



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>

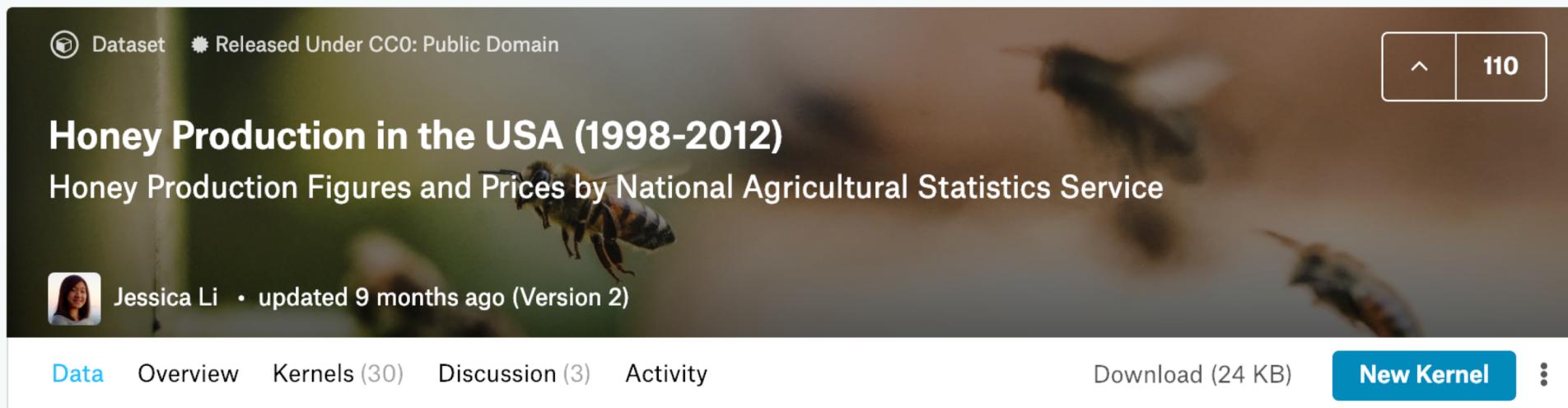
Dataset Released Under CC0: Public Domain 110

## Honey Production in the USA (1998-2012)

Honey Production Figures and Prices by National Agricultural Statistics Service

Jessica Li • updated 9 months ago (Version 2)

Data Overview Kernels (30) Discussion (3) Activity Download (24 KB) New Kernel



Data (24 KB)



Data Sources

About this file

Columns

Title, State, Year, Production, Price

# Packages for this week

---

```
1 install.packages("ganimate")
2 install.packages("gapminder")
3 install.packages("gifski")
4 install.packages("plotly")
5 install.packages("leaflet")
6 install.packages("tigris")
7 install.packages("visNetwork")
8 install.packages("flexdashboard")
9 install.packages("shinydashboard")
10 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.

The screenshot shows a TED talk video player. At the top right are social sharing icons for Share, Add to list, Like, and Rate. Below the video frame, the title 'Hans Rosling | TED2006' and the subtitle 'The best stats you've ever seen' are displayed. The video frame shows Hans Rosling speaking on stage with a large screen behind him displaying colorful data visualizations. A play button icon is in the center of the video frame. Below the video are three buttons: 'Details' (underlined), 'Transcript' (48 languages), and 'Comments (607)'. A progress bar shows the video is at 19:47. To the right of the video frame are the views count (13,098,381), the date (TED2006 | February 2006), and related tags (Africa, Asia, Google). At the bottom left is a small portrait of Hans Rosling.

- [https://www.ted.com/talks/hans\\_rosling\\_shows\\_the\\_best\\_stats\\_you\\_ve\\_ever\\_seen](https://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen)  
or  
<https://youtu.be/hVimVzgtD6w>

---

## ABOUT THE SPEAKER



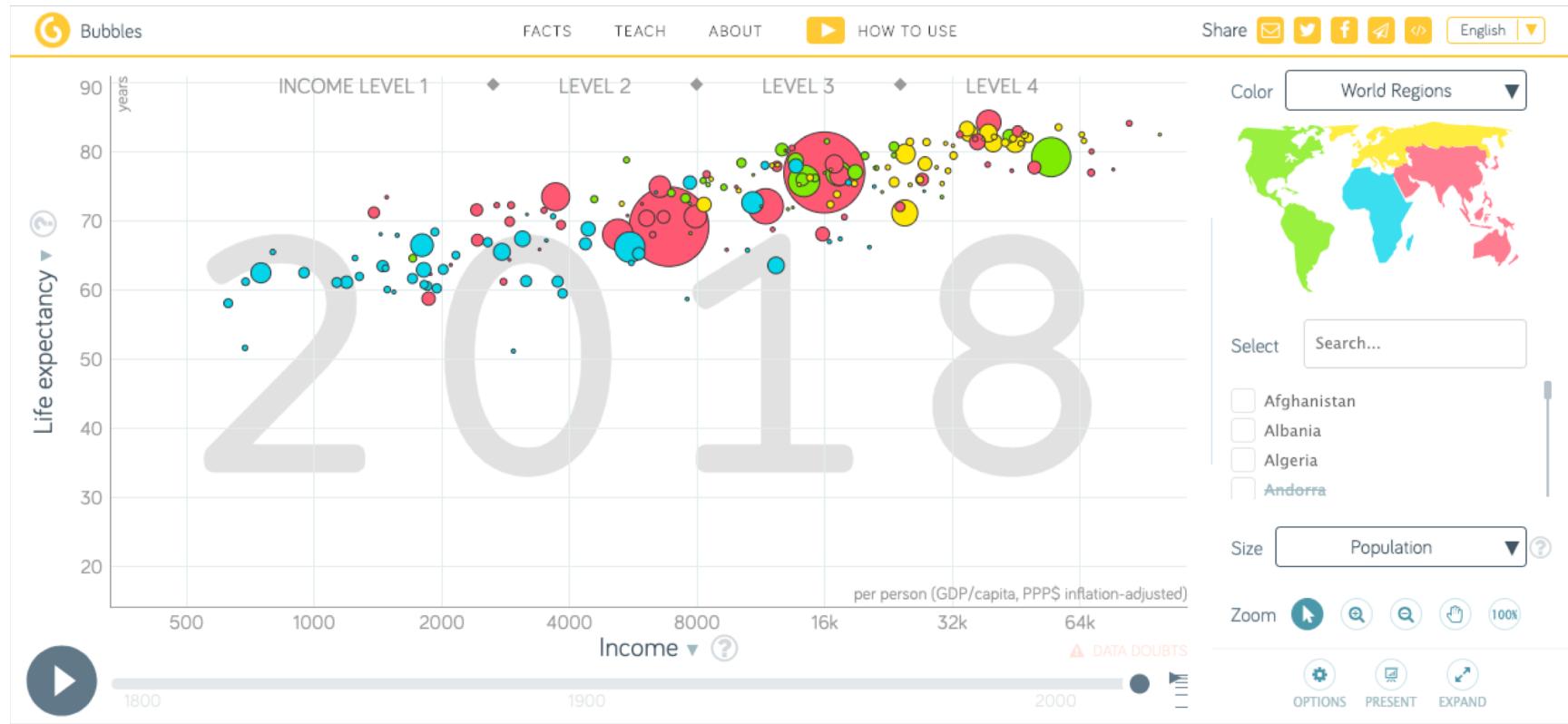
**Hans Rosling** · Global health expert; data visionary

In Hans Rosling's hands, data sings. Global trends in health and economics come to vivid life. And the big picture of global development—with some surprisingly good news—snaps into sharp focus.

\*\*\*

# 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.

# ganimate

- Package developed by Thomas Lin Pedersen
- Readings and Tutorials
  - Overview of ganimate: <https://ganimate.com/>
  - Getting started from ganimate: <https://ganimate.com/articles/ganimate.html>
  - Github front page of ganimate: <https://github.com/thomasp85/ganimate>
- 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",
6 "#FF7185")) +
7   scale_size(range=c(1,16)) +
8   scale_x_log10(breaks=c(500,1000,2000,4000,8000,16000,32000,64000)) +
9   scale_y_continuous(breaks=seq(20,90,10)) +
10  labs(x="Income", y="Life expectancy") +
11  theme(panel.grid.minor = element_blank())
```

