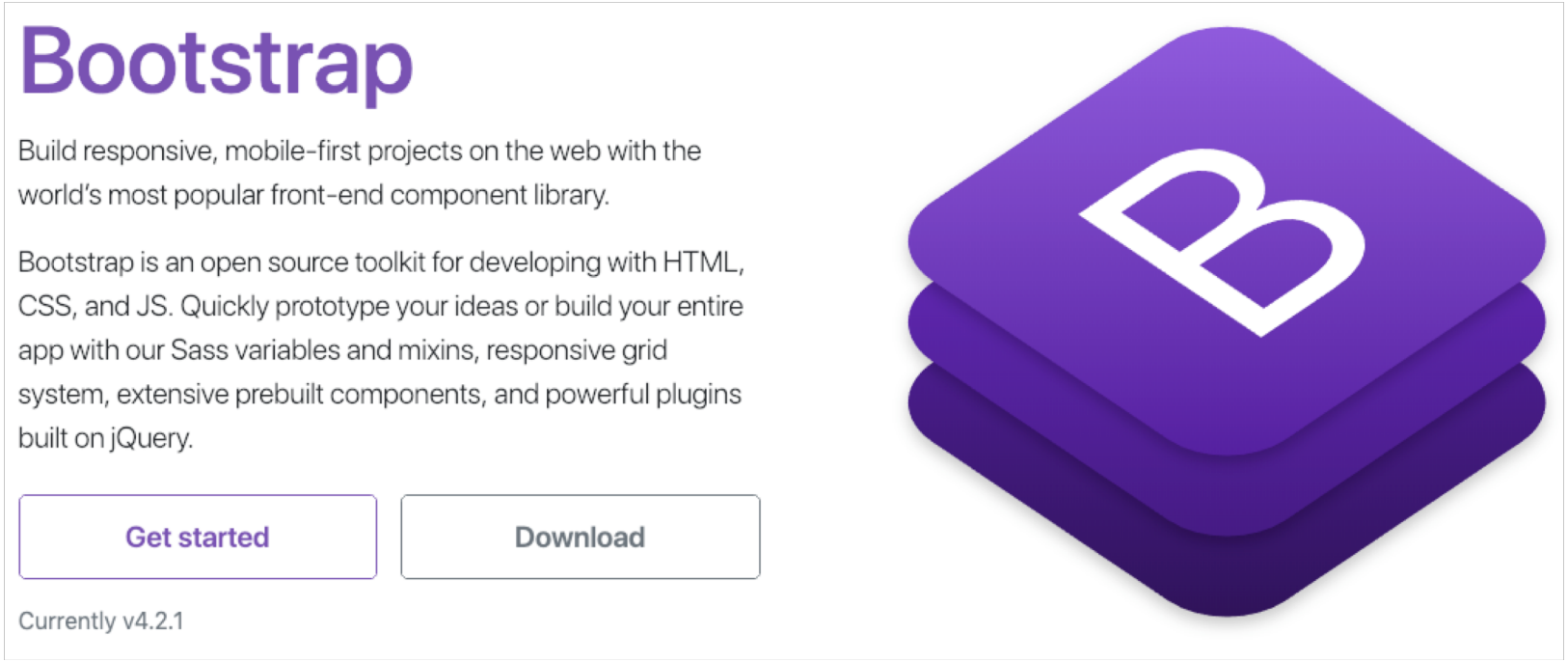
- The interaction techniques so far are ways to engage with the “visualization” itself to explore the data further, whether it is about filtering or getting more details.
- However, oftentimes, it’s useful to have a tool to express your query more explicitly.
- You may want to, for example,
  - select one or a number of specific categories.
  - set a precise range for a continuous variable.
  - search the dataset with a string.
- A user interface is a collection of methods to collect inputs from the user to allow oneself to express their intention in a structured manner.

User interface design basics

- Reading
  - https://www.usability.gov/what-and-why/user-interface-design.html
- List of user interface elements
  - https://www.usability.gov/how-to-and-tools/methods/user-interface-elements.html
  - **Input Controls:** checkboxes, radio buttons, dropdown lists, list boxes, buttons, toggles, text fields, date field
  - **Navigational Components:** breadcrumb, slider, search field, pagination, slider, tags, icons
  - **Informational Components:** tooltips, icons, progress bar, notifications, message boxes, modal windows
  - **Containers:** accordion

Bootstrap

- https://getbootstrap.com/



Explore UI components and themes in Bootstrap framework

- I do not expect you to learn how to use Bootstrap for this course.
- However, the current trend in building an interactive application is using the web technology—HTML, CSS, JavaScript.
- Shiny dashboard, which will be discussed later, also uses Bootstrap.
- Take a look at how Bootstrap styles its UI components here: https://getbootstrap.com/docs/4.2/components/
- Here are some free themes that differently style a Bootstrap-enabled application. https://bootswatch.com/

