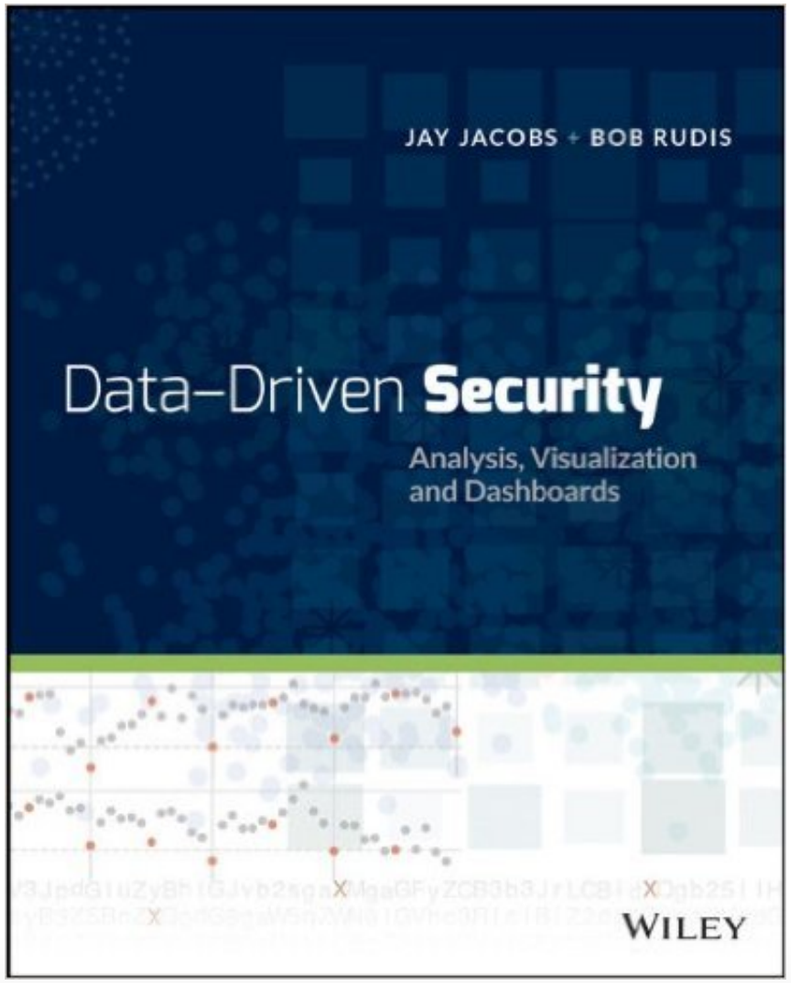
http://dds.ec/ (blog, podcast and...)



Authored/contributed to 12 CRAN packages

Package Name	Updated	Authors	Title	Vignettes	Task View
cdcfluview	2015-08-09	Bob Rudis (@hrbrmstr)	Retrieve U.S. Flu Season Data from the CDC FluView Portal		
cymruservices	2015-07-28	Bob Rudis [aut, cre]	Query Team Cymru IP Address, Autonomous System Number (ASN), Border Gateway Protocol (BGP), Bogon and Malware Hash Data Services		
docxtractr	2015-08-29	Bob Rudis [aut, cre]	Extract Data Tables from Microsoft Word Documents		
ggthemes	2015-07-01	Jeffrey B. Arnold [aut, cre], Gergely Daroczi [c...	Extra Themes, Scales and Geoms for `ggplot2`	1	
iptools	2015-07-23	Bob Rudis [aut, cre], Oliver Key...	Manipulate, Validate and Resolve IP Addresses	2	
longurl	2015-08-21	Bob Rudis [aut, cre]	Expand Short URLs Using the 'LongURL' API		
metricsgraphics	2015-06-14	Bob Rudis [aut, cre], Ali Almossawi [ctb, cph] (...	Create Interactive Charts with the JavaScript 'MetricsGraphics' Library	1	
RBerkeley	2015-07-29	Jeffrey A. Ryan [aut, cre], Bob Rudis [ctb]	Oracle 'Berkeley DB' Interface for R	1	
slackr	2014-09-08	Bob Rudis (@hrbrmstr) & Jay Jacobs (@jayjacobs)	Send messages, images, R objects and files to Slack.com channels/users		WebTechnologies
statebins	2014-08-27	Bob Rudis (@hrbrmstr)	statebins is an alternative to choropleth maps for USA States		
urltools	2015-08-31	Oliver Keyes [aut, cre], Jay Jacobs [aut, cre], ...	Vectorised Tools for URL Handling and Parsing	1	WebTechnologies
viridis	2015-09-14	Simon Garnier [aut, cre], Noam Ross [ctb, cph] (...	Matplotlib Default Color Map	1	
waffle	2015-03-23	Bob Rudis	Create Waffle Chart Visualizations in R		

Where Am I?