# Does it look similar?

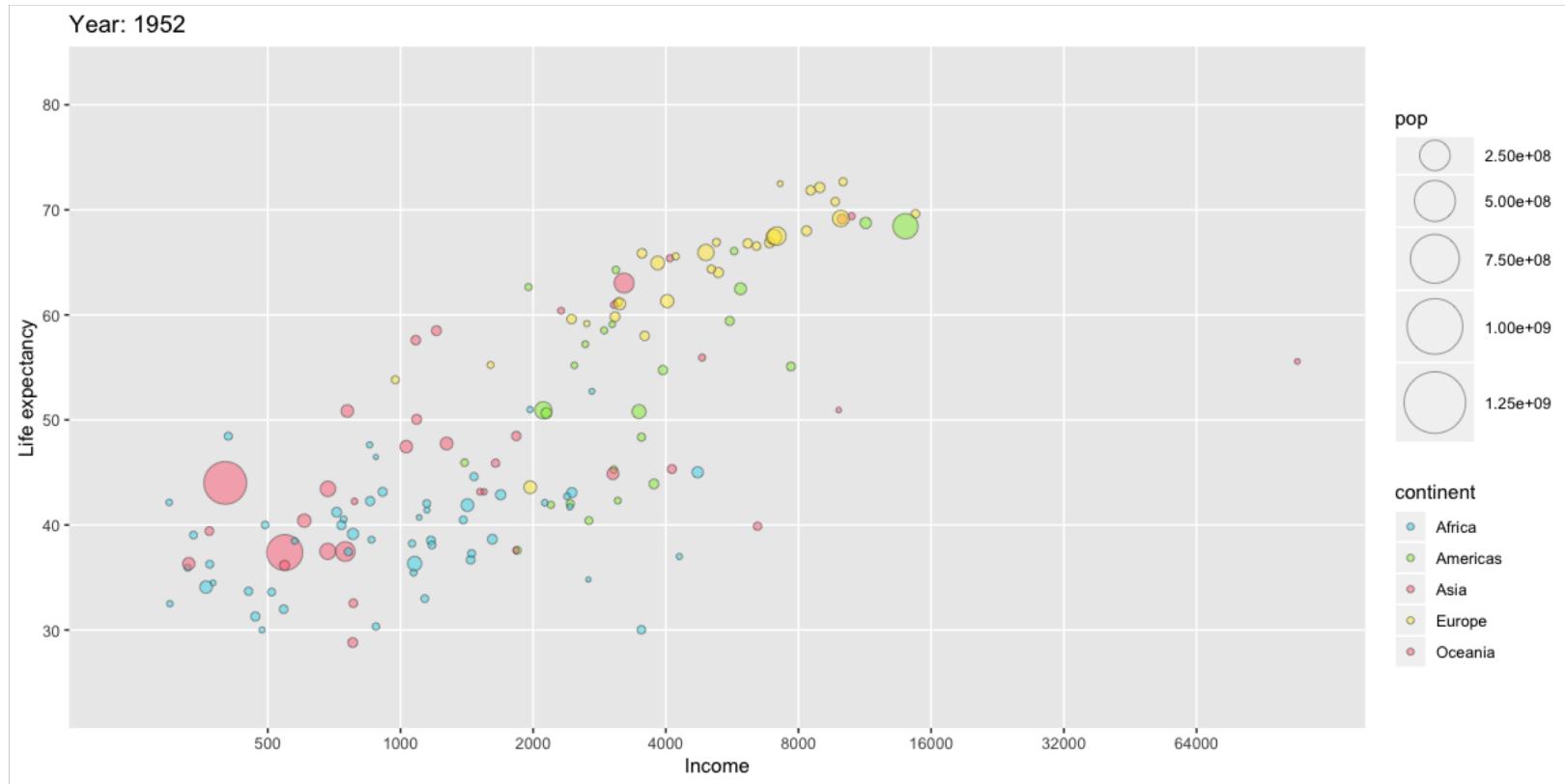


# Let's animate this.

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

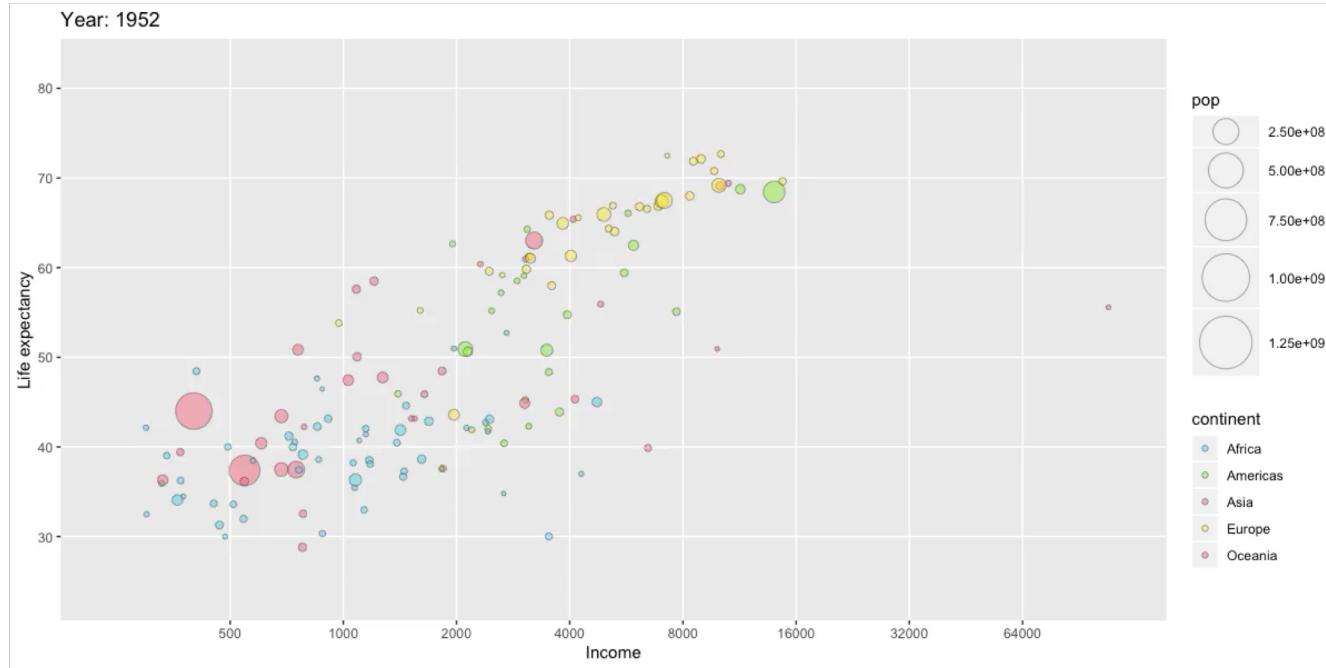
```
1 p <- ggplot(gapminder, aes(gdpPercap, lifeExp, size=pop, fill=continent)) +
2   geom_point(alpha=.5, shape=21, color=gray(0.3)) +
3   scale_fill_manual(values=c("#33D9EB", "#90ED37", "#FF7185", "#FFE938",
4 "#FF7185")) +
5   scale_size(range=c(1,16)) +
6   scale_x_log10(breaks=c(500,1000,2000,4000,8000,16000,32000,64000)) +
7   scale_y_continuous(breaks=seq(20,90,10)) +
8   labs(title="Year: {frame_time}", x="Income", y="Life expectancy") +
9   theme(panel.grid.minor = element_blank()) +
10  transition_time(year) + ease_aes("linear")
11
12 animate(p, width=1200, height=600, res=100)
13 anim_save("img/gapminder_animation.gif")
```

# Animation output: gif



# Using other renderers: mp4

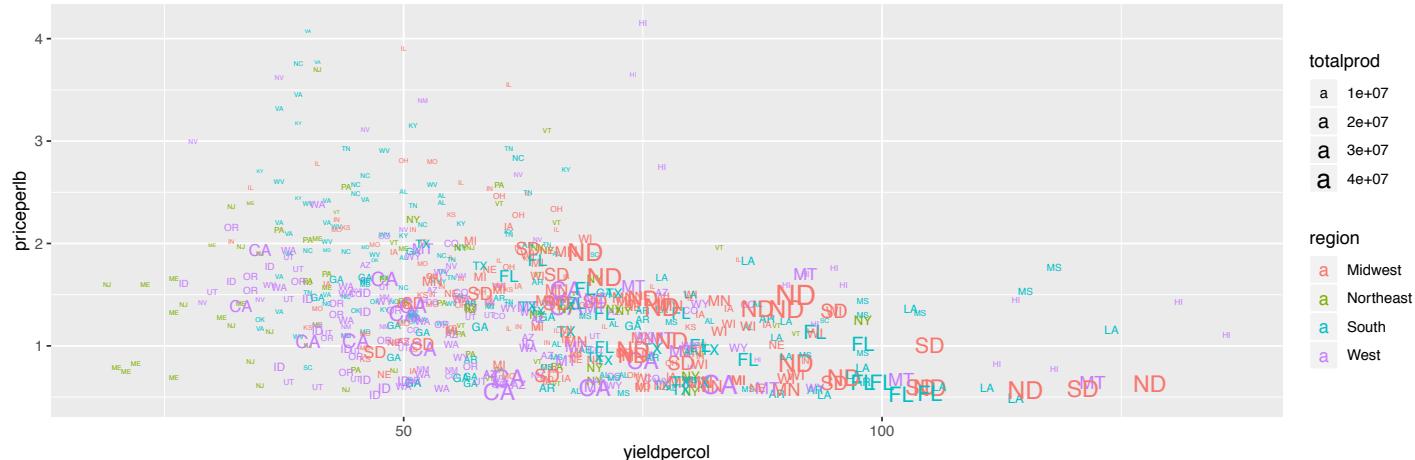
```
1  animate(p, width=1200, height=600, res=100, renderer=ffmpeg_renderer("mp4"))
2  anim_save("img/gapminder_animation.mp4")
```



