Data Storytelling with R

June 1, 2016

DRAFT

Jeff Chen
Chief Data Scientist, DOC

Star Ying
Data Scientist, DOC

Outline

- Refresher and Background
- Beginning with GGPlot2
- DataTables
- dygraph and threejs
- Leaflet
- Knitting with RMarkdown
- Conclusion

COMMERCE DATA SERVICE

Refresher

is a free environment for statistical computing and graphics

- Object oriented programming
- >6000 packages
- Compares favorably to other statistical packages

COMMERCE DATA SERVICE

Refresher

RStudio® is a free user-interface for R

- Includes a console, editor, plus an area to see your object environment, plots, help, and history

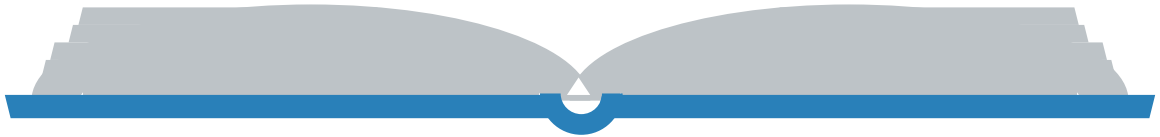
Refresher

```
# this is a comment

this <- assigns_that

c('concatenates','things','together')

matrix(entries,ncol=#columns,nrows=#rows)

dataframe(x=object_1,y=object_2,...)
```