- http://rud.is/b Less infosec, more R & vis
- http://twitter.com/hrbrmstr
- http://github.com/hrbrmstr
- http://stackoverflow.com/users/1457051/hrbrmstr
- bob@rudis.net (if you like waiting for responses)

How You May View R

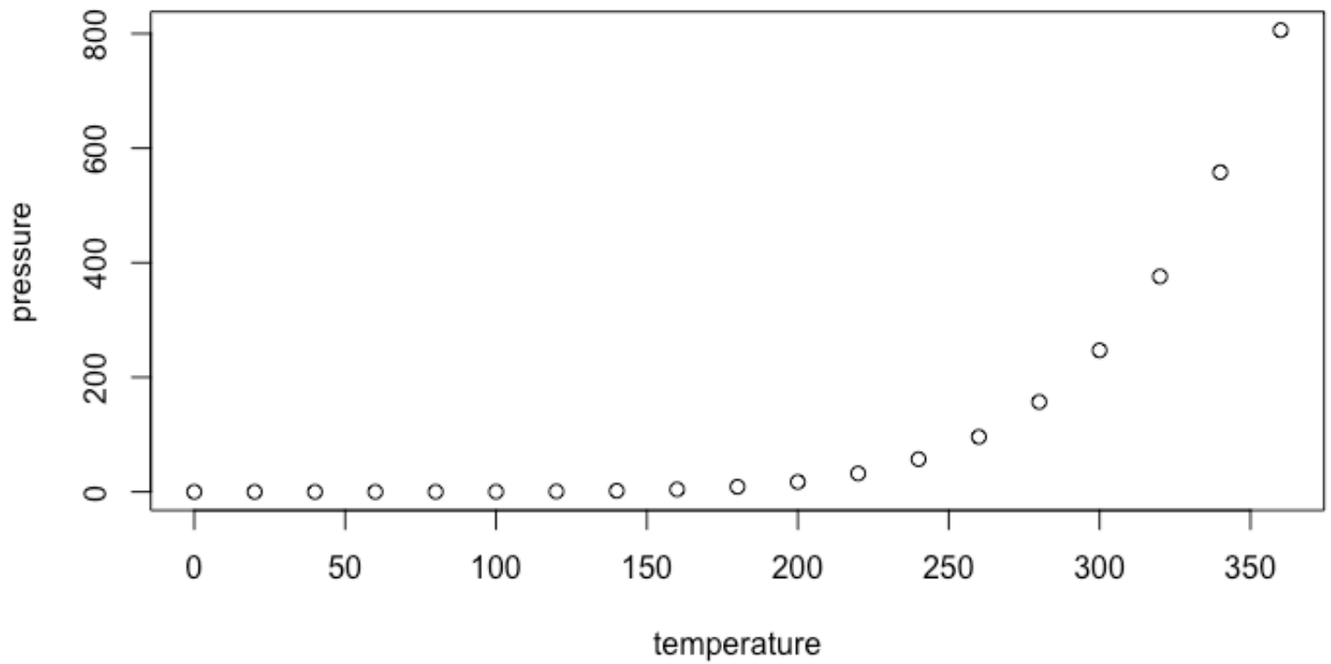
```r
head(pressure, 3)
```

```
##   temperature pressure
## 1           0   0.0002
## 2          20   0.0012
## 3          40   0.0060
```

```r
summary(pressure)
```

```
##   temperature     pressure
##  Min.   :  0   Min.   :  0.0002
##  1st Qu.: 90   1st Qu.:  0.1800
##  Median :180   Median :  8.8000
##  Mean   :180   Mean   :124.3367
##  3rd Qu.:270   3rd Qu.:126.5000
##  Max.   :360   Max.   :806.0000
```

How You May View R



How [I Hope] You Will View R

```r
z <- seq(-10, 10, 0.01)
scatterplot3js(cos(z), sin(z), z, color=rainbow(length(z)))
```

How [I Hope] You Will View R

```
forceNetwork(Links = MisLinks, Nodes = MisNodes, Source = "source",
             Target = "target", Value = "value", NodeID = "name",
             Group = "group", opacity = 0.4)
```

```r
dest <- factor(sprintf("%.2f:%.2f",flights[,3], flights[,4]))
freq <- sort(table(dest), decreasing=TRUE)
frequent_destinations <- names(freq)[1:10]
idx <- dest %in% frequent_destinations
frequent_flights <- flights[idx, ]
latlong <- unique(frequent_flights[,3:4])
earth <- system.file("images/world.jpg",  package="threejs")
globejs(img=earth, lat=latlong[,1], long=latlong[,2],
        arcs=frequent_flights, arcsHeight=0.3, arcsLwd=2,
        arcsColor="#ffff00", arcsOpacity=0.15, atmosphere=TRUE)
```

What is R?

What is R?