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

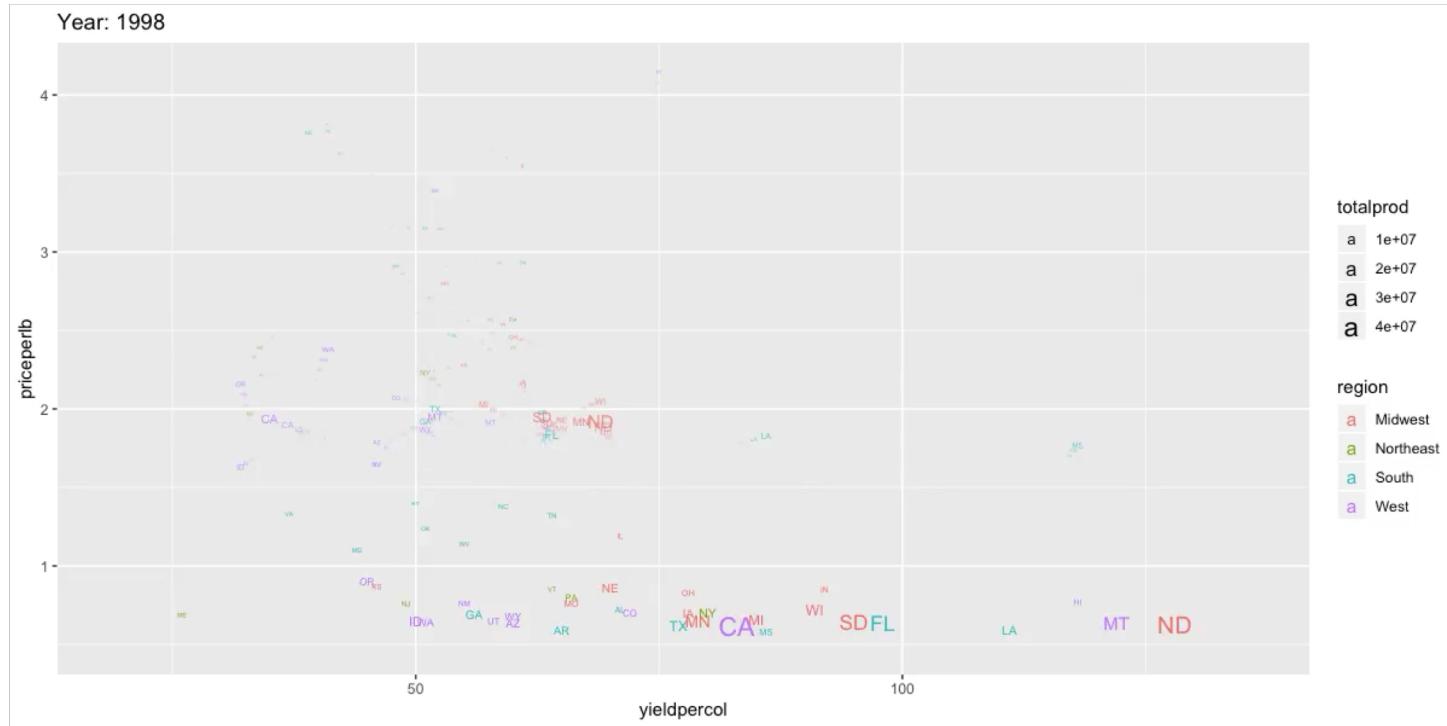
- State regions come from <https://github.com/cphalpert/census-regions>

```
1 regions <- read_csv("https://raw.githubusercontent.com/cphalpert/census-
2 regions/master/us%20census%20bureau%20regions%20and%20divisions.csv") %>%
3   rename(name=State, state='State Code', region=Region, division=Division)
4 honey <- read_csv("data/honeyproduction.csv",
5   col_types=cols(year=col_integer())) %>% left_join(regions)
6 ggplot(honey, aes(yieldpercol, priceperlb, color=region, label=state,
7   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-Alaska>
  - <https://github.com/thomasp85/gganimate/wiki/Tracking-of-hurricanes-and-typhoons>

# Interactive Visualization

# Motivation for interactivity in visualization

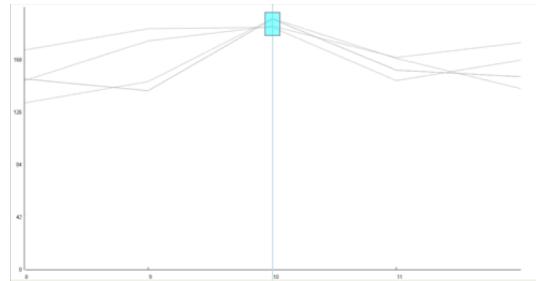
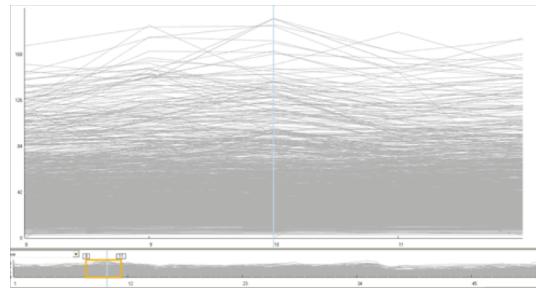
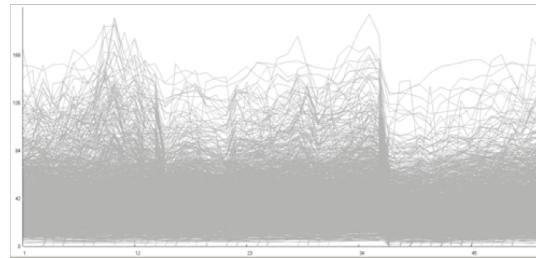
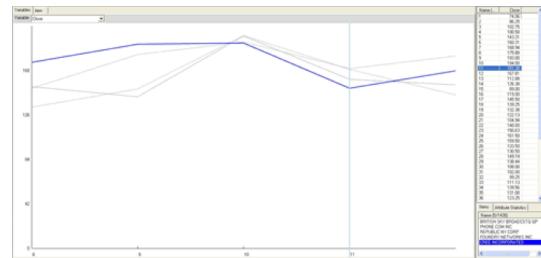
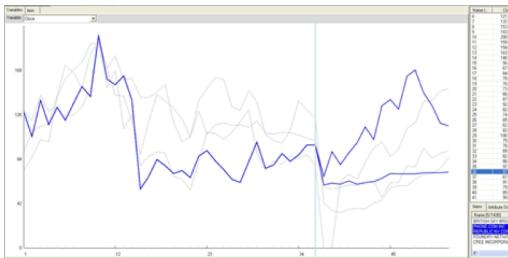
- Animation vs. interaction
  - Animation is a great way to exploit the unused “time” dimension in visualizing data.
  - However, animation may still be considered “static” in the sense that viewers can only see the data in a way that the creator wants them to see.
  - Interactivity turns those viewers 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 verbatim.  
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.



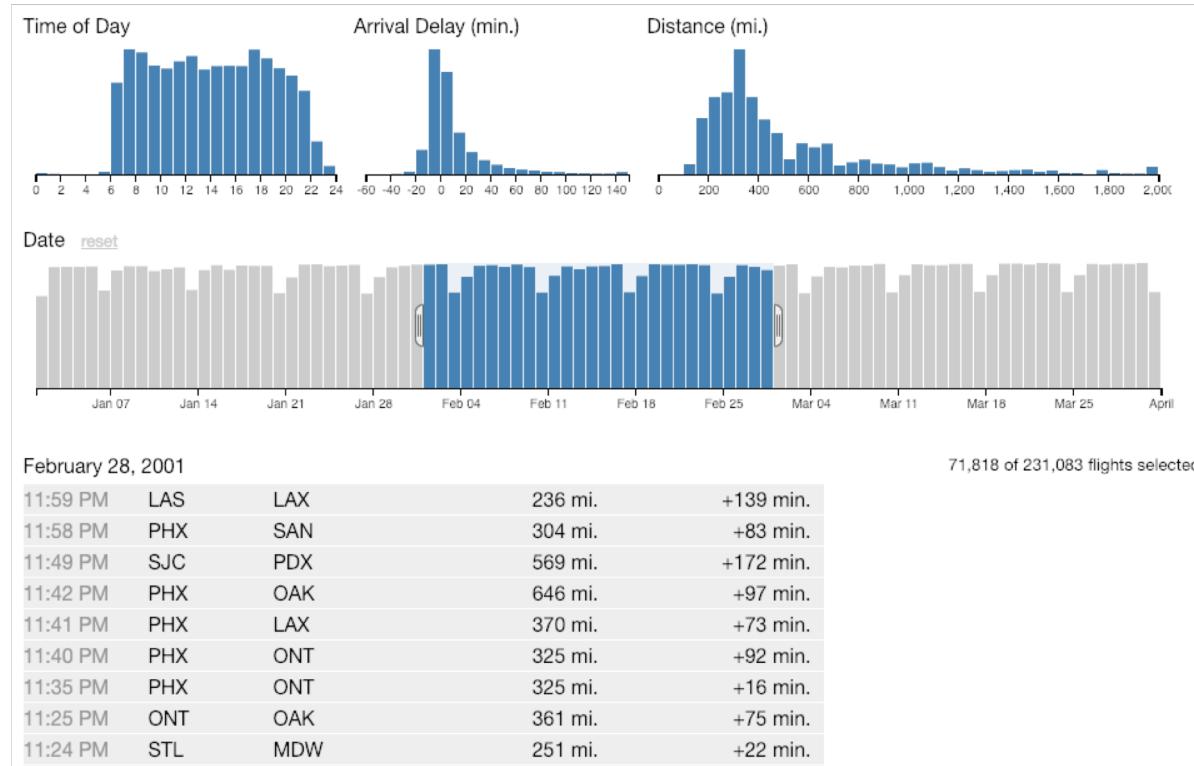