COMMERCE DATA SERVICE

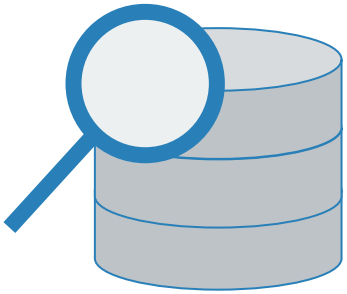
Background

Data Storytelling



COMMERCE DATA SERVICE

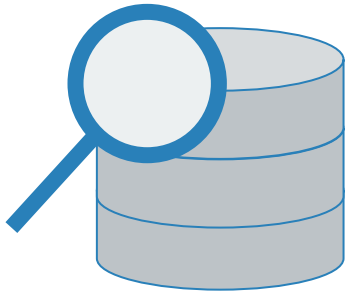
Background



Data Storytelling vs Data Analysis

COMMERCE DATA SERVICE

Background



Presenting your work and results

Lead to a decision or action

Focus toward your client

High level of polish

External review

COMMERCE DATA SERVICE

Background

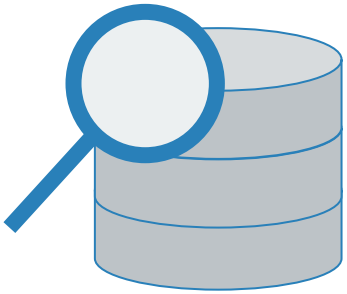
Presenting your work and results

Lead to a decision or action

Focus toward your client

High level of polish

External review



Elucidating your process and analysis

Focus on analysis and modeling

Made with researcher in mind

Low level of polish

Internal use

Background

Data Storytelling

Content

COMMERCE DATA SERVICE

Background

Data Storytelling

Content

Visualization

COMMERCE DATA SERVICE

Background

Data Storytelling

Content

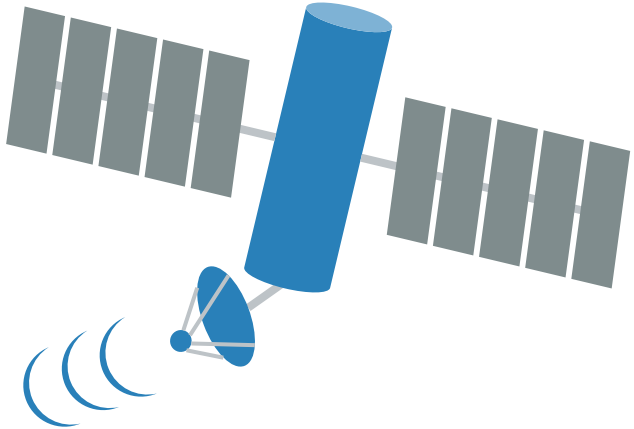
Visualization

Interactivity

COMMERCE DATA SERVICE

Background

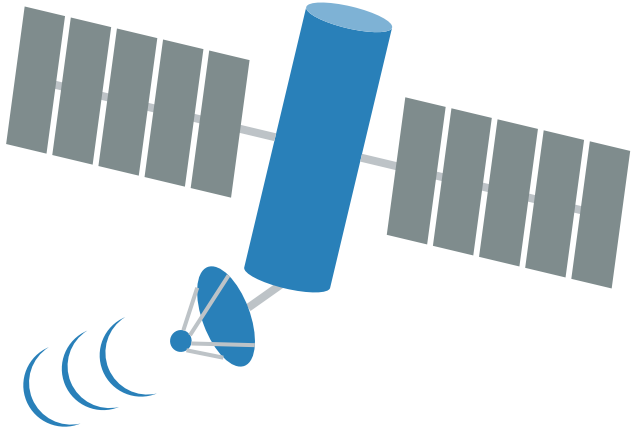
Content



COMMERCE DATA SERVICE

Background

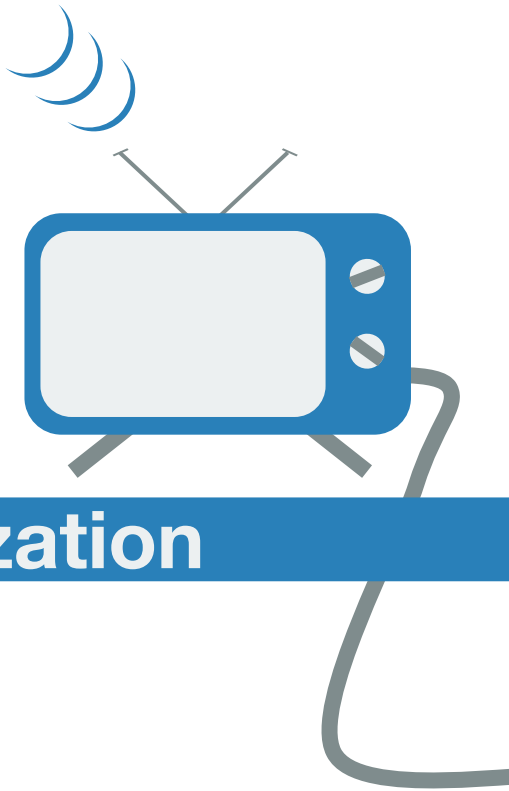
Content



- Focus on the main takeaway you want to convey
- Consider the audience's perspective
- Limit technical details to audience skill level
- Highlight critical assumptions made in analysis
- Should not cover everything done in data analysis
- Order content to lead to main takeaway