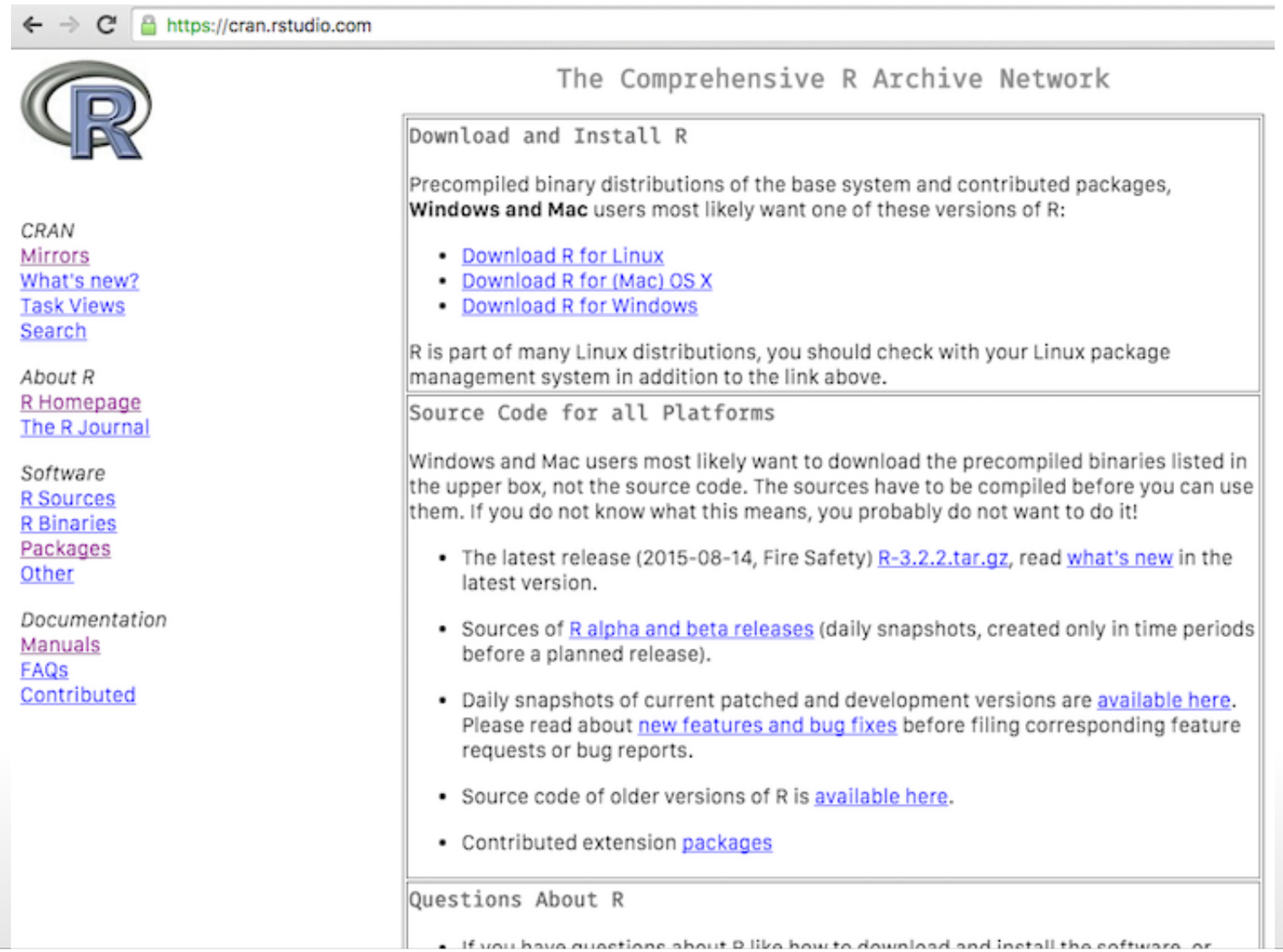
- R is a programming language
- R is statistical software
- R is an environment for interactive data analysis+visualization
- R is a community

What is R's Relationship with TGW?

- R can help you **access/acquire, clean and reformat data**
- R lets you **statistically analyze** data to find **insights**
- R enables **rapid, iterative protyping** of visualizations to help **communicate** those insights
- R helps make those steps **organized** and **repeatable/reproducible**