# Interaction techniques in information visualization

[https://infovis-wiki.net/wiki/Category:Interaction\\_Techniques](https://infovis-wiki.net/wiki/Category:Interaction_Techniques)

- (Coordinated) multiple views
  - [https://infovis-wiki.net/wiki/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/Dynamic_query)
  - <https://infovis-wiki.net/wiki/Filtering>
- Zoom
  - <https://infovis-wiki.net/wiki/Zoom>
  - [https://infovis-wiki.net/wiki/Semantic\\_Zoom](https://infovis-wiki.net/wiki/Semantic_Zoom)
- Details on demand
  - [https://infovis-wiki.net/wiki/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\\_Browsing](https://infovis-wiki.net/wiki/Linking_and_Browsing)
- Focus+context
  - <https://infovis-wiki.net/wiki/Focus-plus-Context>
  - [https://infovis-wiki.net/wiki/Fisheye\\_View](https://infovis-wiki.net/wiki/Fisheye_View)
  - [https://infovis-wiki.net/wiki/Magic\\_Lens](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>

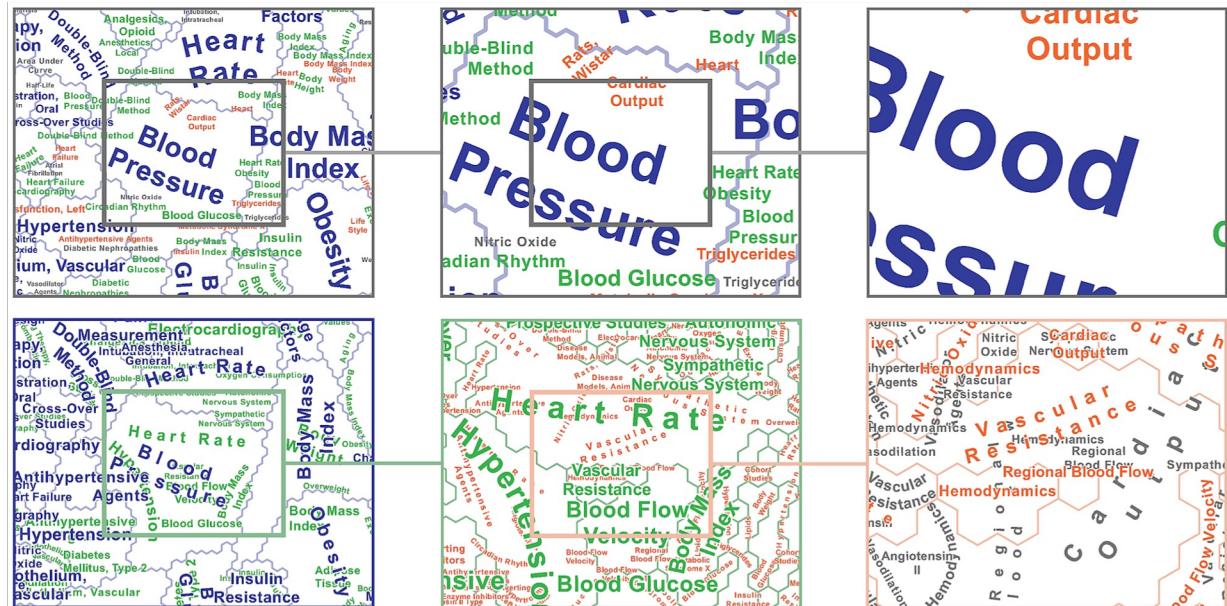
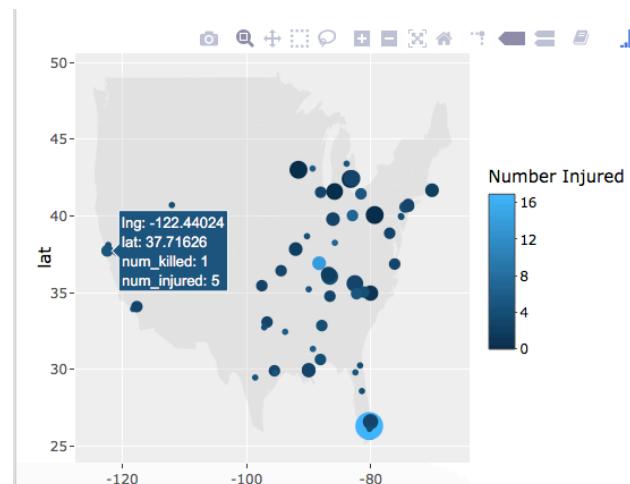
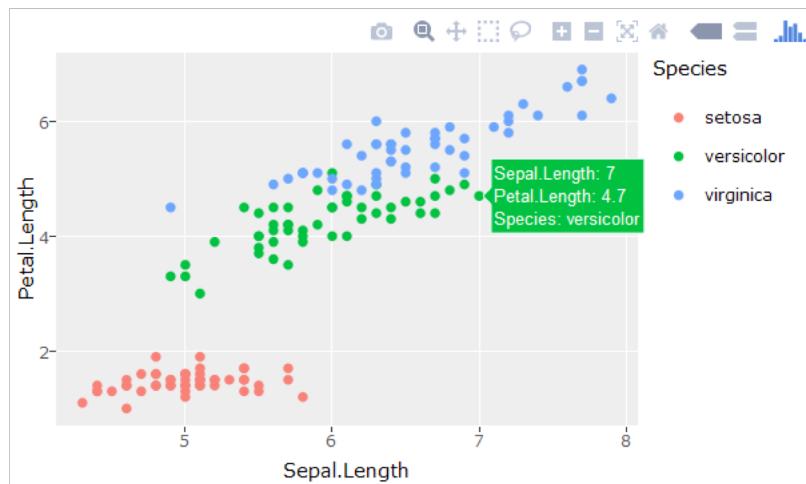


Figure from Skupin, A., R. Biberstine, J. and Börner, K. (2013) 'Geometric zooming versus semantic zooming.' PLOS ONE. doi: 10.1371/journal.pone.0058779.g008.

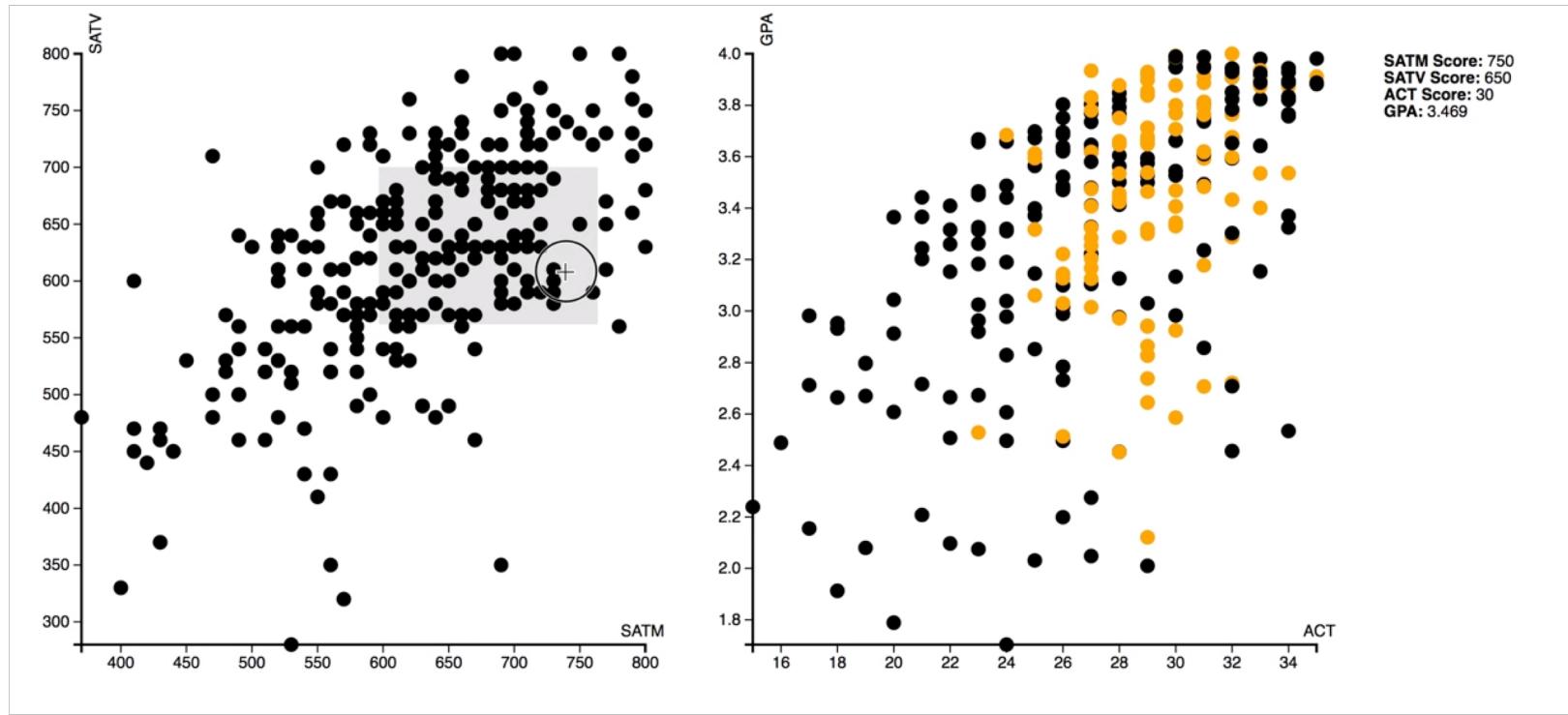
# Details-on-demand

- 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>



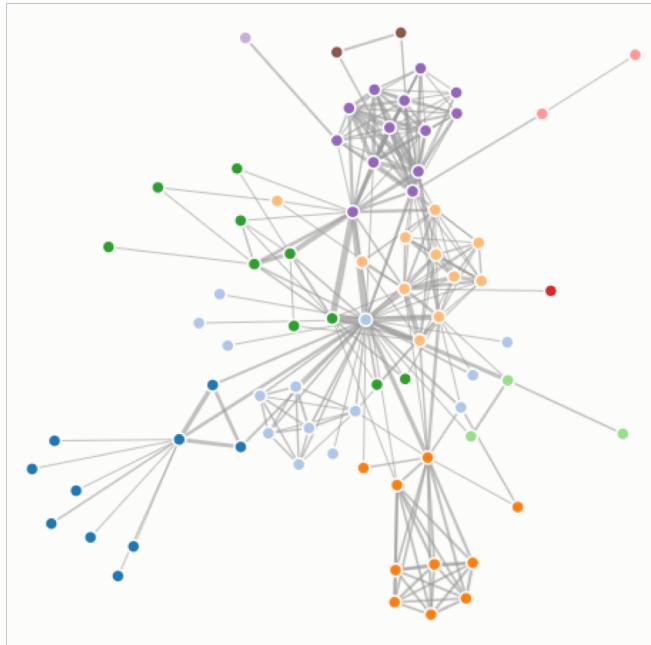
# Brushing and linking

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



# Focus+context technique: fisheye distortion

- <https://bostocks.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/>