COMMERCE
DATA SERVICE

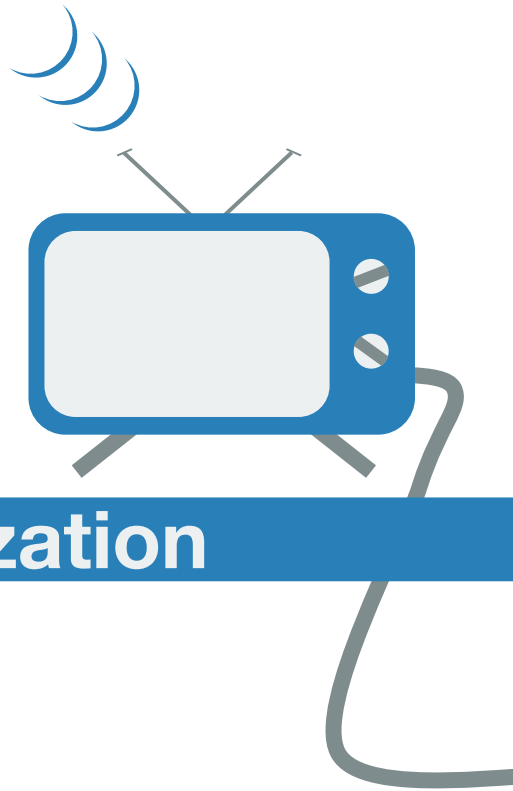
Background



Visualization

COMMERCE DATA SERVICE

Background

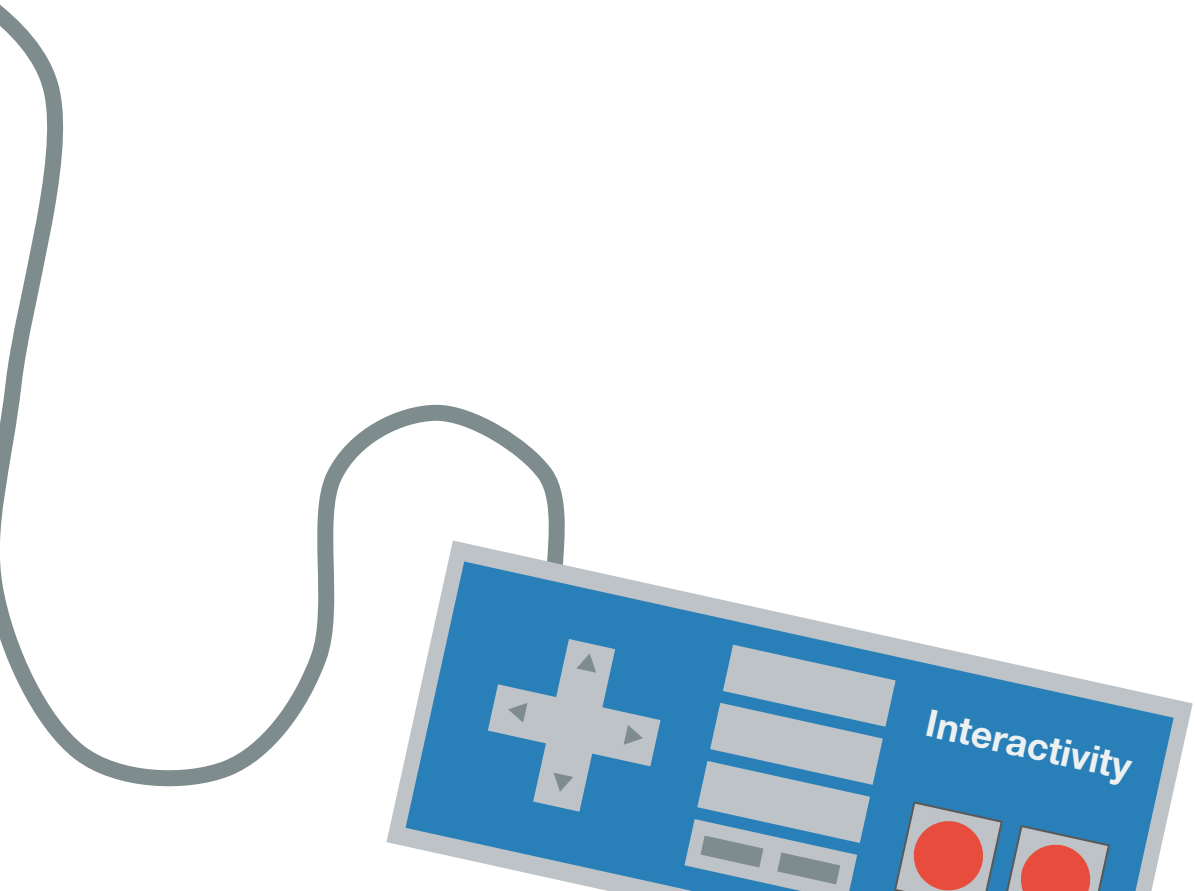


Visualization

- Only include pertinent visualizations
- Should clarify specific components of your data analysis
- Highlight and reinforce your main takeaway
- Mirror the order of the content
- Use to punctuate and control tempo
- Be stylistically cohesive

COMMERCE DATA SERVICE

Background



COMMERCE
DATA SERVICE

Background

Balance between visually arresting and actionable

Dashes should map 1 to 1 with specific policy decisions

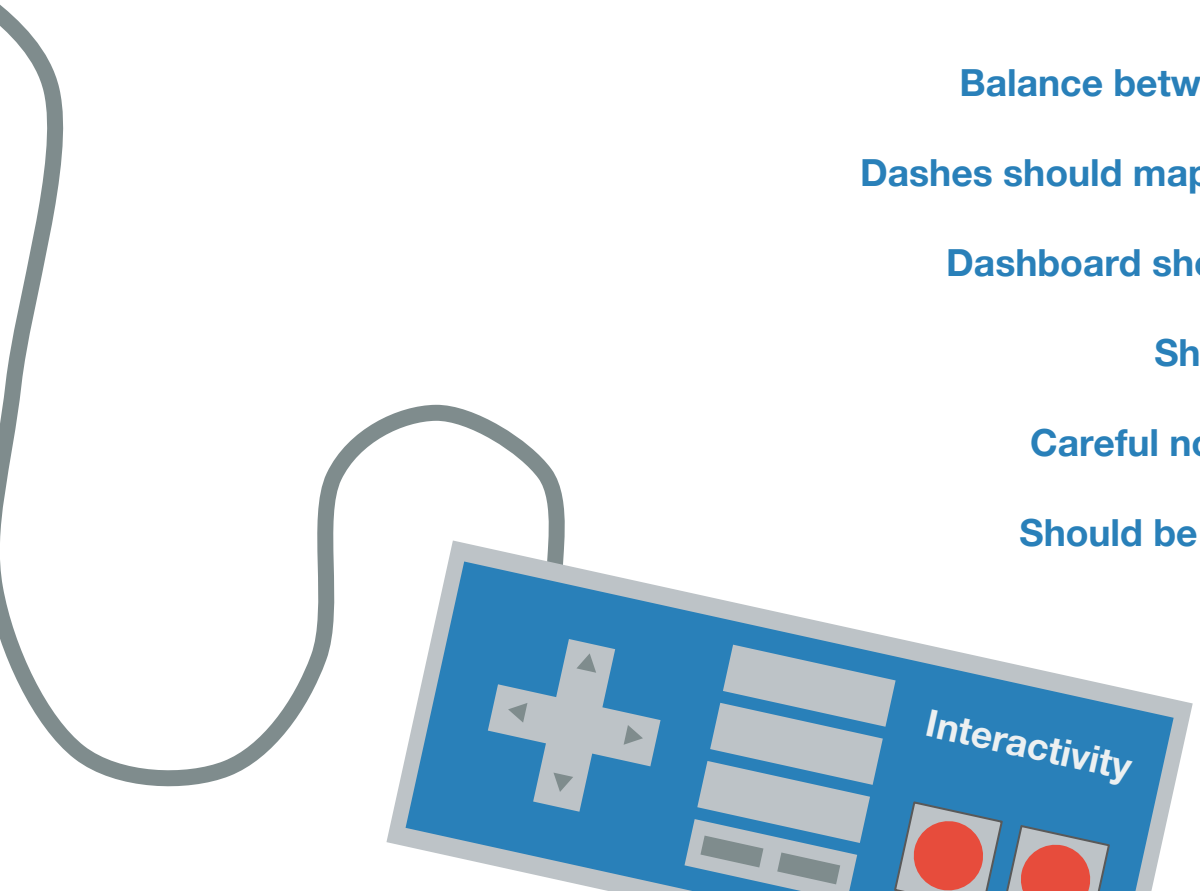
Dashboard should not be exploratory data analysis

