

# Data Storytelling with

June 1, 2016



**Jeff Chen**

Chief Data Scientist, DOC

**Star Ying**  
Data Scientist, DOC



Refresher and Background

Beginning with GGPlot2

DataTables

dygraph and threejs

Leaflet

Knitting with RMarkdown

Conclusion



## **R** is a free environment for statistical computing and graphics

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

## Studio<sup>®</sup> is a free user-interface for R

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

```
# this is a comment
```

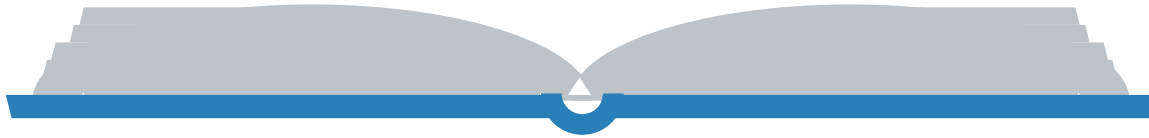
```
this <- assigns_that
```

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

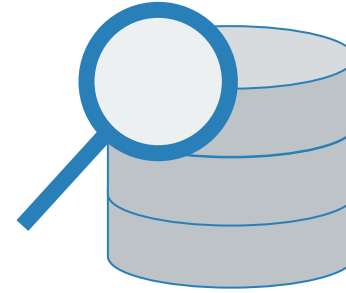
```
matrix(entries, ncol=#columns, nrow=#rows)
```