# Bootstrap

Build responsive, mobile-first projects on the web with the world's most popular front-end component library.

Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build your entire app with our Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.

[Get started](#)[Download](#)

Currently v4.2.1



# 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/>

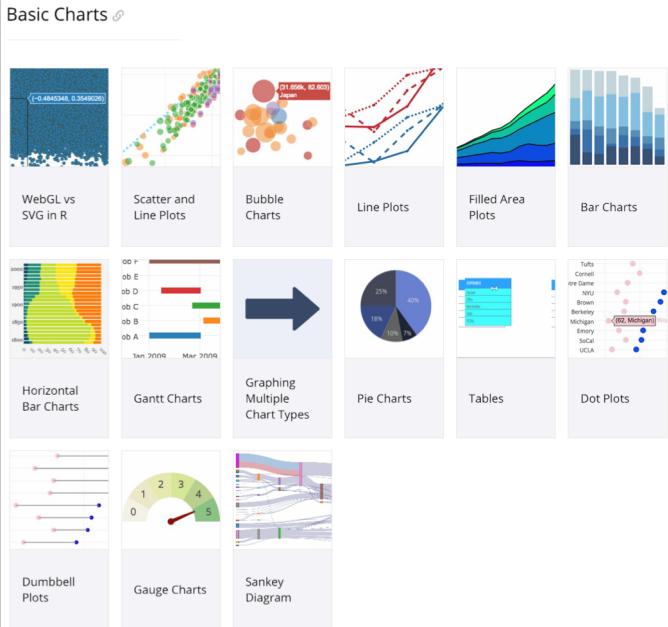
# Adding Interactivity

# 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/>

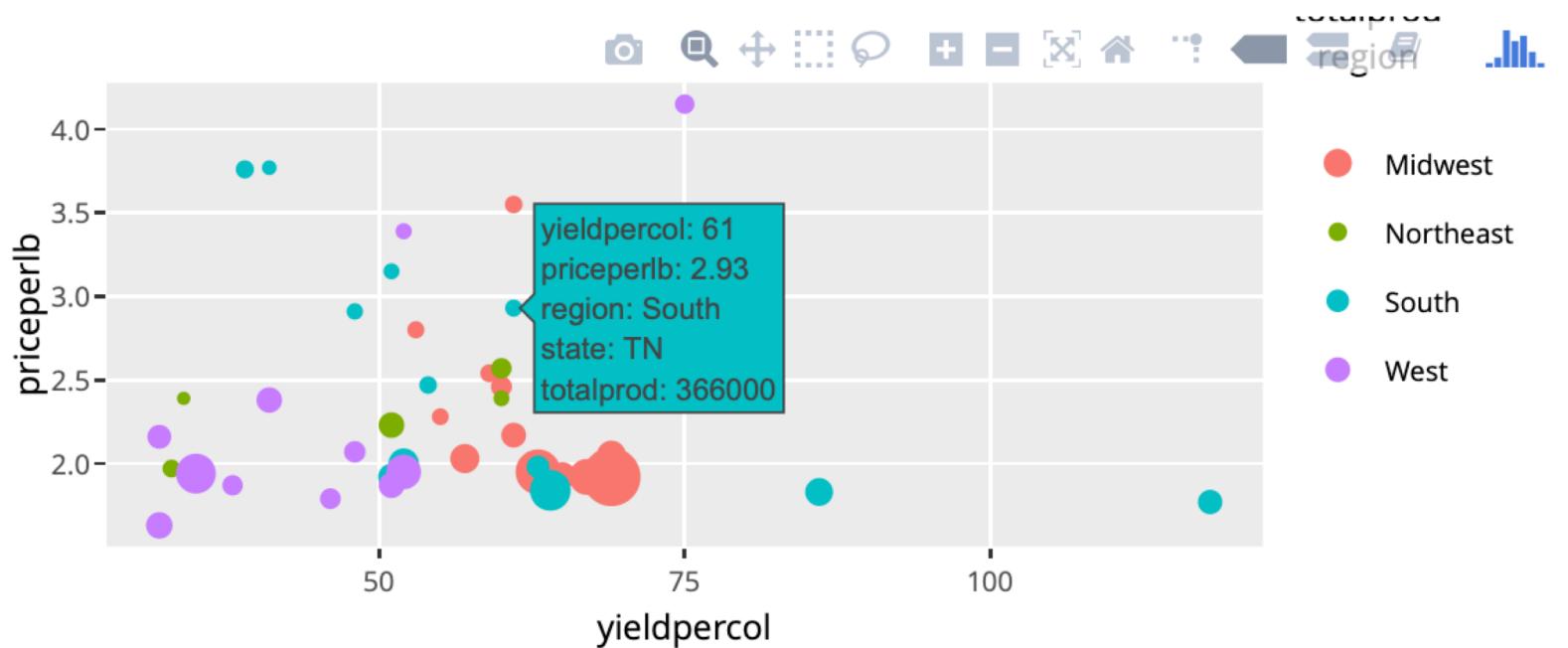


The screenshot shows the main page of the 'Plotly R Open Source Graphing Library'. It features a sidebar with navigation links like 'Help', 'Open Source Graphing Libraries', 'R', 'Quick Start' (selected), 'Getting Started', 'Cheat Sheet', 'Full Reference', 'User Guide', 'Use Offline', 'ggplot2', 'Shiny Gallery', 'Shiny for Python', 'Examples' (selected), 'Plotly Fundamentals' (selected), 'Basic', 'Statistical', 'Scientific', 'Financial', 'Maps', '3D', 'Subplots', and 'Transforms'. The main content area includes:

- A large 'R' logo.
- A text block: 'Plotly's R graphing library makes interactive, publication-quality graphs online. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, and 3D (WebGL based) charts.'
- A 'Search' input field.
- A section titled 'Plotly Fundamentals' with cards for 'Dashboards', 'Updating Plotly Graphs', 'Static Image Export', 'Embedding Graphs with Knitr', and 'More Plotly Fundamentals'.
- Links for 'SIGN IN', 'SIGN UP', and 'REQUEST DEMO'.
- A 'Fork on Github' button.