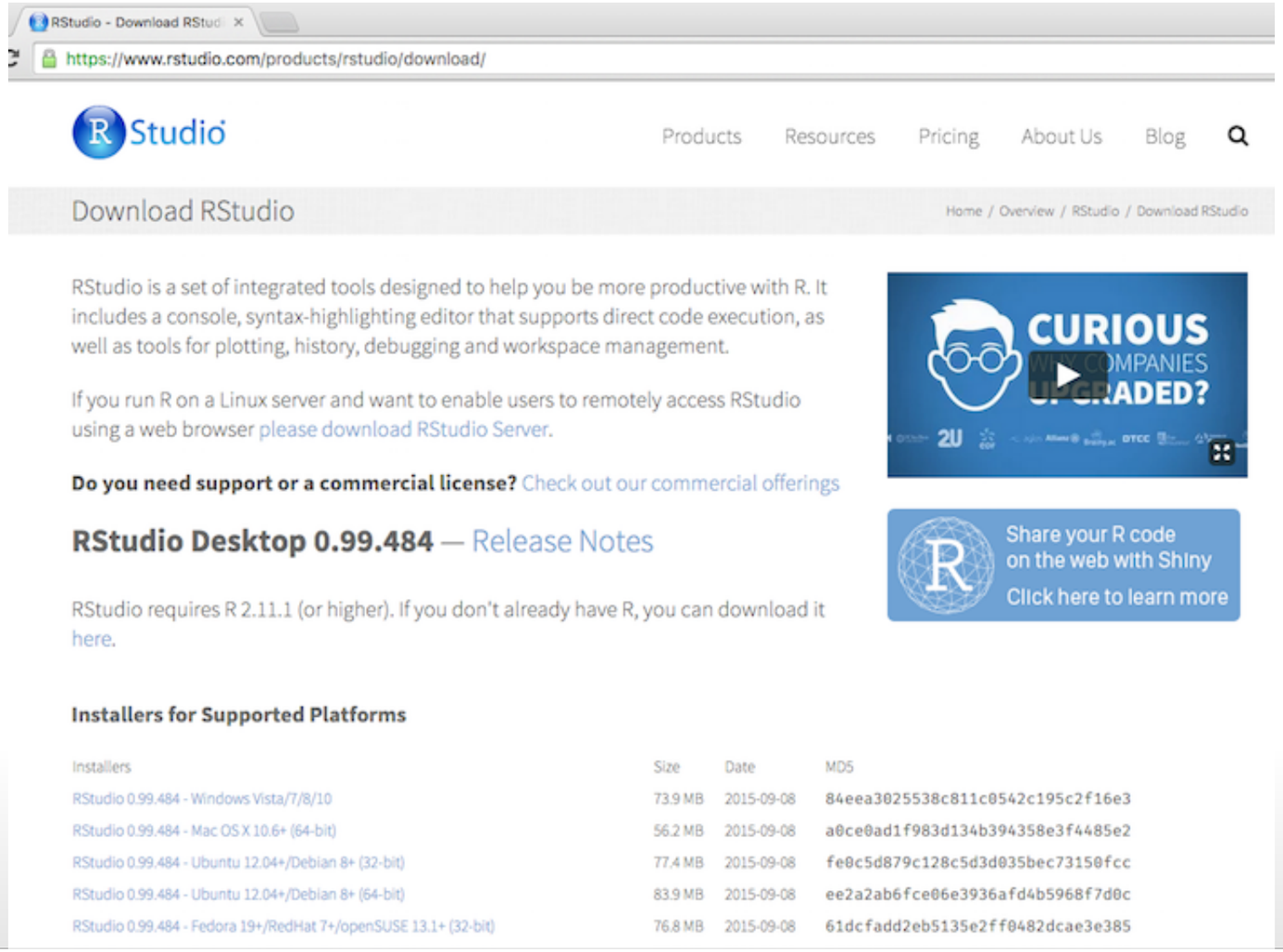
Getting Started with R

http://cran.rstudio.com

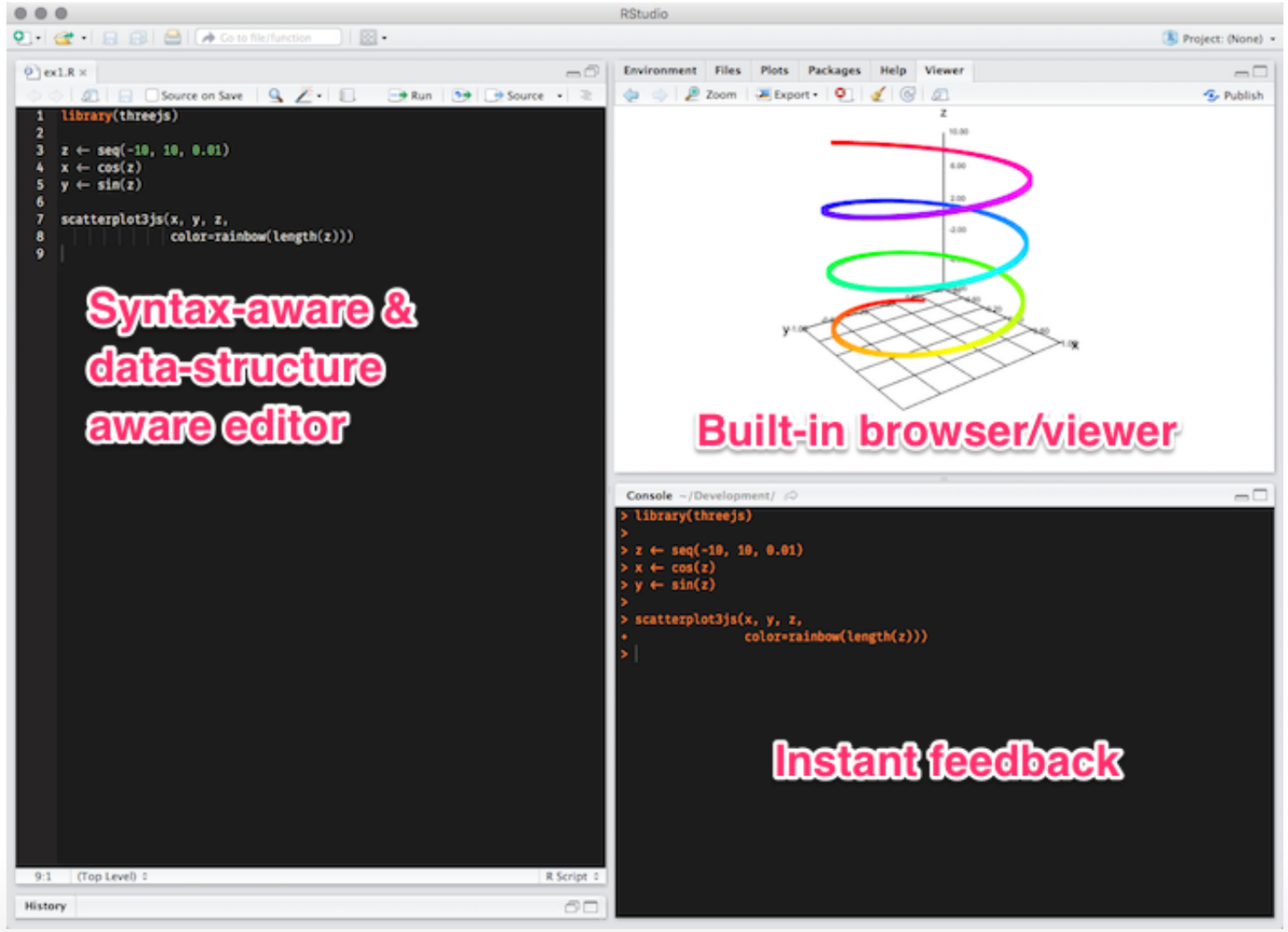