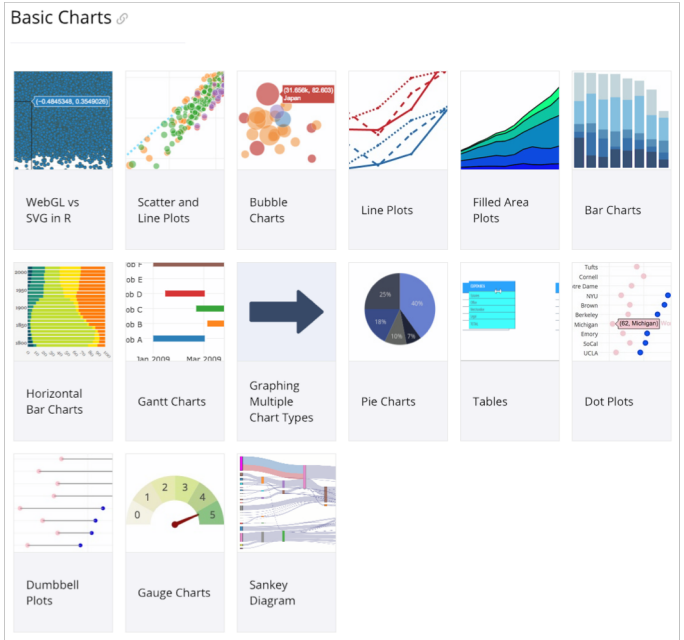
Adding Interactivity

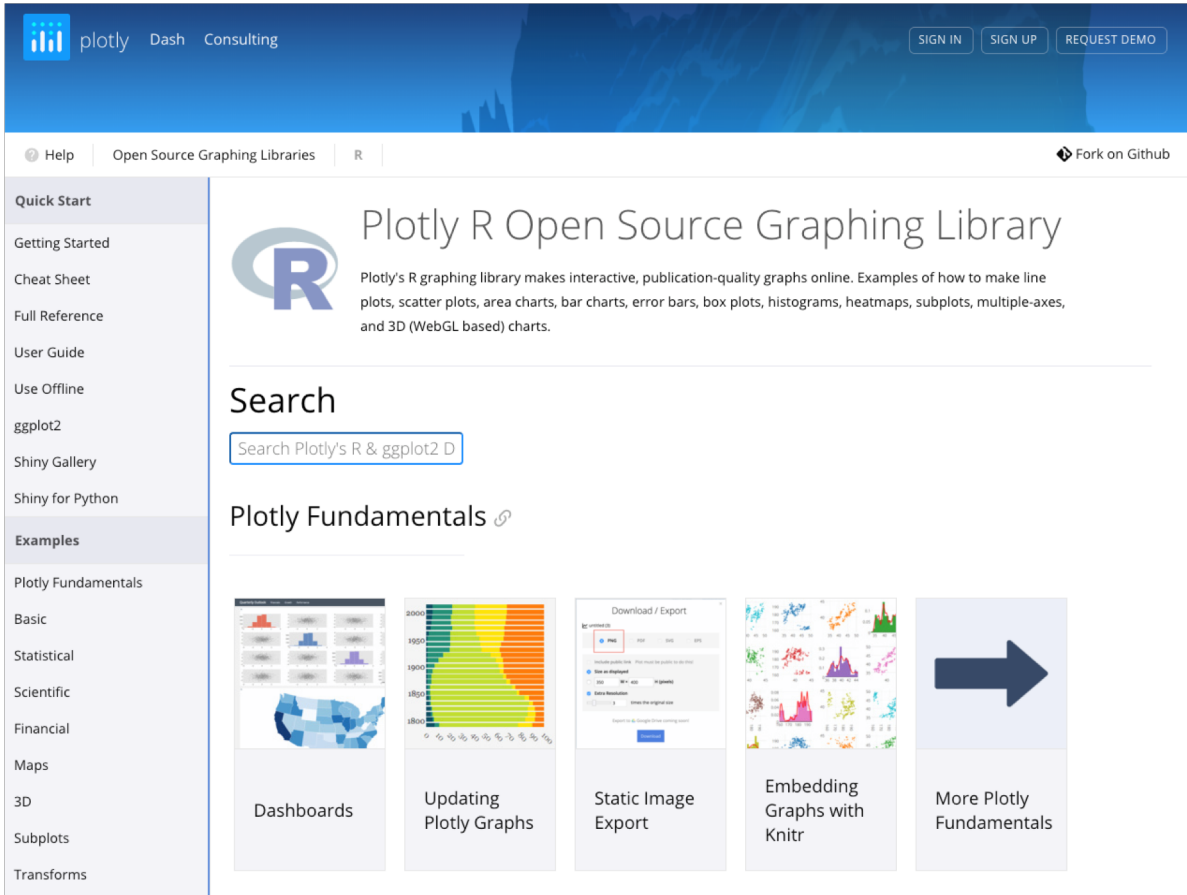
Infusing interactivity to static visualizations

- What visualizations have we learned to create so far?
  - Standard plots with ggplot
  - Geospatial visualization
  - Network visualization
- Let’s take a step back from user interface and start with creating an interactive version of these visualization.