# Using ggplotly to make standard ggplots interactive

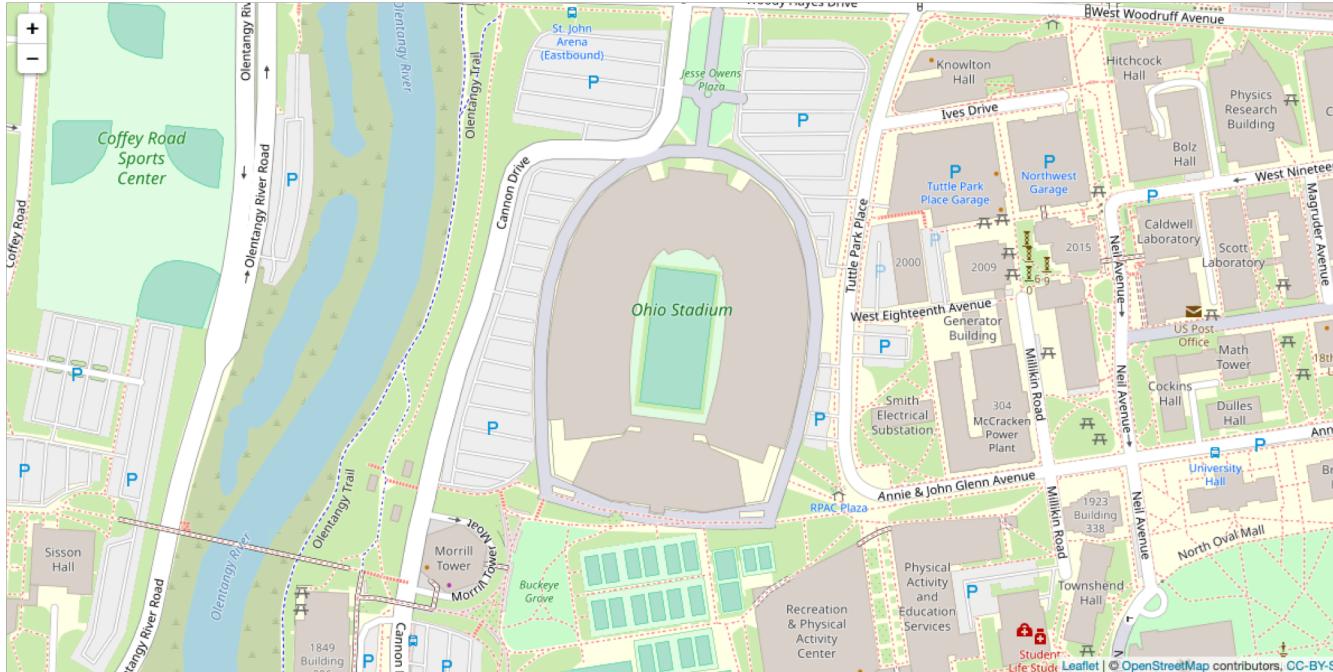
```
1 honey2012 <- honey %>% filter(year==2012)
2 p <- ggplot(honey2012, aes(yieldpercol, priceperlb, color=region,
3 label=state, size=totalprod)) + geom_point()
4 ggplotly(p)
```



# 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

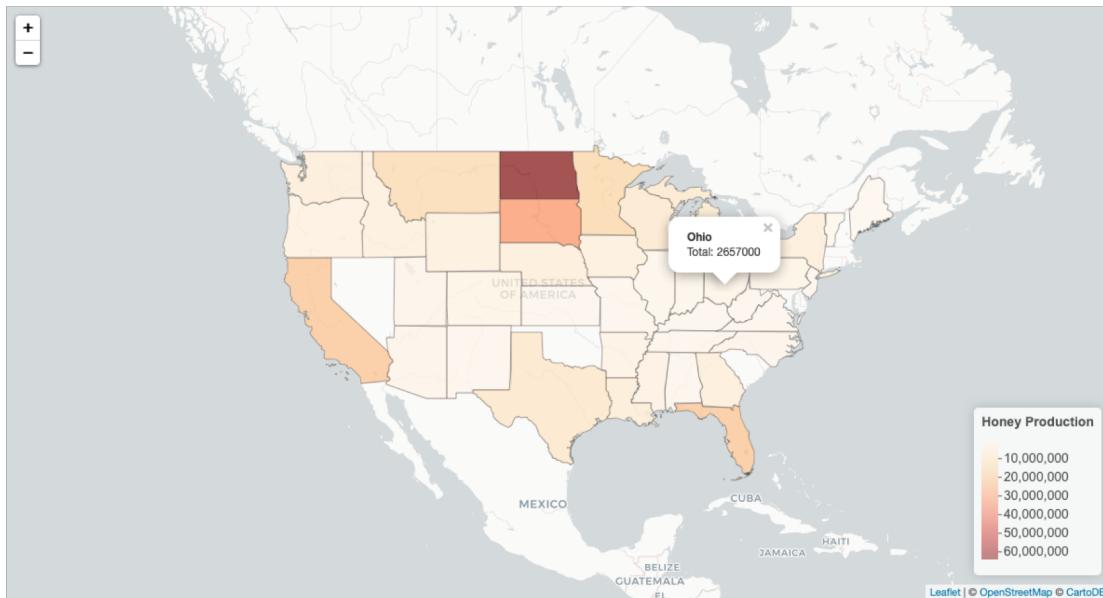
```
1 library(tigris)
2 states <- states(cb=T)
3 states_honey2012 <- geo_join(states, honey2012, "STUSPS", "state")
4 pal <- colorNumeric("OrRd", domain=states_honey2012$prodvalue)
5 states_honey2012 <- subset(states_honey2012, !is.na(prodvalue))
6 popups <- sprintf("<b>%s</b><br>Total: %d", states_honey2012$name,
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] "04000000US01" "04000000US02" "04000000US04" "04000000US05" ...	
... ..\$ GEOFID : 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] "04000000US01" "04000000US04" "04000000US05" "04000000US06" ...	
... ..\$ GEOFID : 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 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 ...	
... ..\$ 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")
```



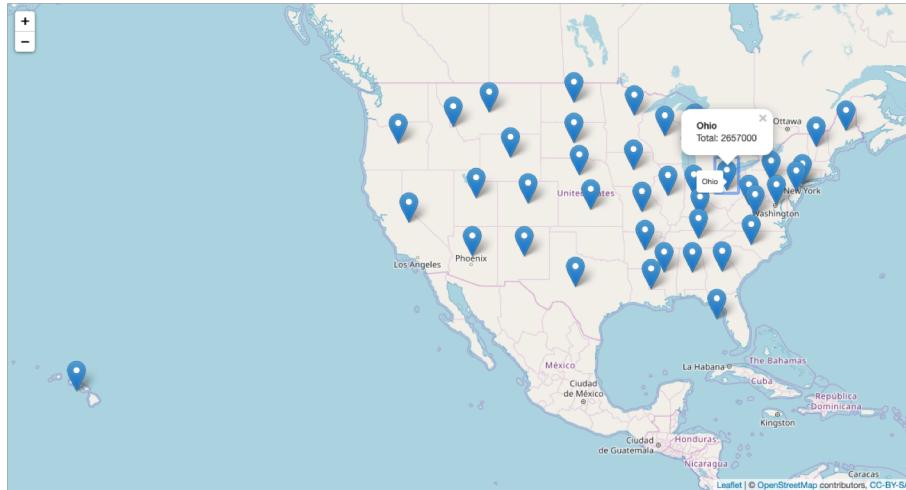
Code example adapted from:  
[http://learn.r-journalism.com/en/mapping/census\\_maps/census-maps/](http://learn.r-journalism.com/en/mapping/census_maps/census-maps/)

# Interactive map with overlays

- Latitudes and longitudes for U.S. state centers are from:

<https://www.kaggle.com/washimahmed/usa-latlong-for-state-abbreviations>