Getting Started with R/RStudio

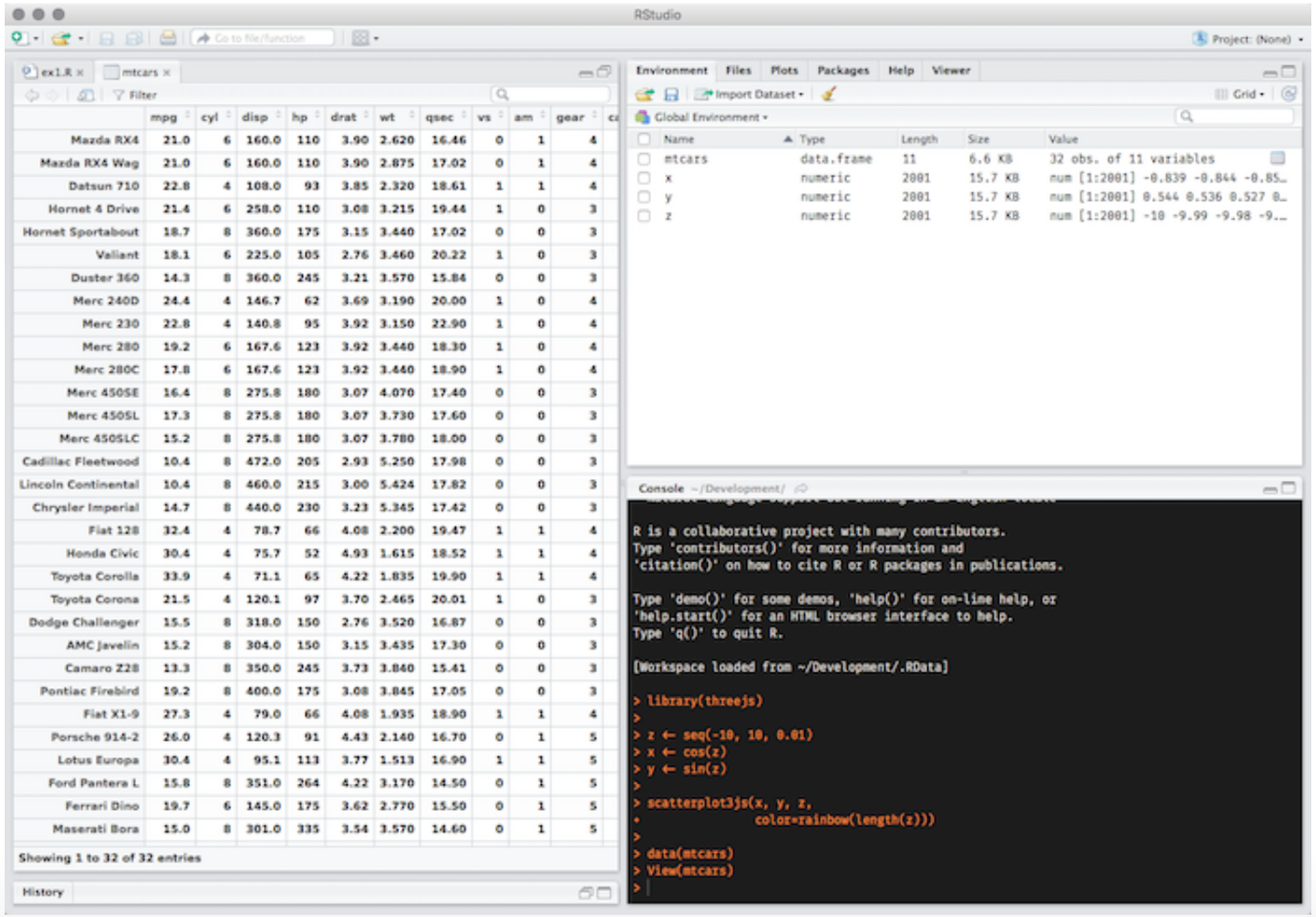
https://www.rstudio.com/products/rstudio/download/