plotly Gallery

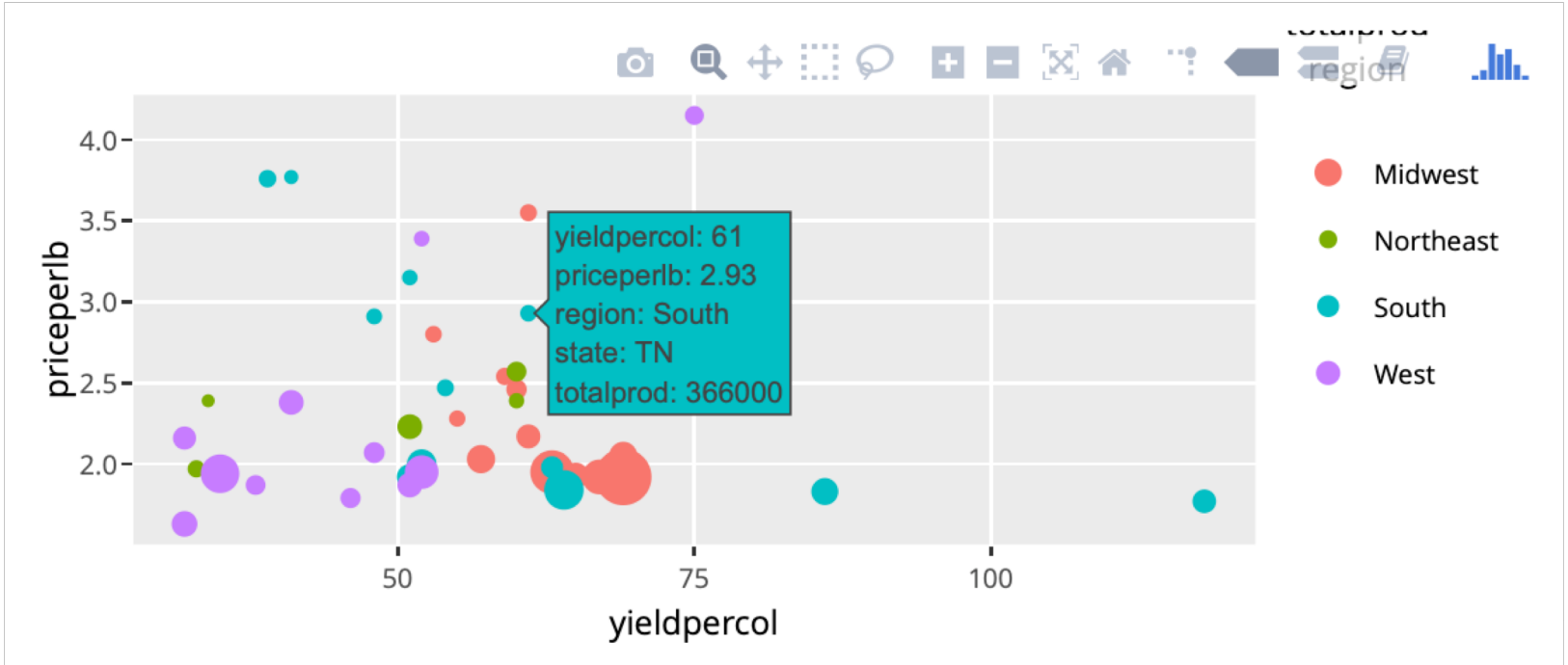
- https://plot.ly/r/





Using ggplotly to make standard ggplots interactive

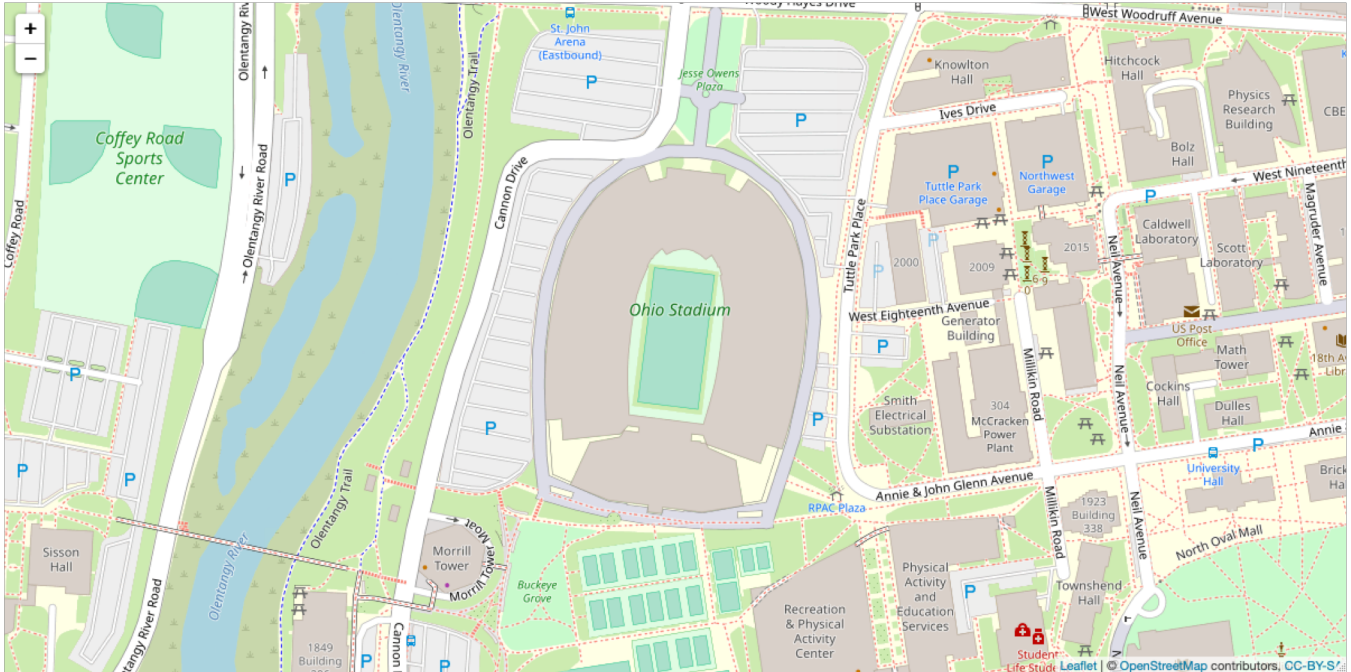
```
1  honey2012 <- honey %>% filter(year==2012)
2  p <- ggplot(honey2012, aes(yieldpercol, priceperlb, color=region,
   label=state, size=totalprod)) + geom_point()
3  ggplotly(p)
```



Creating interactive geospatial visualization: leaflet

- Loading a basic interactive map

```
1  library(leaflet)
2  leaflet() %>% setView(lng=-83.01973, lat=40.00166, zoom=17) %>% addTiles()
```