```
1 stll <- read_csv("data/statelatlong.csv") %>% rename(state=State)
2 honey2012 %>% left_join(stll) %>% mutate(plab=sprintf("<b>%s</b><br>Total:<br>%d", name, prodvalue)) %>%
3   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/>

The screenshot shows the DataCamp course landing page for 'Interactive Maps with leaflet in R'. At the top, it says 'INTERACTIVE COURSE' and the course title 'Interactive Maps with leaflet in R'. Below the title are two buttons: 'Start Course For Free' (yellow) and 'Play Intro Video' (blue). To the right is a circular icon containing a stylized map and the text 'INTERACTIVE MAPS WITH LEAFLET IN R'. At the bottom, it shows course statistics: '4 hours', '16 Videos', '55 Exercises', '2,463 Participants', and '4,500 XP'.

## Course Description

Get ready to have some fun with maps! Interactive Maps with leaflet in R will give you the tools to make attractive and interactive web maps using spatial data and the tidyverse. In this course, you will create maps using the IPEDS dataset, which contains data on U.S. colleges and universities. Along the way, you will customize our maps using labels, popups, and custom markers, and add layers to enhance interactivity. Following the course, you will be able to create and customize your own interactive web maps to reveal patterns in your data.

This course is part of these tracks:

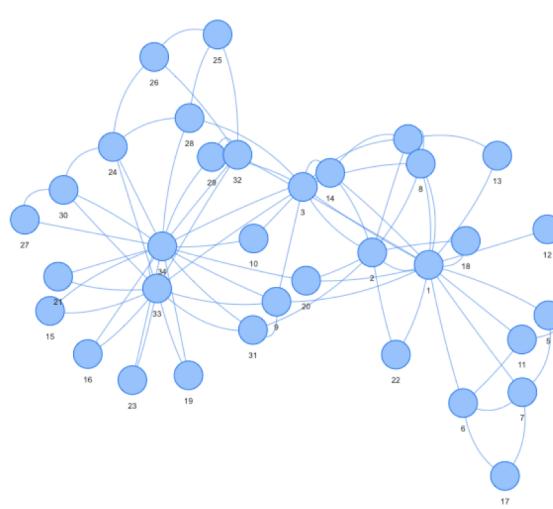
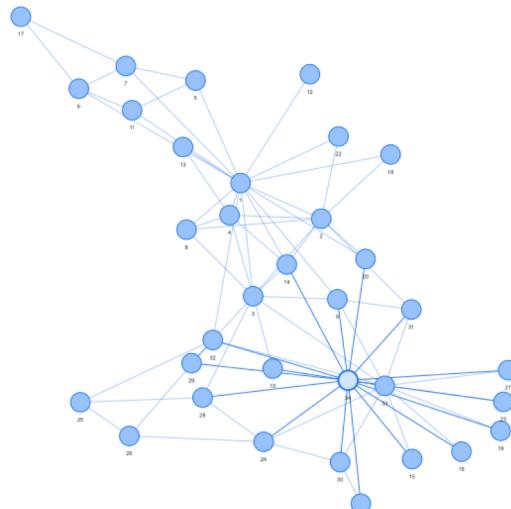
Spatial Data with R

# 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](http://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)
```



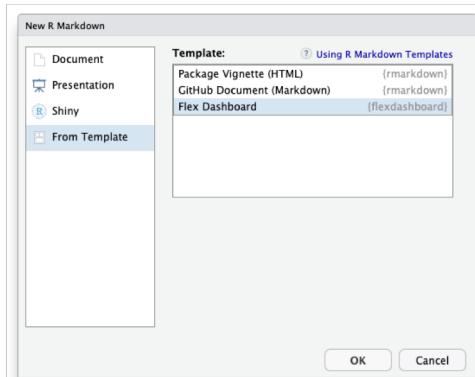
# Dashboard Design

# 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>

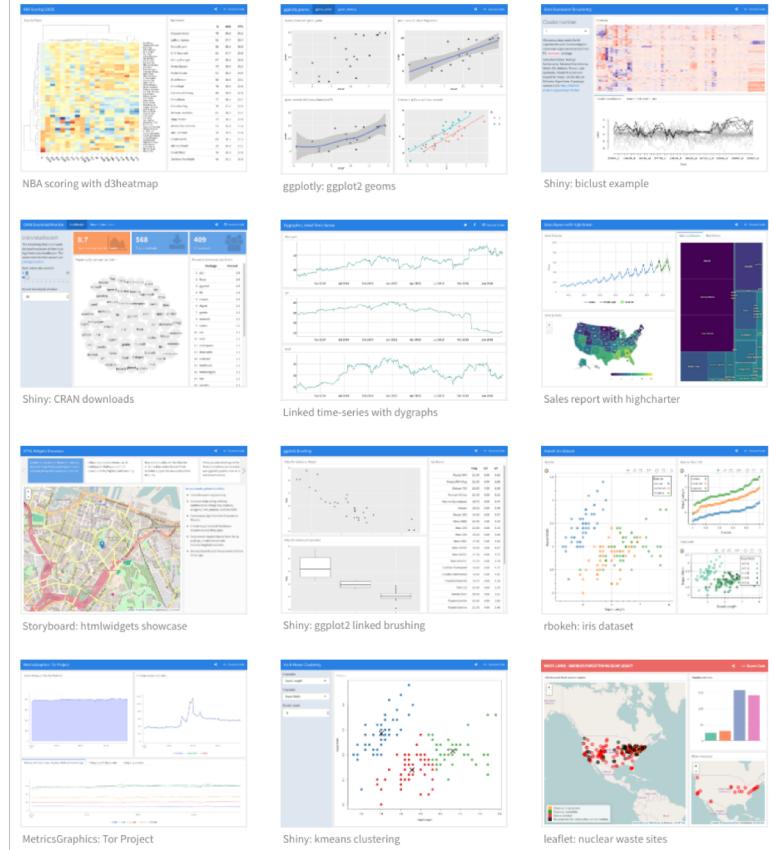
# flexdashboard

- 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.rstu](https://rmarkdown.rstudio.com/flexdashboard/examples.html)dio.com/flexdashboard/examples.html



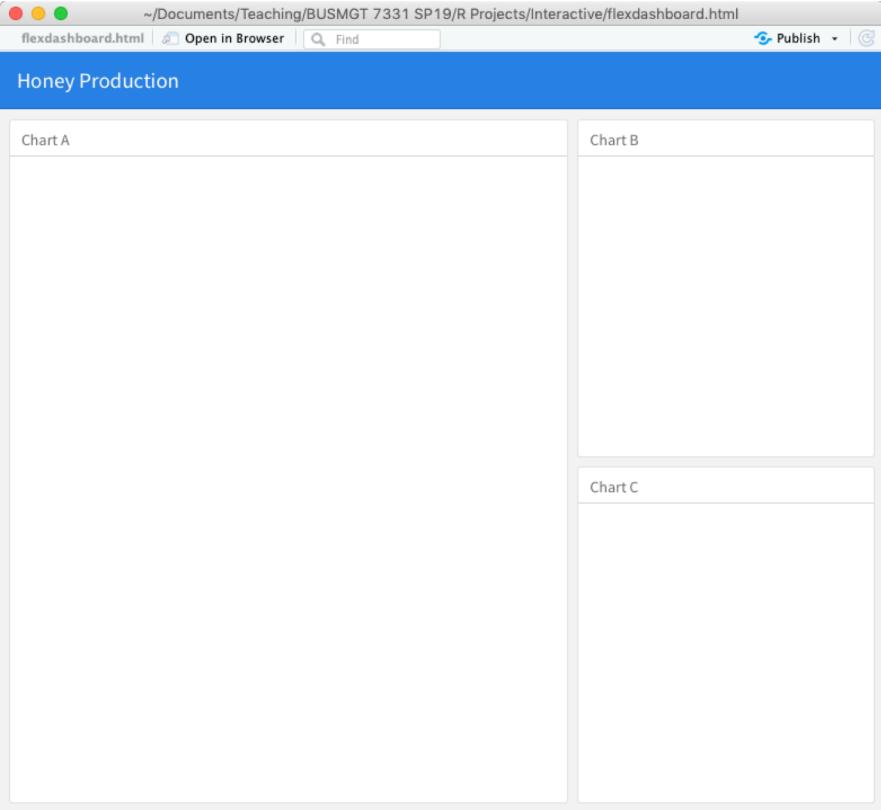
## flexdashboard Examples

The examples below illustrate the use of flexdashboard with various packages and layouts. If you want to learn more about how the dashboards were created each example includes a link to its source code.



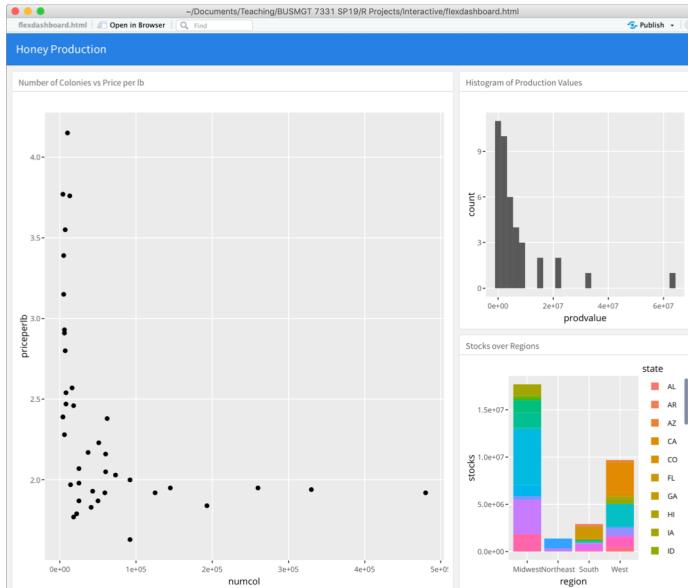
# Layout of a default flexdashboard