RStudio


RStudio
ex1.R
library(threejs)
z ← seq(-10, 10, 0.01)
x ← cos(z)
y ← sin(z)
scatterplot3js(x, y, z,
color=rainbow(length(z)))
Syntax-aware &
data-structure
aware editor
Environment
Files
Plots
Packages
Help
Viewer
Built-in browser/viewer
Console
> library(threejs)
> z ← seq(-10, 10, 0.01)
> x ← cos(z)
> y ← sin(z)
> scatterplot3js(x, y, z,
+ color=rainbow(length(z)))
Instant feedback
History

RStudio


RStudio
ex1.R
mtcars
Filter
Environment
Files
Plots
Packages
Help
Viewer
Import Dataset
Grid
Global Environment
Name
Type
Length
Size
Value
mtcars data.frame 11 6.6 KB 32 obs. of 11 variables
x numeric 2001 15.7 KB num [1:2001] -0.839 -0.844 -0.85…
y numeric 2001 15.7 KB num [1:2001] 0.544 0.536 0.527 0…
z numeric 2001 15.7 KB num [1:2001] -10 -9.99 -9.98 -9…
Console ~/Development/
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/Development/.RData]
> library(threejs)
>
> z <- seq(-10, 10, 0.01)
> x <- cos(z)
> y <- sin(z)
>
> scatterplot3js(x, y, z,
+ color=rainbow(length(z)))
>
> data(mtcars)
> View(mtcars)
>
Showing 1 to 32 of 32 entries
History

R is Familiar

- Dynamic (like JavaScript & Python)
- Has variables (like JavaScript & Python)
- ...functions (like JavaScript & Python)
- ...loops (like JavaScript & Python)
- ...regular expressions (like JavaScript & Python)
- ...and, help from friends (packages) (like Node or Python modules)

R is Different

It's "vectorized" (think `map()` or `[ for ]`)

```
a <- 1:10
sum(a)
```

```
## [1] 55
```

R is Different

It *really* likes something called "data frames" (Python does too, now)

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

R is Different

It has an affintity for arcane punctuation:

```
`huh?   ` <- iris$Sepal.Length[[2]] / iris[2,1] * 3 %>% sqrt()
print(`huh?   `)
```

```
## [1] 1.732051
```

R is Different

And, complex+efficient algorithms can be confusing:

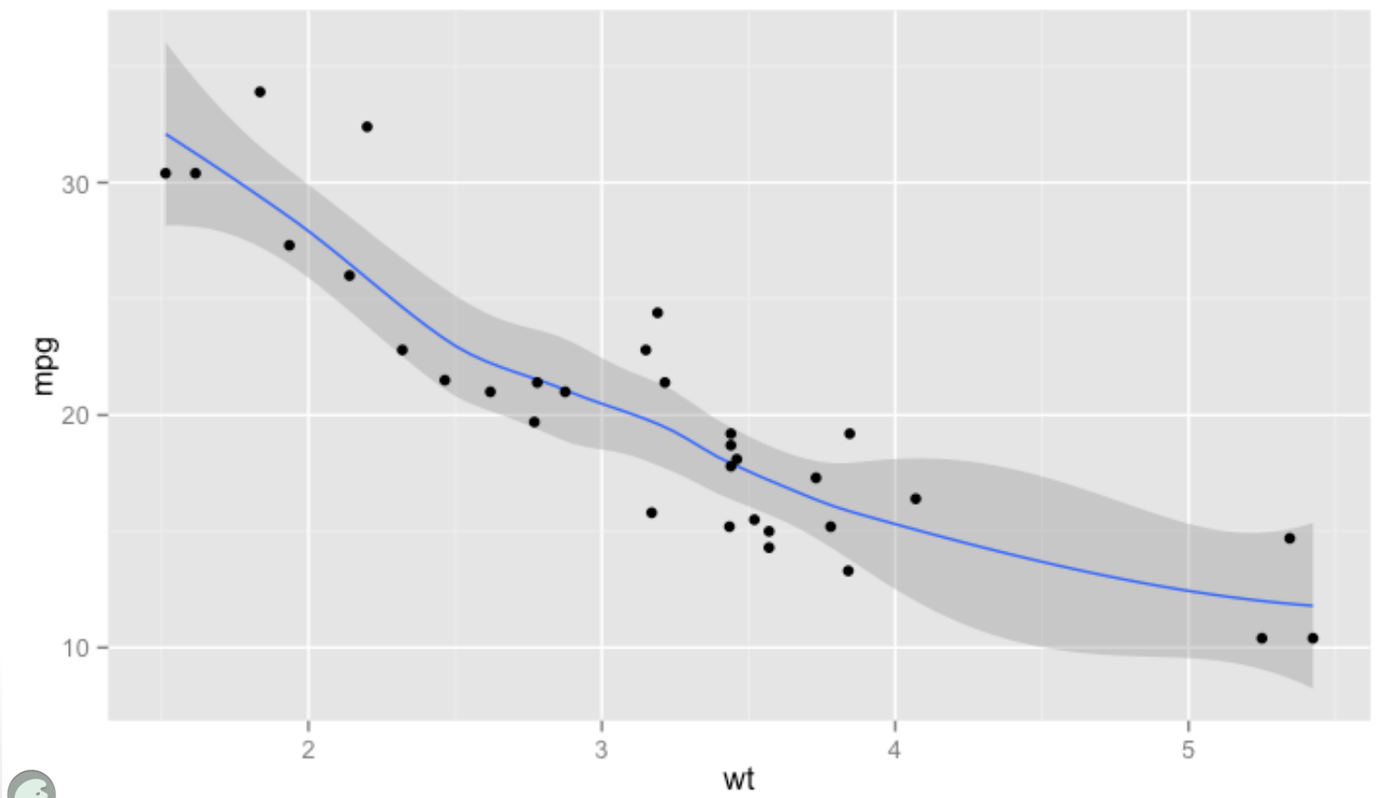
```
dat <- readLines(textConnection(" 3   weeks,   2  days,  4 hours
 4 week,  6 days,  12 hours
4 day, 3 hours
 7 hours
8  hour"))

sapply(str_split(str_trim(dat), ",[ ]*"), function(x) {
  sum(sapply(x, function(y) {
    bits <- str_split(str_trim(y), "[ ]+")[[1]]
    duration(as.numeric(bits[1]), bits[2])
  })) / 3600
})
```

```
## [1] 556 828  99   7   8
```

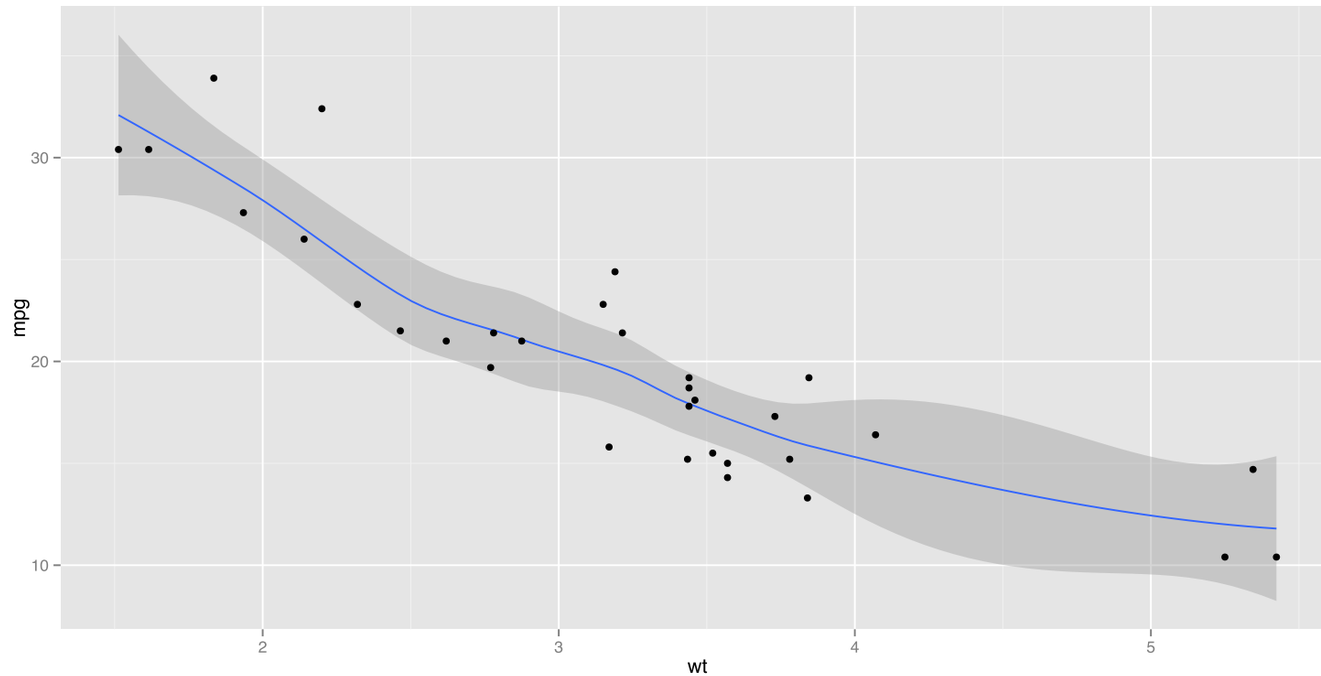
R & The Graphical Web

```
library(ggplot2)
g1 <- ggplot(mtcars, aes(x=wt, y=mpg)) + geom_smooth() + geom_point()
print(g1)
```