Setting up the data for interactive choropleth map

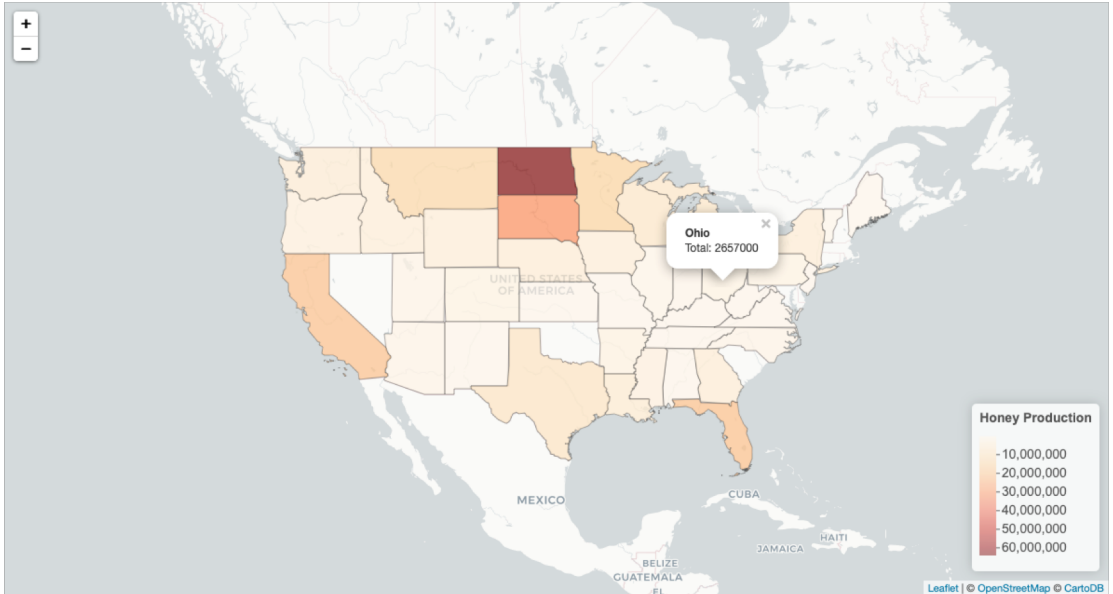
```
1  library(tigris)
2  states <- states(cb=T)
3  states_honey2012 <- geo_join(states, honey2012, "STUSPS", "state")
4  pal <- colorNumeric("OrRd", domain=states_honey2012$prodvalue)
5  states_honey2012 <- subset(states_honey2012, !is.na(prodvalue))
6  popups <- sprintf("<b>%s</b><br>Total: %d", states_honey2012$name,
   states_honey2012$prodvalue)
```

```
states                                  Large SpatialPolygonsDataFrame (56 elements, 6.8 Mb)
  ..@ data :'data.frame': 56 obs. of 9 variables:
  .. ..$ STATEFP : chr [1:56] "01" "02" "04" "05" ...
  .. ..$ STATENS : chr [1:56] "01779775" "01785533" "01779777" "00068085" ...
  .. ..$ AFFGEOID: chr [1:56] "0400000US01" "0400000US02" "0400000US04" "0400000US05" ...
  .. ..$ GEOID : chr [1:56] "01" "02" "04" "05" ...
  .. ..$ STUSPS : chr [1:56] "AL" "AK" "AZ" "AR" ...
  .. ..$ NAME : chr [1:56] "Alabama" "Alaska" "Arizona" "Arkansas" ...
  .. ..$ LSAD : chr [1:56] "00" "00" "00" "00" ...
  .. ..$ ALAND : chr [1:56] "131173688951" "1477946266785" "294198560125" "134771517596" ...
  .. ..$ AWATER : chr [1:56] "4593686489" "245390495931" "1027346486" "2960191698" ...
  ..@ polygons :List of 56
  .. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
  .. .. .. ..@ Polygons :List of 7
  .. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
  .. .. .. .. .. ..@ labpt : num [1:2] -88 30.5
  .. .. .. .. .. ..@ area : num 0.000532
  .. .. .. .. .. ..@ hole : logi FALSE
  .. .. .. .. .. ..@ ringDir: int 1
  .. .. .. .. .. ..@ coords : num [1:15, 1:2] -88.1 -88.1 -88 -88 -88 ...
  .. .. .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
  .. .. .. .. .. ..@ labpt : num [1:2] -88.2 30.3
  .. .. .. .. .. ..@ area : num 6.21e-06
```

```
states_honey2012                        Large SpatialPolygonsDataFrame (40 elements, 3.4 Mb)
  ..@ data :'data.frame': 40 obs. of 20 variables:
  .. ..$ STATEFP : chr [1:40] "01" "04" "05" "06" ...
  .. ..$ STATENS : chr [1:40] "01779775" "01779777" "00068085" "01779778" ...
  .. ..$ AFFGEOID : chr [1:40] "0400000US01" "0400000US04" "0400000US05" "0400000US06" ...
  .. ..$ GEOID : chr [1:40] "01" "04" "05" "06" ...
  .. ..$ STUSPS : chr [1:40] "AL" "AZ" "AR" "CA" ...
  .. ..$ NAME : chr [1:40] "Alabama" "Arizona" "Arkansas" "California" ...
  .. ..$ LSAD : chr [1:40] "00" "00" "00" "00" ...
  .. ..$ ALAND : chr [1:40] "131173688951" "294198560125" "134771517596" "403501101370" ...
  .. ..$ AWATER : chr [1:40] "4593686489" "1027346486" "2960191698" "20466718403" ...
  .. ..$ state : chr [1:40] "AL" "AZ" "AR" "CA" ...
  .. ..$ numcol : num [1:40] 8000 22000 25000 330000 25000 59000 10000 92000 7000 8000 ...
  .. ..$ yieldpercol: num [1:40] 54 46 63 35 48 51 75 32 61 59 ...
  .. ..$ totalprod : num [1:40] 432000 1012000 1575000 11550000 1200000 ...
  .. ..$ stocks : num [1:40] 65000 253000 189000 3119000 468000 ...
  .. ..$ priceperlb : num [1:40] 2.47 1.79 1.98 1.94 2.07 1.92 4.15 1.63 3.55 2.54 ...
  .. ..$ prodvalue : num [1:40] 1067000 1811000 3119000 22407000 2484000 ...
  .. ..$ year : int [1:40] 2012 2012 2012 2012 2012 2012 2012 2012 2012 2012 ...
  .. ..$ name : chr [1:40] "Alabama" "Arizona" "Arkansas" "California" ...
  .. ..$ region : chr [1:40] "South" "West" "South" "West" ...
  .. ..$ division : chr [1:40] "East South Central" "Mountain" "West South Central" "Pacific" ...
```