Should clarify not compound questions

Careful not to overwhelm with too much noise

Should be used to declare your main takeaway

Be stylistically cohesive



GGPlot2

Simplified API versus vanilla R graphics API

Staple of data analysis process

Can be used for data storytelling as well

Produces well formed static visualizations

Useful in quickly wireframing an interactive visualization

GGPlot2

```
#install ggplot2
install.packages('ggplot2')

#load library
library(ggplot2)

#check example dataset
head(economics_long)
```

GGPlot2

```
#create a ggplot area and add specific geoms
ggplot(data, aes(x,y,color,group)) + geom_point()

#example with economics_long
ggplot(economics_long, aes(x=date, y=value01,
   color=variable, group=variable))
   + geom_line()

#example as area contours
ggplot(economics_long, aes(x=date, y=value01,
   color=variable, group=variable))
   + geom_area()
```

GGPlot2

```
#reshape your own tabular data

#install reshape2
install.packages('reshape2')

#load library
library(reshape2)

#command to change to long form
new_var <- melt(dataframe, time_var)
```

COMMERCE DATA SERVICE

GGPlot2

```
#check example dataset
head(economics)

#pick a variable you want to plot
head(economics[c(1,3)])

#reshape data
economics_melt <- melt(economics[c(1,3)],date)

#check new set
head(economics_melt)
```

Exercise

Reshape another variable in economics dataset and use ggplot2 to plot

COMMERCE DATA SERVICE

Break

10 minute break

COMMERCE DATA SERVICE

DataTables

Present tabular data in a searchable paginated table

Useful for presenting the cleaned raw data to the client

Convey the shape of data

Useful in connecting to subject matter knowledge

Useful in discussing specific cases

COMMERCE DATA SERVICE

DataTables

```
#install DataTables
install.packages('DT')

#load package
library(DT)
```

COMMERCE DATA SERVICE

DataTables

```
#create interactive table
datatable(data, options = list(),
    rownames, colnames, container, caption = NULL,
    filter = c("none", "bottom", "top"),
    ...)

#simple example
datatable(economics)

#customize page length
datatable(economics, options = list(pagelength = 50))
```

dygraph

R interface to popular dygraphs javascript library

Useful in presenting time-series data interactive form

Automatically graphs xts time-series objects

Helpful in visualizing long time-series

dygraph

```
#install dygraphs and xts
install.packages('dygraphs')
install.packages('xts')

#load packages
library(dygraphs)
library(xts)
```

dygraph

```
#convert dataframe to xts
xts(x = NULL,
    order.by = index(x),
    frequency = NULL,
    unique = TRUE,
    tzone = Sys.getenv("TZ"),
    ...)

#create a dygraph
dygraph(data, main = NULL, xlab = NULL, ylab = NULL,
   periodicity = NULL,
   group = NULL, width = NULL, height = NULL)
```

dygraph

```
#convert string date to date format
date_me <- as.Date(economics[,1], format='%Y-%m-%d')

#combine with variable of interest
value_me <- cbind(date_me, economics[2])

#convert data frame to xts format
plot_me <- xts(value_me, order.by=value_me[,1])
```

dygraph

```
#call dygraph
dygraph(plot_me)

#add some flare
dygraph(plot_me) %>% dyRangeSelector()
```