```
1 ---  
2 title: "Honey Production"  
3 output:  
4   flexdashboard::flex_dashboard:  
5     orientation: columns  
6     vertical_layout: fill  
7 ---  
8  
9 `r setup, include=FALSE}  
10 library(flexdashboard)  
11 ````  
12  
13 Column {data-width=650}  
14 ---  
15  
16 ## Chart A  
17  
18 `r}  
19  
20 ````  
21  
22 Column {data-width=350}  
23 ---  
24  
25 ## Chart B  
26  
27 `r}  
28  
29 ````  
30  
31 ## Chart C  
32  
33 `r}  
34  
35 ````  
36  
37
```



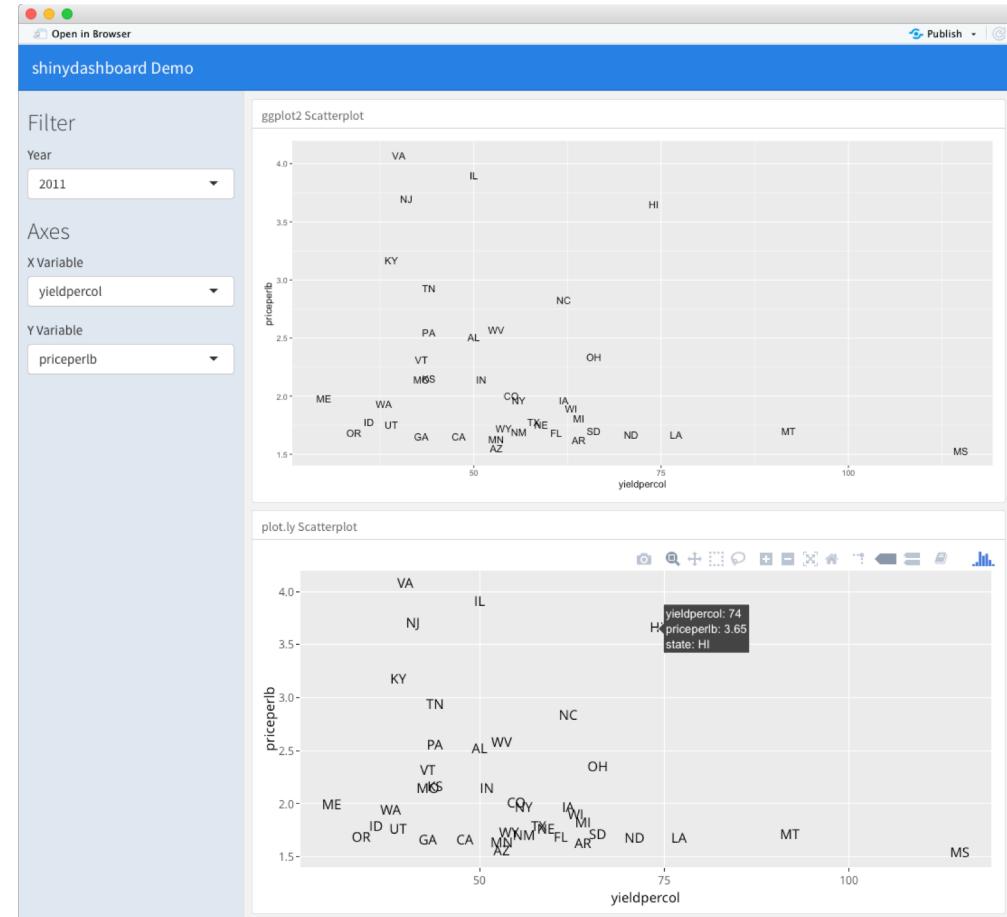
# Fleshing out the wireframe

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



# Interactive data exploration with shinydashboard

```
1 ---  
2 title: "shinydashboard Demo"  
3 output:  
4   flexdashboard:flex_dashboard:  
5     orientation: columns  
6     vertical_layout: fill  
7 runtime: shiny  
8 ---  
9  
10 ````{r setup, include=FALSE}  
11 library(tidyverse)  
12 library(plotly)  
13 honey <- read_csv("data/honeyproduction.csv", col_types=cols(year=col_integer()))  
14 ````  
15  
16 Inputs {.sidebar}  
17 -----  
18  
19 ## Filter  
20 ````{r}  
21 selectInput("year", "Year", sort(unique(honey$year)))  
22 ````  
23  
24 ## Axes  
25 ````{r}  
26 selectInput("xcol", "X Variable", names(honey)[2:7])  
27 selectInput("ycol", "Y Variable", names(honey)[2:7])  
28  
29  
30 Column  
31 -----  
32  
33 ## ggplot2 Scatterplot  
34 ````{r}  
35 selectedData <- reactive({honey %>% filter(year==input$year)})  
36 renderPlot({  
37   ggplot(selectedData(), aes(label=state)) +  
38   geom_text(aes_string(x=input$xcol, y=input$ycol))  
39 })  
40 ````  
41  
42 ## plot.ly Scatterplot  
43 ````{r}  
44 renderPlotly({  
45   p <- ggplot(selectedData(), aes(label=state)) +  
46   geom_text(aes_string(x=input$xcol, y=input$ycol))  
47   ggplotly(p)  
48 })  
49 ````
```



# 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](#)

INTERACTIVE COURSE

## Building Dashboards with flexdashboard

[Start Course For Free](#)

🕒 4 hours | ➜ 14 Videos | ➤ 50 Exercises | 🌐 1,508 Participants | 📈 4,150 XP



**Course Description**

Communication is a key part of the data science process. Dashboards are a popular way to present data in a cohesive visual display. In this course you'll learn how to assemble your results into a polished dashboard using the flexdashboard package. This can be as simple as adding a few lines of R Markdown to your existing code, or as rich as a fully interactive Shiny-powered experience. You will learn about the spectrum of dashboard creation tools available in R and complete this course with the ability to produce a professional quality dashboard.

This course is part of these tracks:  
[Shiny Fundamentals with R](#)

INTERACTIVE COURSE

## Building Dashboards with shinydashboard

[Start Course For Free](#) | [Play Intro Video](#)

🕒 4 hours | ➜ 13 Videos | ➤ 45 Exercises | 🌐 8,031 Participants | 📈 3,750 XP



**Course Description**

Once you've started learning tools for building interactive web applications with shiny, this course will translate this knowledge into building dashboards. Dashboards, a common data science deliverable, are pages that collate information, often tracking metrics from a live-updating data source. You'll gain more expertise using shiny while learning to build and design these dynamic dashboards. In the process, you'll pick up tips to optimize performance as well as best practices to create a visually appealing product.

This course is part of these tracks:  
[Shiny Fundamentals with R](#)

# 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)