Creating an interactive choropleth map

```
1  leaflet(data=states_honey2012) %>% addProviderTiles("CartoDB.Positron") %>%
2    setView(-98.483330, 38.712046, zoom = 4) %>%
3    addPolygons(fillColor=~pal(prodvalue),
4      fillOpacity=0.7, weight=1, color=gray(.3), popup=~popups) %>%
5    addLegend(pal=pal, values=prodvalue, position="bottomright", title="Honey
   Production")
```

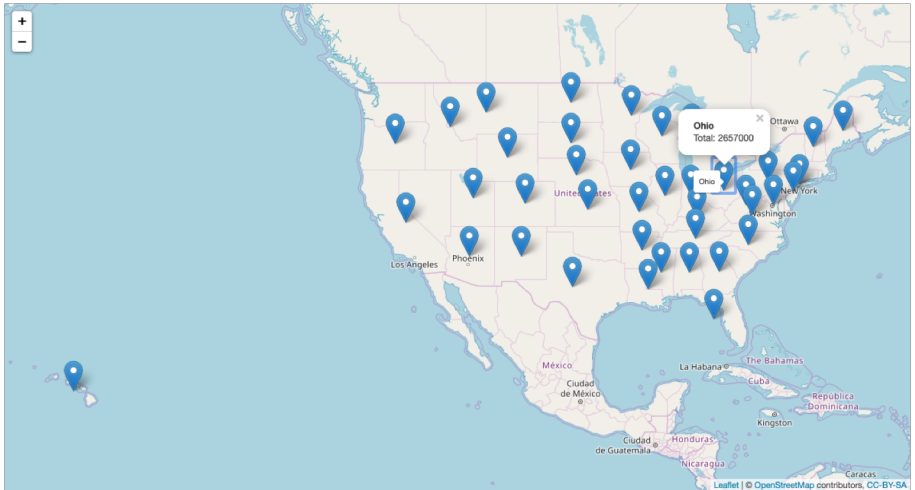


Code example adapted from:
http://learn.r-journalism.com/en/mapping/census_maps/census-maps/

Interactive map with overlays

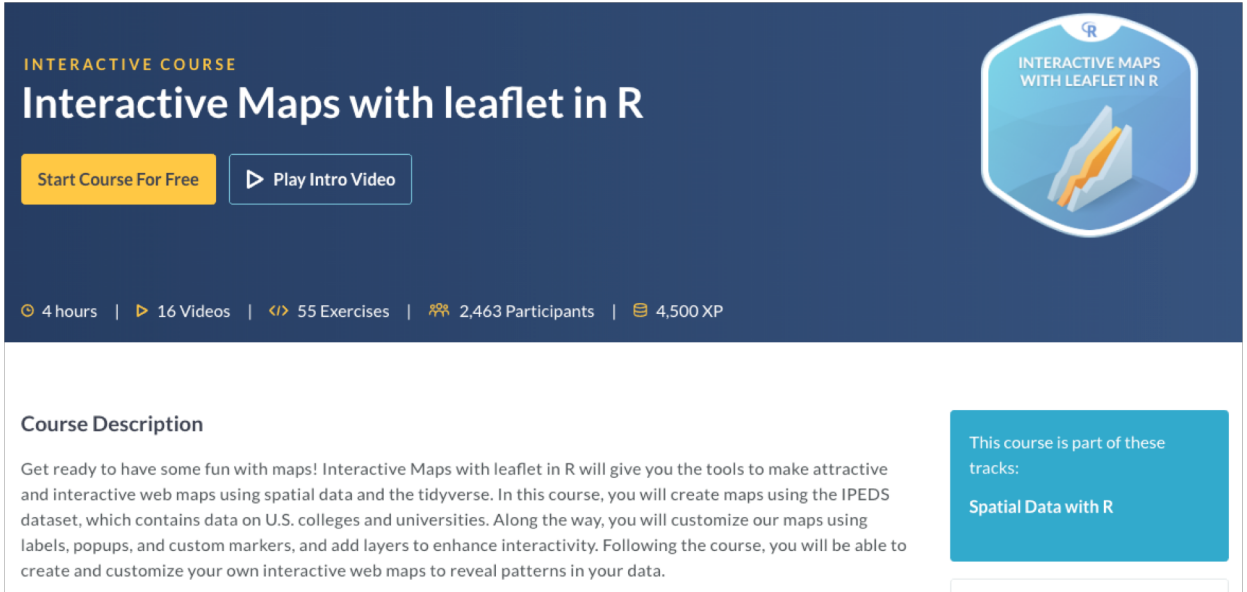
- Latitudes and longitudes for U.S. state centers are from: https://www.kaggle.com/washimahmed/usa-latlong-for-state-abbreviations

```
stll <- read_csv("data/statelatlong.csv") %>% rename(state=State)
honey2012 %>% left_join(stll) %>% mutate(plab=sprintf("<b>%s</b><br>Total: %d", name, prodvalue)) %>%
  leaflet() %>% addTiles() %>% addMarkers(~Longitude, ~Latitude, popup=~plab, label=~name)
```



DataCamp on using leaflet

- Complete the following Data Camp courses on interactive geospatial visualization.
  - [4] Interactive Maps with leaflet in R
- Leaflet's step-by-step introduction looks great, too.
  - https://rstudio.github.io/leaflet/