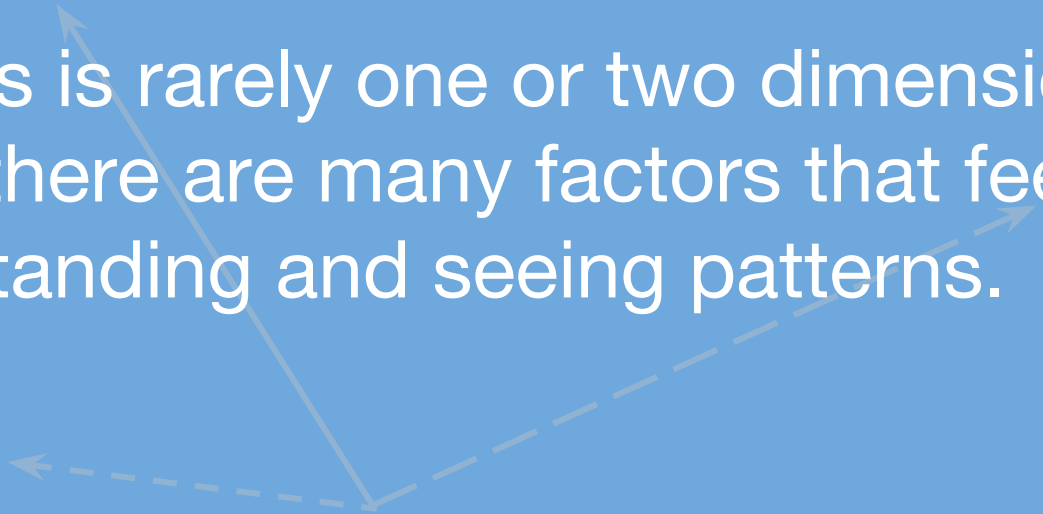
dygraph

```
#plotting multiple series together

#create another xts object
value_me_again <- cbind(date_me, economics
[6])
plot_me_again <- xts(value_me_again,
   order.by=value_me_again[,1])

#pass both to dygraph
dygraphs(cbind(plot_me, plot_me_again))
```

COMMERCE DATA SERVICE

Part II: Framing Thought

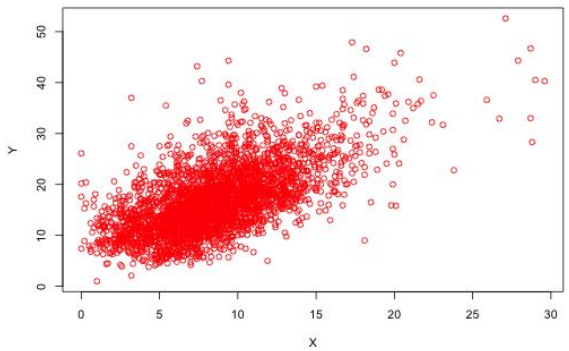
Analysis is rarely one or two dimensional. Often times, there are many factors that feed into understanding and seeing patterns.

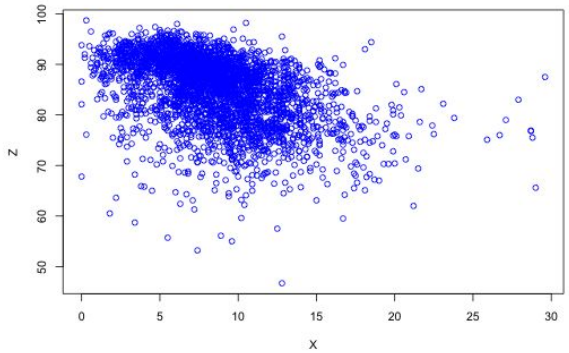


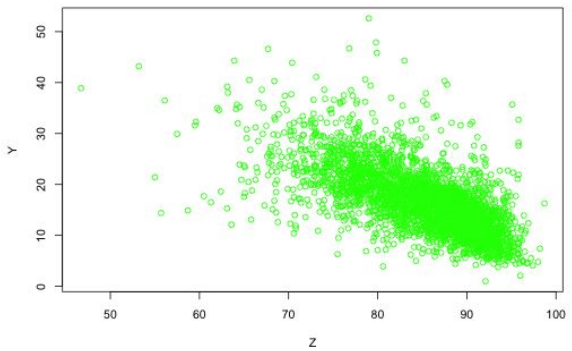
COMMERCE DATA SERVICE

Example

Socioeconomic status is comprised of many factors like poverty status, education, and employment.







COMMERCE DATA SERVICE

Part Deux

Explore county-level (n=3200) American Community Survey data to extract insight from ‘high-dimensional’ data.

COMMERCE DATA SERVICE

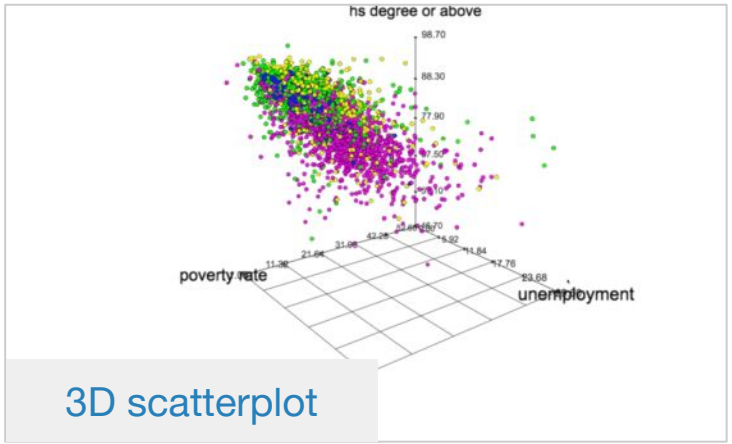
Part Deux

```
#Before we start, just take a moment to load up the base data
setwd("/your/working/dir")
data <- read.csv("data.csv")

#Check your column names
colnames(data)
```