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

## Data Storytelling



# Background



## Data Storytelling vs Data Analysis



# Background



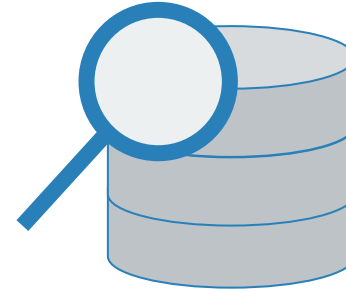
**Presenting your work and results**

**Lead to a decision or action**

**Focus toward your client**

**High level of polish**

**External review**





**Presenting your work and results**

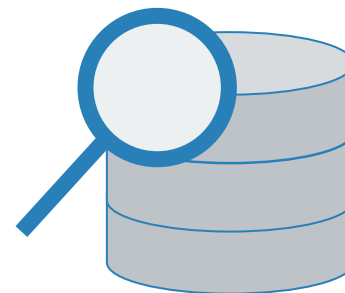
**Lead to a decision or action**

**Focus toward your client**

**High level of polish**

**External review**

# Background



**Elucidating your process and analysis**

**Focus on analysis and modeling**

**Made with researcher in mind**

**Low level of polish**

**Internal use**

## Data Storytelling

### Content

# Background

## Data Storytelling

Content

Visualization

## Data Storytelling

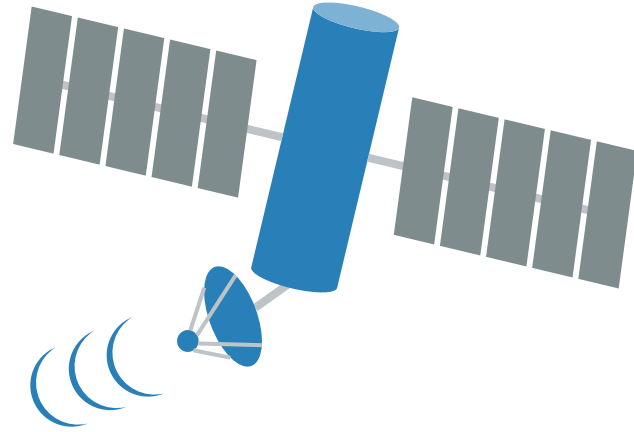
Content

Visualization

Interactivity

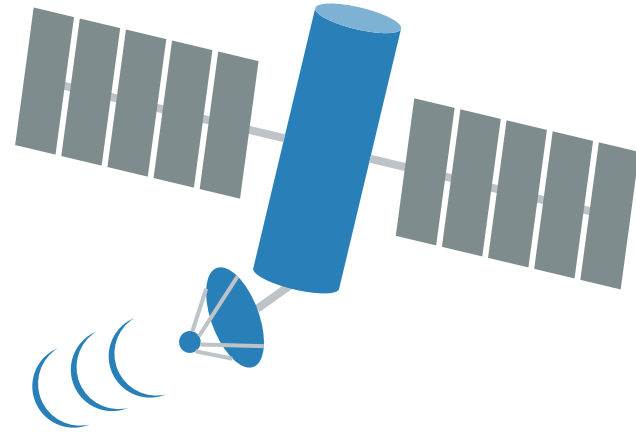
# Background

## Content

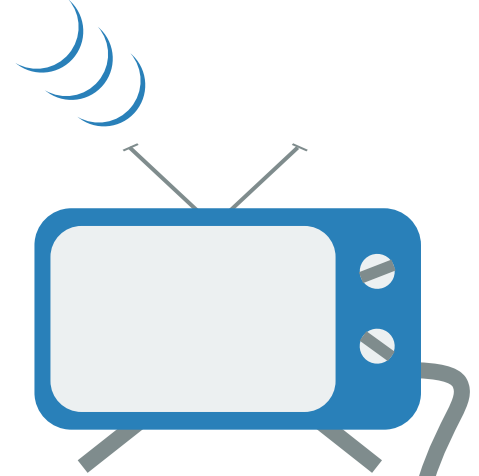


## Content

- 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



# Background



## Visualization



**Only include pertinent visualizations**

**Should clarify specific components of your data analysis**

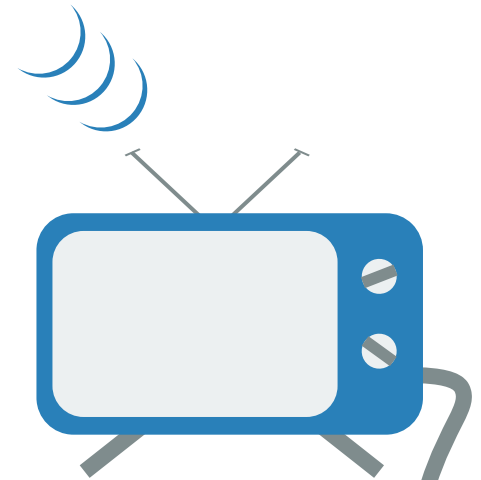
**Highlight and reinforce your main takeaway**

**Mirror the order of the content**

**Use to punctuate and control tempo**

**Be stylistically cohesive**

# Background



# Visualization

# Background



# 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



**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**

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

```
#load library  
library(ggplot2)
```

```
#check example dataset  
head(economics_long)
```

```
#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()
```

#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)
```

```
#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



**10 minute  
break**

**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**

# DataTables

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

```
#load package  
library(DT)
```

```
#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))
```

**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**

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

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

```
#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)
```



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

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

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

```
#call dygraph  
dygraph(plot_me)
```

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

```
#plotting multiple series together
```

```
#create another xts object
```

```
value_me_again <- cbind(date_me, economics  
[6])
```

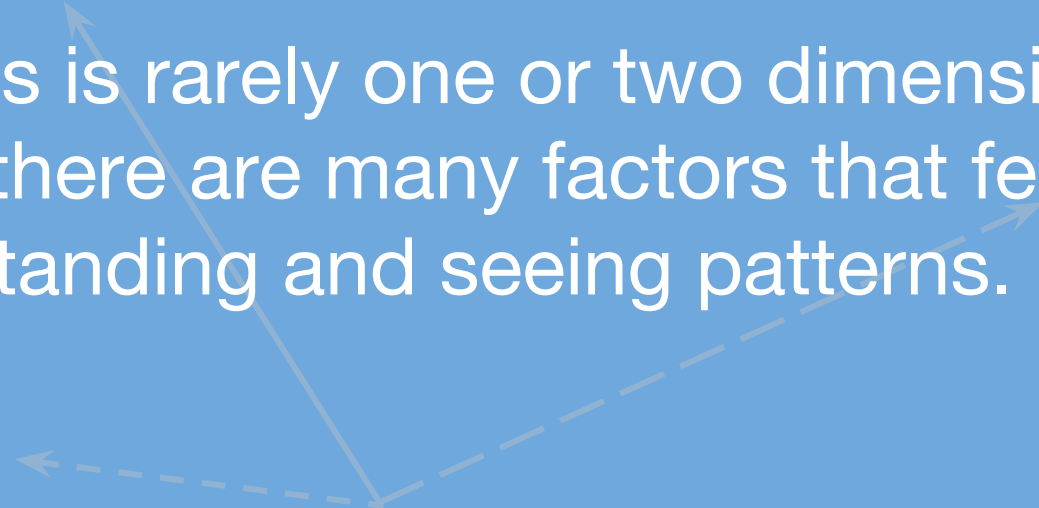
```
plot_me_again <- xts(value_me_again,  
  order.by=value_me_again[,1])
```

```
#pass both to dygraph
```