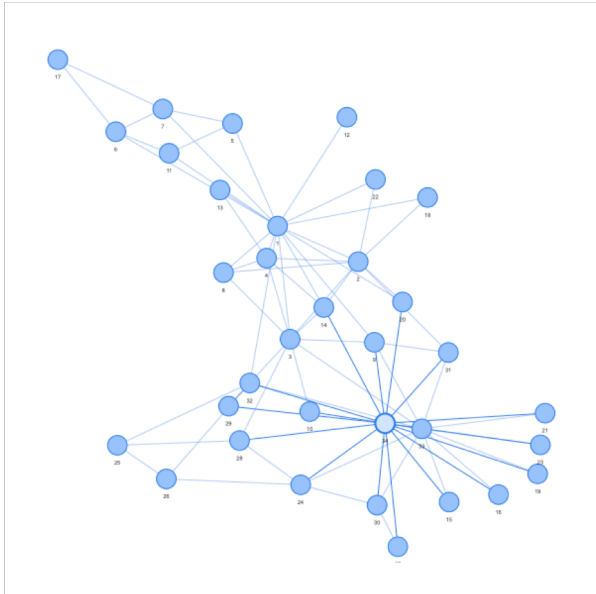


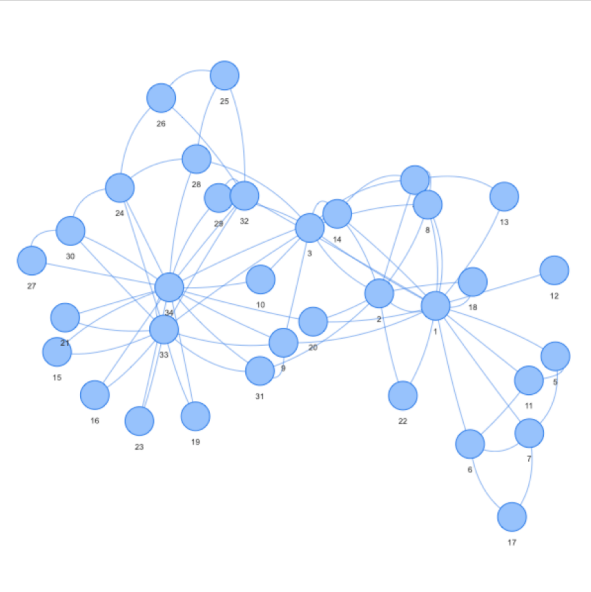
Interactive network visualization

- Creating interactive network visualization in R is not very well standardized yet.
- The library called D3.js implemented in JavaScript is probably most widely used framework for interactive network visualization, but it requires of course JavaScript programming and web development skills.
- There are a few viable R packages that aim to bring interactivity to network visualization in R. (visNetwork and networkD3)
- In my view, visNetwork package is currently the most usable one.
- Reading
  - Section 6.2 and 6.3 from Ognyanova, K. (2018) *Network visualization with R.* available at www.kateto.net/network-visualization

Interactive visualization of igraph object with visNetwork

```
1  library(igraph)
2  library(visNetwork)
3  g <- make_graph("Zachary")
4  visIgraph(g, physics=T, smooth=T)
5  g2 <- g %>% toVisNetworkData()
6  visNetwork(g2$nodes, g2$edges)
```





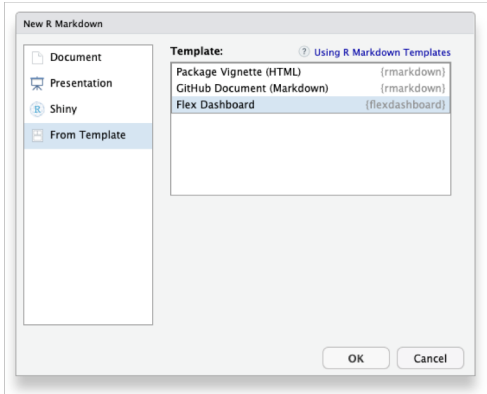
Dashboard Design

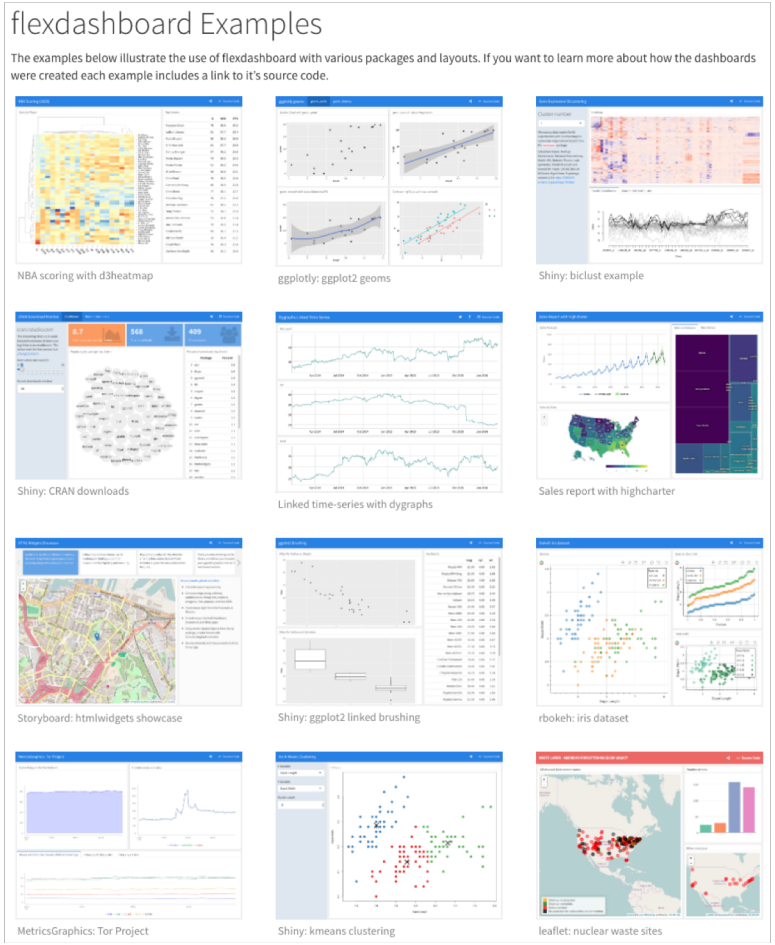
Motivation for designing a dashboard

- So far, we have learned several different visualization.
- Suppose a situation where you (or someone) need to regularly check the data being updated regularly. And you check the data with a preset collection of visualizations.
- It would be quite inefficient to generate individual visualizations every time you have to check the data. It will be tolling if the interval is like every day.
- Dashboard (imagine the dashboard in a car) tries to solve this problem with:
  - having a composition of preset visualizations [must have],
  - optionally providing ways to drill down into the data as needed [good to have].
- To my survey, R has three ways to generate a dashboard.
  - flexdashboard, shinydashboard, Shiny
  - https://stackoverflow.com/questions/37992147/r-shiny-which-hammer-straight-shiny-flexdashboard-or-shinydashboard