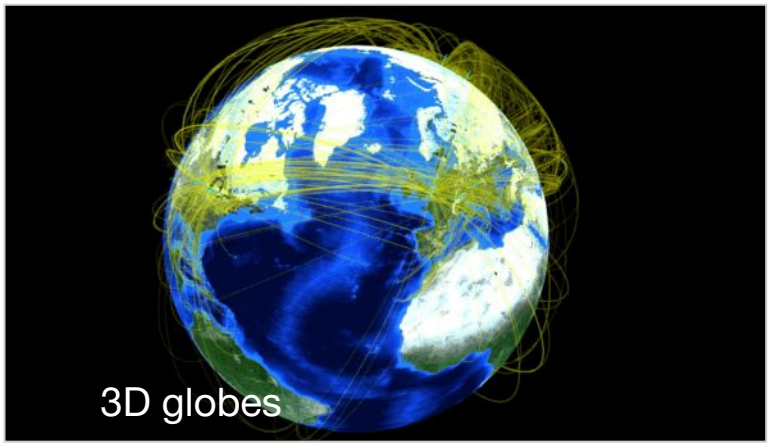
COMMERCE DATA SERVICE

threejs

- Builds upon three.js visualization engine for web browsers
- Accepts vectors, matrices and data frames to create different types of interactive visualizations:



3D scatterplot



3D globes

COMMERCE
DATA SERVICE

threejs

YOUR GOAL THIS SESSION IS TO BUILD A 3D SCATTER

threejs

```
#install threejs (option 1)
devtools::install_github("bwlewis/rthreejs")

#install threejs (option2)
install.packages("threejs")

#load package
library(threejs)
```

threejs

```
#basic syntax
scatterplot3js(x, y, z,
               axisLabels = c(x,z,y),
               color = "steelblue",
               size = 1,
               labels = NULL,
               flip.y = TRUE,
               renderer = c("auto",
               "canvas", "webgl"), ...)
```

COMMERCE DATA SERVICE

threejs

```
#Get a rudimentary 3D plot
scatterplot3js(data$emp_status,
               data$pct_poverty,
               data$hs_grad)
```

COMMERCE DATA SERVICE

threejs

```
#Add axis labels
scatterplot3js(data$emp_status,
data$pct_poverty,data$hs_grad,
                        axisLabels=c(
                            "Unemployment",
                            "HS degree or above",
                            "poverty rate"))
```

threejs

```
#Renderer controls the style + quality
scatterplot3js(data$emp_status,
data$pct_poverty,data$hs_grad,axisLabels=c
("Unemployment","HS degree or above","poverty
rate"),renderer="canvas")
```

COMMERCE DATA SERVICE

threejs

```
#Set the color (col), point size, and orient the y-axis
scatterplot3js(data$emp_status,
data$pct_poverty,data$hs_grad,axisLabels=c("Unemployment","HS degree or above","poverty rate"),renderer="canvas",
   flip.y=FALSE, col="slategrey",size=0.5)
```

threejs

```
#Add context with point labels
scatterplot3js(data$emp_status,
data$pct_poverty,data$hs_grad,axisLabels=c
("Unemployment","HS degree or above","poverty
rate"),renderer="canvas",flip.y=FALSE, col="
slategrey",size=0.5,
labels =data$region_name)
```

What we can tell from the 3D graph thus far:

- Inverse: More education is associated with less poverty and less unemployment
- Direct: More poverty is associated with more unemployment

We can learn more by adding **color.**

We can learn more by

grouping points with color.

threejs

```
#how many regions?
 unique(data$region)
 unique(data$region_name)

#Create a new var, assign colors
 data$colors <- ""
 data$colors[data$region==1] <- "#011EFE0"
 data$colors[data$region==2] <- "#0BFF01"
 data$colors[data$region==3] <- "#FE00F6"
 data$colors[data$region==4] <- "#FDFE02"
```

threejs

```
#Sort regions (important!!)
data <- data[order(data$region),]

#Set col = data$colors
scatterplot3js(data$emp_status,
data$pct_poverty,data$hs_grad,axisLabels=c
("Unemployment","HS degree or above","poverty
rate"),renderer="canvas",labels
=data$region_name,flip.y=FALSE,size=0.5
col=data$colors)
```

threejs

```
#Finishing
scatterplot3js(data$emp_status,
data$pct_poverty,data$hs_grad,axisLabels=c
("Unemployment","HS degree or above","poverty
rate"),renderer="canvas",
   flip.y=FALSE, col=data$colors,size=0.5)
```

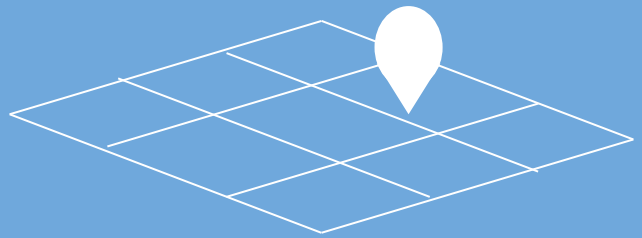
Exercise

- Adjust `size` option to determine if there are points hidden away in middle of point cloud
- Substitute variable data$emp_status with “log (data$households)”. How does the clustering change?