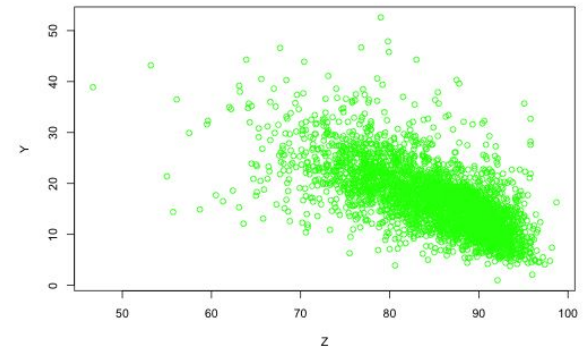
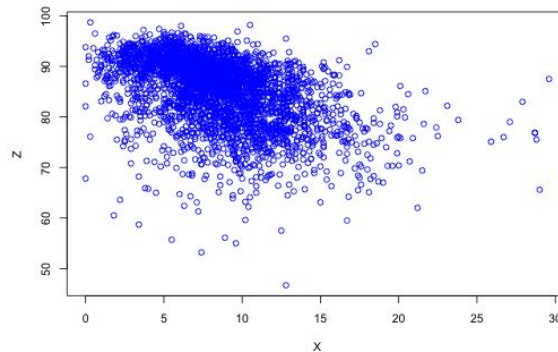
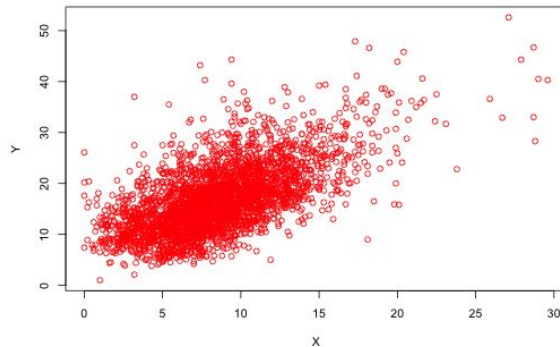
```
dygraphs(cbind(plot_me, plot_me_again))
```

## Part II: Framing Thought

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

A diagram of a 3D coordinate system with three axes. One axis is a solid line pointing towards the top-left, another is a dashed line pointing towards the bottom-left, and the third is a dashed line pointing towards the top-right. All three axes meet at a common origin point.

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



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

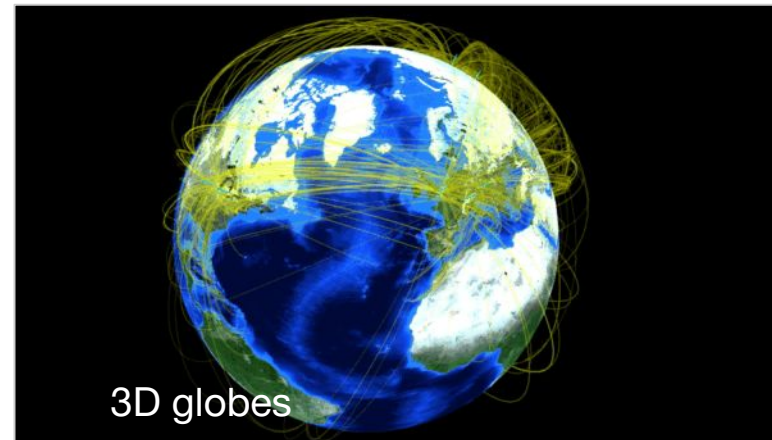
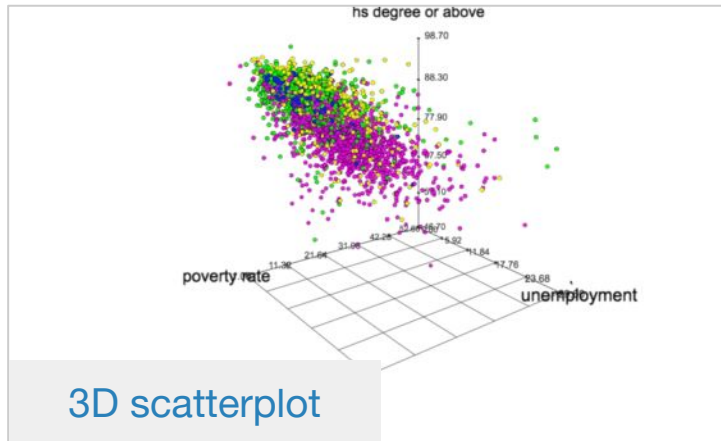
**#Before we start, just take a moment to load up the base data**

```
setwd("/your/working/dir")  
data <- read.csv("base_file.csv")
```