flexdashboard

- flexdashboard is a client-side running dashboard.
- It is basically an R Markdown document with special flexdashboard template.
- The official walkthrough page is great.
  - https://rmarkdown.rstudio.com/flexdashboard/using.html
- Take a look at the gallery to see what you can do.
  - https://rmarkdown.rstudio.com/flexdashboard/examples.html





Layout of a default flexdashboard

````
---
title: "Honey Production"
output:
  flexdashboard::flex_dashboard:
    orientation: columns
    vertical_layout: fill
---

```{r setup, include=FALSE}
library(flexdashboard)
```

Column {data-width=650}
-----------------------------------------------------------------------

### Chart A

```{r}

```

Column {data-width=350}
-----------------------------------------------------------------------

### Chart B

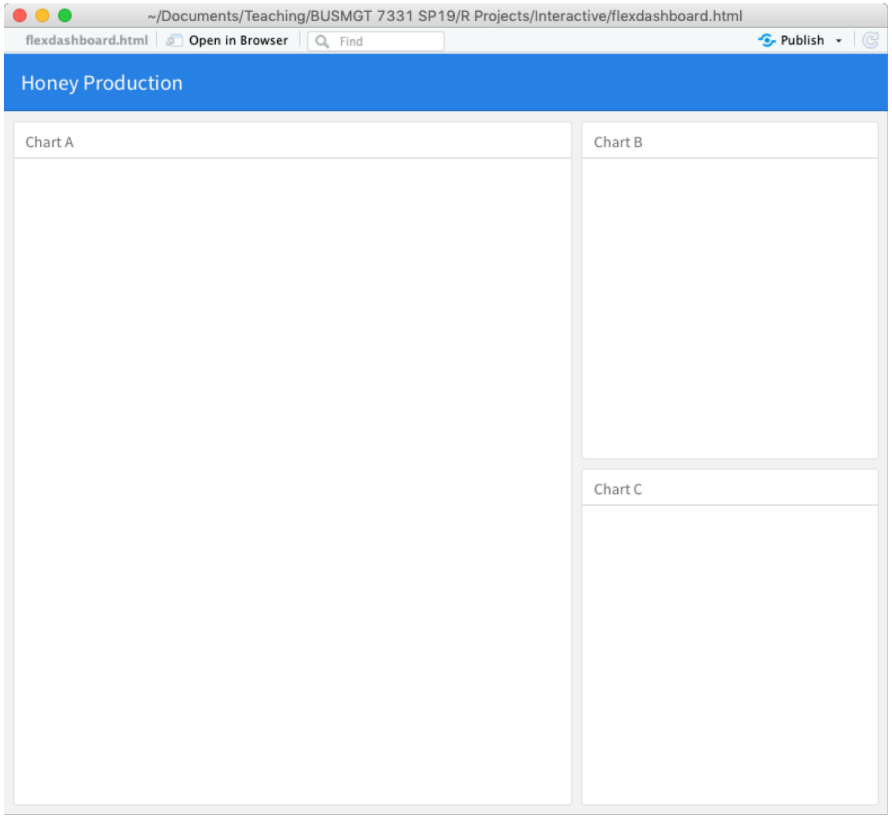
```{r}

```

### Chart C

```{r}

```
````



Fleshing out the wireframe

```
p <- ggplot(honey2012, aes(numcol, priceperlb, label=state)) + geom_point()
ggplotly(p)
p <- ggplot(honey2012, aes(prodvalue)) + geom_histogram()
ggplotly(p)
p <- ggplot(honey2012, aes(region, stocks, fill=state)) + geom_col()
ggplotly(p)
```

Interactive data exploration with shinydashboard

```
---
title: "shinydashboard Demo"
output:
  flexdashboard::flex_dashboard:
    orientation: columns
    vertical_layout: fill
runtime: shiny
---