COMMERCE DATA SERVICE

Maps

Sometimes graphs don't get the point across. **Maps** can provide enable easier discovery of patterns.

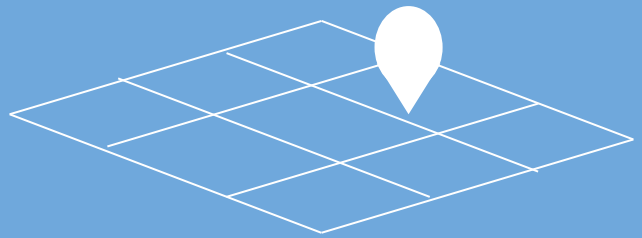


COMMERCE DATA SERVICE

Maps

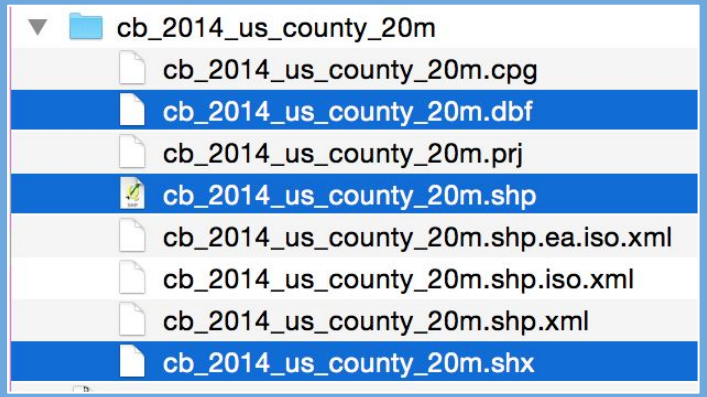
GIS = Geographic Information Systems

Common file type = shapefile or .shp



COMMERCE DATA SERVICE

Maps



Shapefiles need 3 files

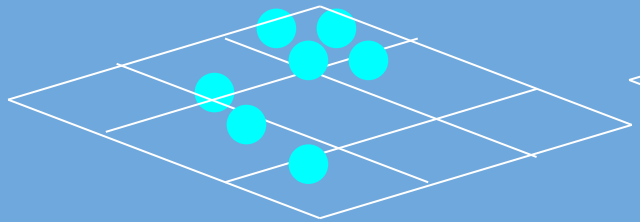
All come as a package

- .shp = contains geometry data
- .dbf = attributes (e.g. data)
- .shx = shape index, positional index

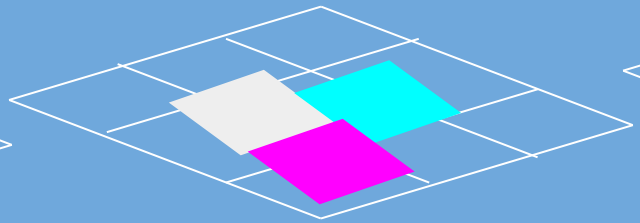
COMMERCE DATA SERVICE

Maps

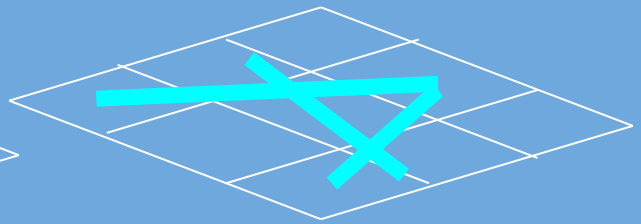
From 3 basic files in a shapefile...