**#Check your column names**

```
colnames(data)
```

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







**YOUR GOAL THIS SESSION IS  
TO BUILD A 3D SCATTER**

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

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

```
#load package  
library(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"), ...)
```

**#Get a rudimentary 3D plot**

```
scatterplot3js(data$emp_status,  
               data$pct_poverty,  
               data$hs_grad)
```

## #Add axis labels

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

**#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")
```

**#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)
```

## #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  
grouping points with **color**.

**#how many regions?**

```
unique(data$region)  
unique(data$region_name)
```

**#Create a new var, assign colors**

```
data$colors <- ""  
data$colors[data$region==1] <- "#11EFE0"  
data$colors[data$region==2] <- "#0BFF01"  
data$colors[data$region==3] <- "#FE00F6"  
data$colors[data$region==4] <- "#FDFE02"
```

**#Sort regions (important!!)**

```
data <- data[order(data$region), ]
```

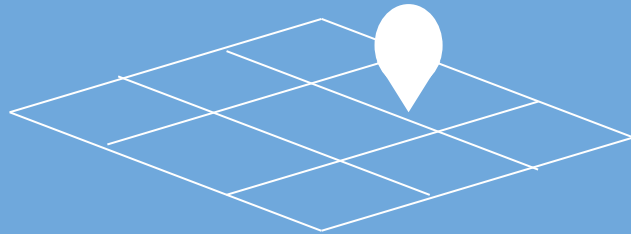
## #Finishing

```
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)
```

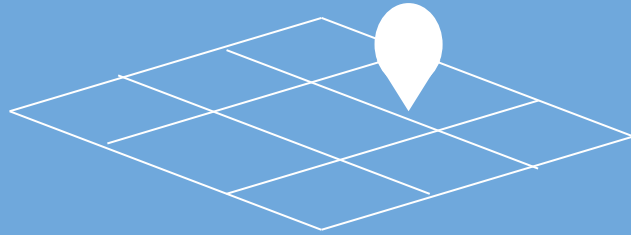
## Exercise

- 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?

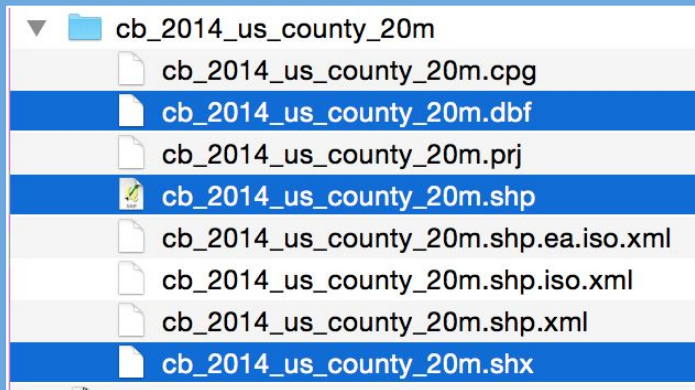
Sometimes graphs don't get the point across.  
**Maps** can provide enable easier discovery of patterns.



GIS = Geographic Information Systems  
Common file type = shapefile or .shp





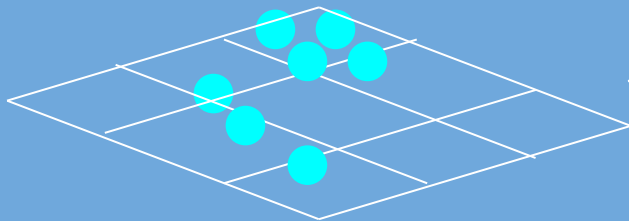


## Shapefiles need 3 files

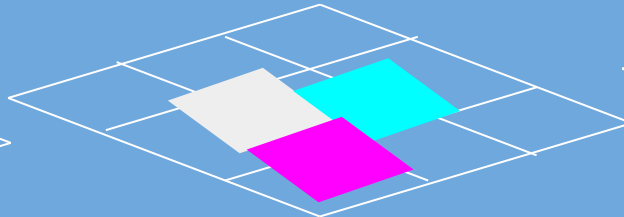
*All come as a package*

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

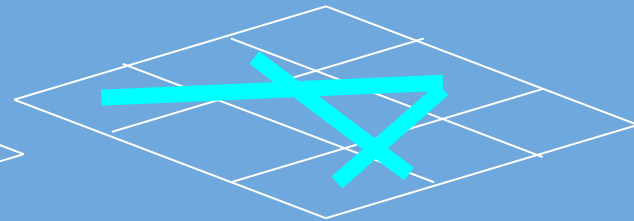
From 3 basic files in a shapefile...