```{r setup, include=FALSE}
library(tidyverse)
library(plotly)
honey <- read_csv("data/honeyproduction.csv", col_types=cols(year=col_integer()))
```

Inputs {.sidebar}

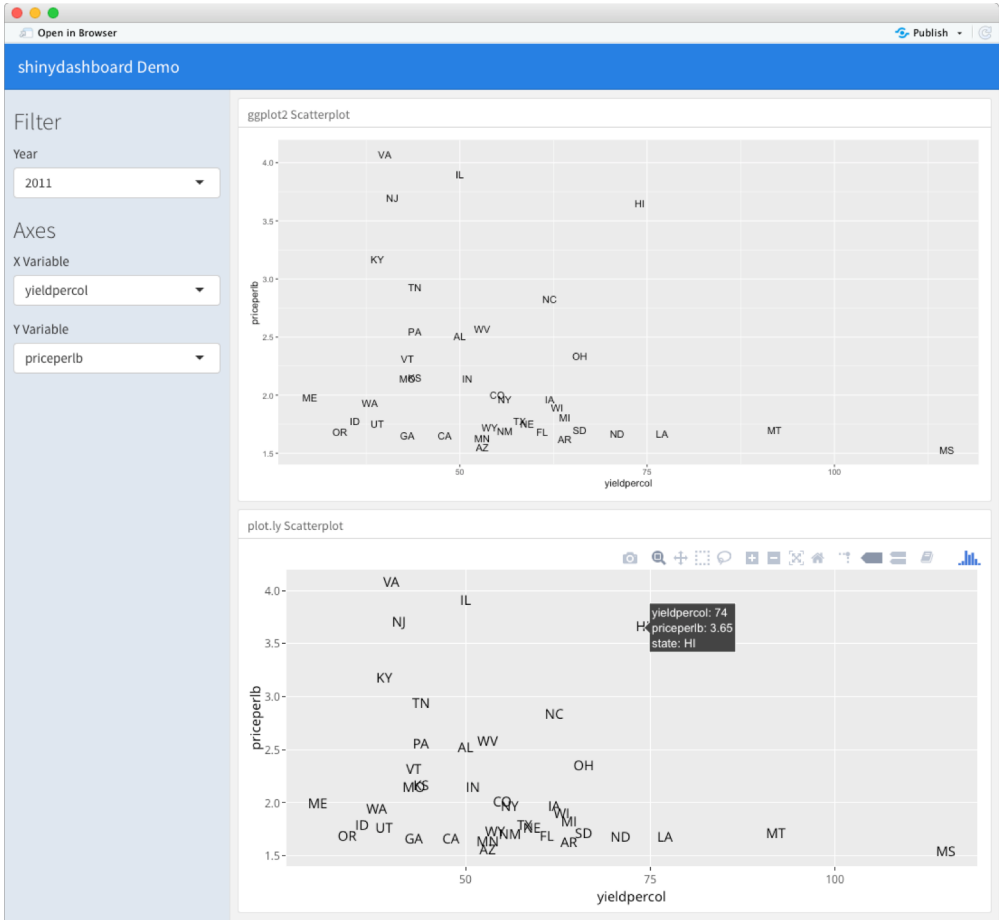
Filter
```{r}
selectInput("year", "Year", sort(unique(honey$year)))
```

Axes
```{r}
selectInput("xcol", "X Variable", names(honey)[2:7])
selectInput("ycol", "Y Variable", names(honey)[2:7])
```

Column
----

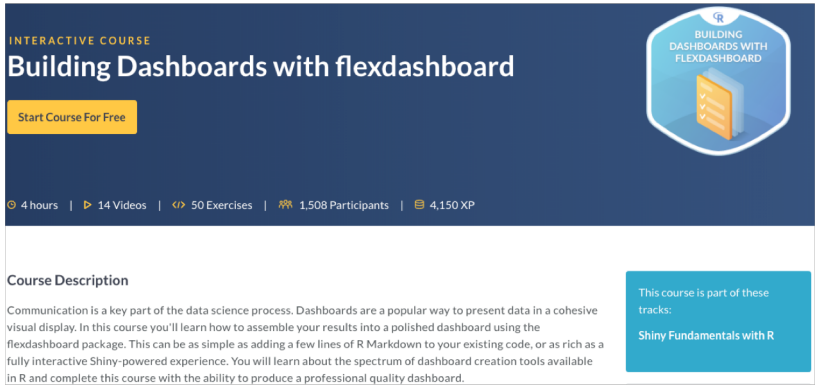
ggplot2 Scatterplot
```{r}
selectedData <- reactive({honey %>% filter(year==input$year)})
renderPlot({
  ggplot(selectedData(), aes(label=state)) +
    geom_text(aes_string(x=input$xcol, y=input$ycol))
})
```

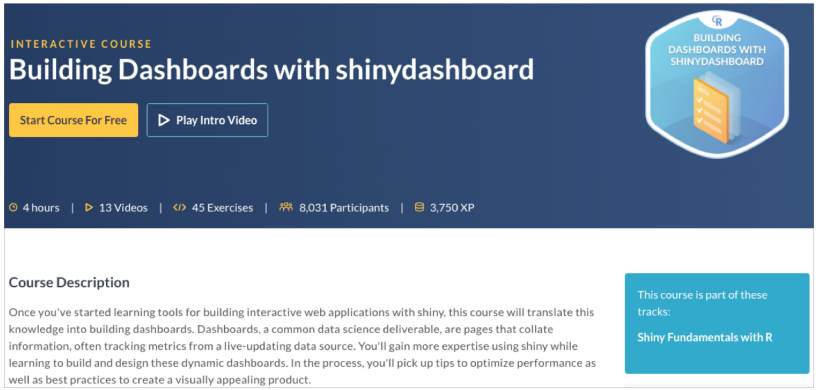
plot.ly Scatterplot
```{r}
renderPlotly({
  p <- ggplot(selectedData(), aes(label=state)) +
    geom_text(aes_string(x=input$xcol, y=input$ycol))
  ggplotly(p)
})
```
```



Data Camp courses on dashboard design

- Complete the following two Data Camp courses on dashboard design.
  - [3] Building Dashboards with shinydashboard
  - [4] Building Dashboards with flexdashboard





Weekly Recap

Things we covered this week

- Animated Visualization
  - Why animation?
  - Added time dimension
  - Output formats: gif or mp4
- Interactive Visualization
  - The Visual-Information Seeking Mantra
  - Interaction techniques
  - User interface basics
- Adding Interactivity
  - plotly
  - leaflet
  - visNetwork
- Dashboard Design
  - flexdashboard
  - shinydashboard

Things to do this week

- 4 Quizzes (Due: Thursday, February 14, 11:59pm)
  - Week 6.1. Animated Visualization
  - Week 6.2. Interactive Visualization
  - Week 6.3. Adding Interactivity
  - Week 6.4. Dashboard Design
- 1 Weekly Problem (Due: Friday, February 15, 11:59pm)
- 3 DataCamp Courses (Due: Friday, February 15, 11:59pm)
  - Spatial Data with R
    - [4] Interactive Maps with leaflet in R
  - Shiny Fundamentals with R
    - [3] Building Dashboards with shinydashboard
    - [4] Building Dashboards with flexdashboard
- 1 In-class Activity (Due: Saturday, February 16, 11:59pm)