points



polygons



lines

COMMERCE DATA SERVICE

Maps

Basic workflow

Import .shp
Import data

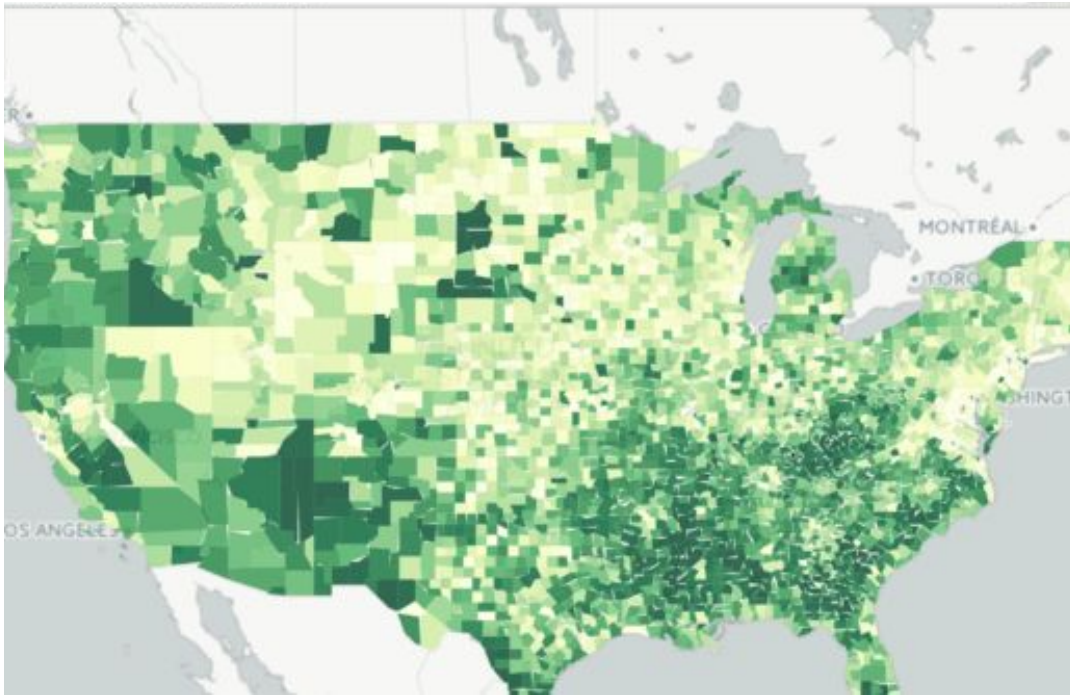
Join .shp
to data

Map using
leaflet

COMMERCE DATA SERVICE

leaflet

- Builds upon leaflet.js to create interactive web maps
- Functionality to take a dataset to map in minutes



COMMERCE DATA SERVICE

Leaflet

```
#install leaflet
install.packages("leaflet")
install.packages("rgdal")
install.packages("stringr")

#load package
library(leaflet) #mapping
library(rgdal) #geoprocessing
library(stringr) #text manipulation
```

COMMERCE DATA SERVICE

Leaflet

```
#load in shapefile
shp = readOGR("shp/cb_2014_us_county_20m.shp",
              "cb_2014_us_county_20m")

#load map
leaflet(data = shp) %>%
     addPolygons(fillColor = "blue")
```

COMMERCE DATA SERVICE

Leaflet

```
#neaten up the polygons
leaflet(data = shp) %>%
      addPolygons(fillColor ="blue",
      fillOpacity= 0.8,color = "white",
      weight = 0.5)
```

COMMERCE DATA SERVICE

Leaflet

```
#add in a background using free map tiles
leaflet(data = shp) %>%
      addPolygons(fillColor ="blue",
      fillOpacity= 0.8,color = "white",
      weight = 0.5) %>%
      addProviderTiles("CartoDB.Positron")
```

COMMERCE DATA SERVICE

Leaflet

```
#center on the contiguous US, zoom in
leaflet(data = shp) %>%
      addPolygons(fillColor ="blue",
      fillOpacity= 0.8,color = "white",
      weight = 0.5) %>%
      addProviderTiles("CartoDB.Positron") %>%
      setView(lng = -98.3,lat = 39.5, zoom = 4)
```

COMMERCE DATA SERVICE

Leaflet

```
#center on the contiguous US, zoom in
leaflet(data = shp) %>%
      addPolygons(fillColor ="blue",
      fillOpacity= 0.8,color = "white",
      weight = 0.5) %>%
      addProviderTiles("CartoDB.Positron") %>%
      setView(lng = -98.3,lat = 39.5, zoom = 4)
```

COMMERCE DATA SERVICE

Maps

Basic workflow

Import .shp

Import data

Join .shp to data

Map using leaflet

COMMERCE DATA SERVICE

Leaflet

```
##GOAL: join data and shp@data. Figure out which
columns go together

#Check column names in data
   str(data)

#Check column names in shp
   str(shp@data)
```

COMMERCE DATA SERVICE

Leaflet

```
##Standardize columns before merge
# Convert data$GEOID into 5 character string
data$GEOID <- str_pad(as.character(data$id2), 5,
pad = "0")

#Convert shp@data$GEOID into a string
   shp@data$GEOID <- as.character(shp@data$GEOID)

#Join data using GEOID
   shp <- merge(shp,data,id="GEOID")
```

Leaflet

```
##Setup color palette (Yellow to Green, 30 breaks)
#Wrapper function for converting data into color scale
   palette <- colorQuantile("YlGn", NULL, n = 30)
```

COMMERCE DATA SERVICE

Leaflet

```
#center on the contiguous US, zoom in
leaflet(data = shp) %>%
      addPolygons(fillColor=~palette(pct_poverty),
      , fillOpacity= 0.8,color = "white",
      weight = 0.5) %>%
      addProviderTiles("CartoDB.Positron") %>%
      setView(lng = -98.3,lat = 39.5, zoom = 4)
```

What we can tell from the leaflet map

- Clear patterns in poverty by geographic region
- Northeast and northern midwest are better off

Exercise

- Substitute other variables for pct_poverty. Do the patterns change?
- Substitute “CartoDB.Positron” with other tiles like CartoDB.DarkMatter or Stamen.Toner

COMMERCE DATA SERVICE

Break

10 minute break

COMMERCE DATA SERVICE

Brought to you by

Commerce Data Academy

DataAcademy@doc.gov