points



polygons



lines

## Basic workflow

Import .shp  
Import data

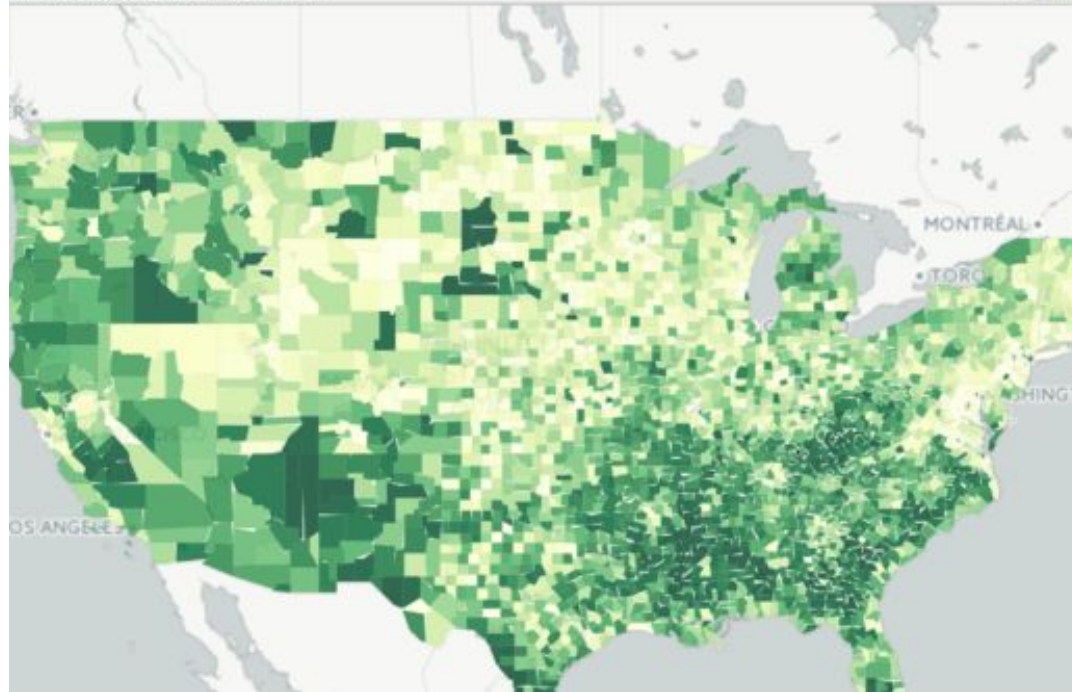


Join .shp  
to data



Map using  
leaflet

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



## **#install leaflet**

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

## **#load package**

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

## **#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")
```

**#neaten up the polygons**

```
leaflet(data = shp) %>%  
  addPolygons(fillColor = "blue",  
    fillColorOpacity = 0.8, color = "white",  
    weight = 0.5)
```

**#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")
```



```
#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)
```



**10 minute  
break**

## Basic workflow

Import .shp  
Import data



**Join .shp  
to data**



Map using  
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)`

```
##Standardize columns before merge  
# Convert data$GE0ID into 5 character string  
data$GE0ID <- str_pad(as.character(data$id2), 5,  
pad = "0")  
  
#Convert shp@data$GE0ID into a string  
shp@data$GE0ID <- as.character(shp@data$GE0ID)  
  
#Join data using GE0ID  
shp <- merge(shp, data, id="GE0ID")
```

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

**#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**

Your graphs do not need to just stay in RStudio.  
They can live as web pages by using  
RMarkdown.

```
#install leaflet  
install.packages("knitr")
```

```
#Quick demo now
```

## Key considerations

- When adding in code to run in RMarkdown, RStudio will need all the code to execute including working directory, datasets, etc
- RMarkdown basically executes the code, takes the output and renders in an HTML format

## Exercise

- Copy your code from one of the packages and insert into RMarkdown. Execute to see what happens
- Experiment with the following:
  - `{r echo=F}`
  - `{r message=F}`
  - `{r warning=F}`

# Conclusion

Visualizations are a tool in telling a story.  
Remember to start with the question and build  
visuals to tell the story. No visualization should  
ever be shown without a purpose.

# Conclusion

[http://www.htmlwidgets.org/showcase\\_leaflet.html](http://www.htmlwidgets.org/showcase_leaflet.html)

# Commerce Data Academy

[DataAcademy@doc.gov](mailto:DataAcademy@doc.gov)