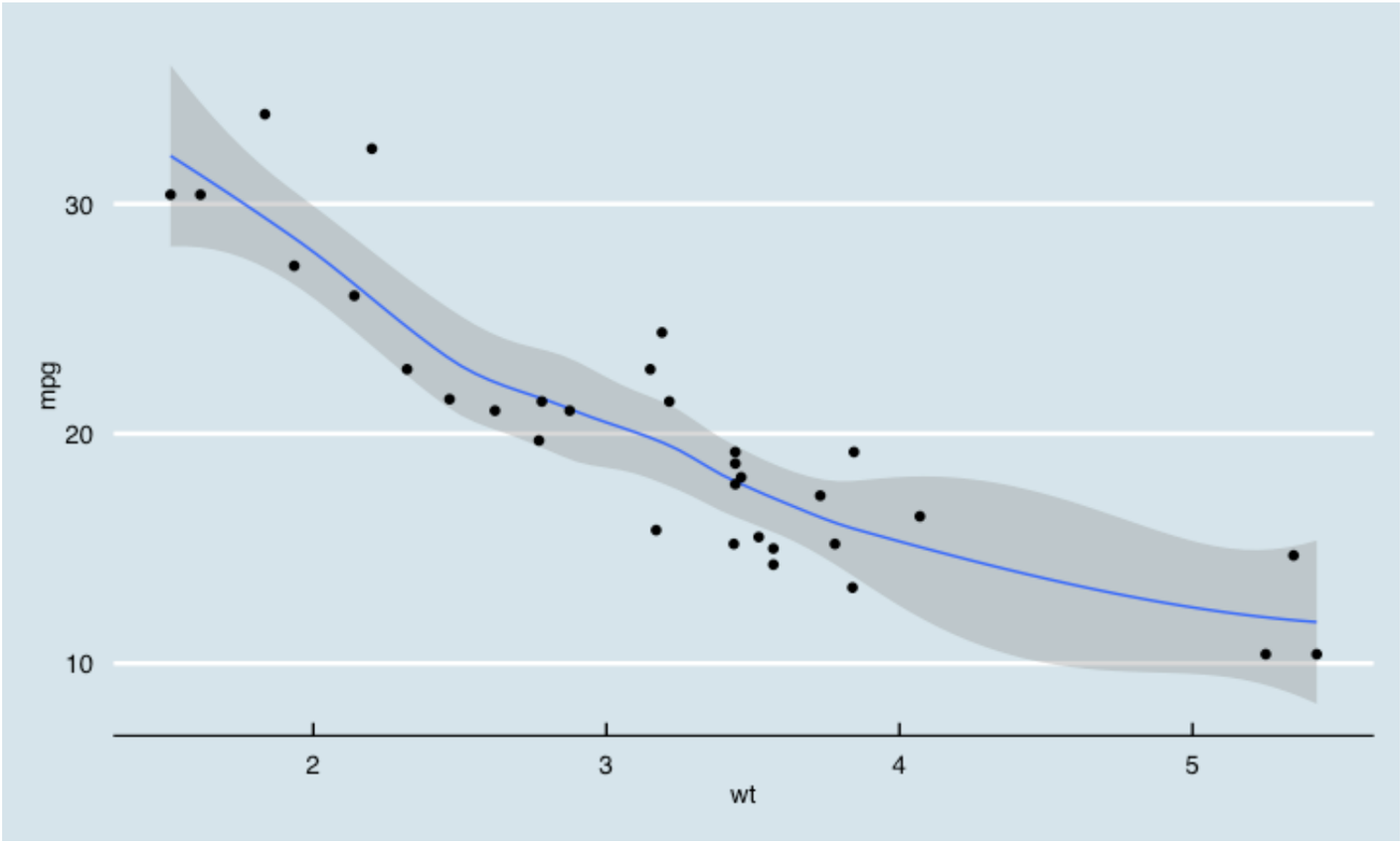
This is all it takes to turn that plot into an editable/usable SVG graphic:

```
ggsave(g1, "img/g1.svg")
```

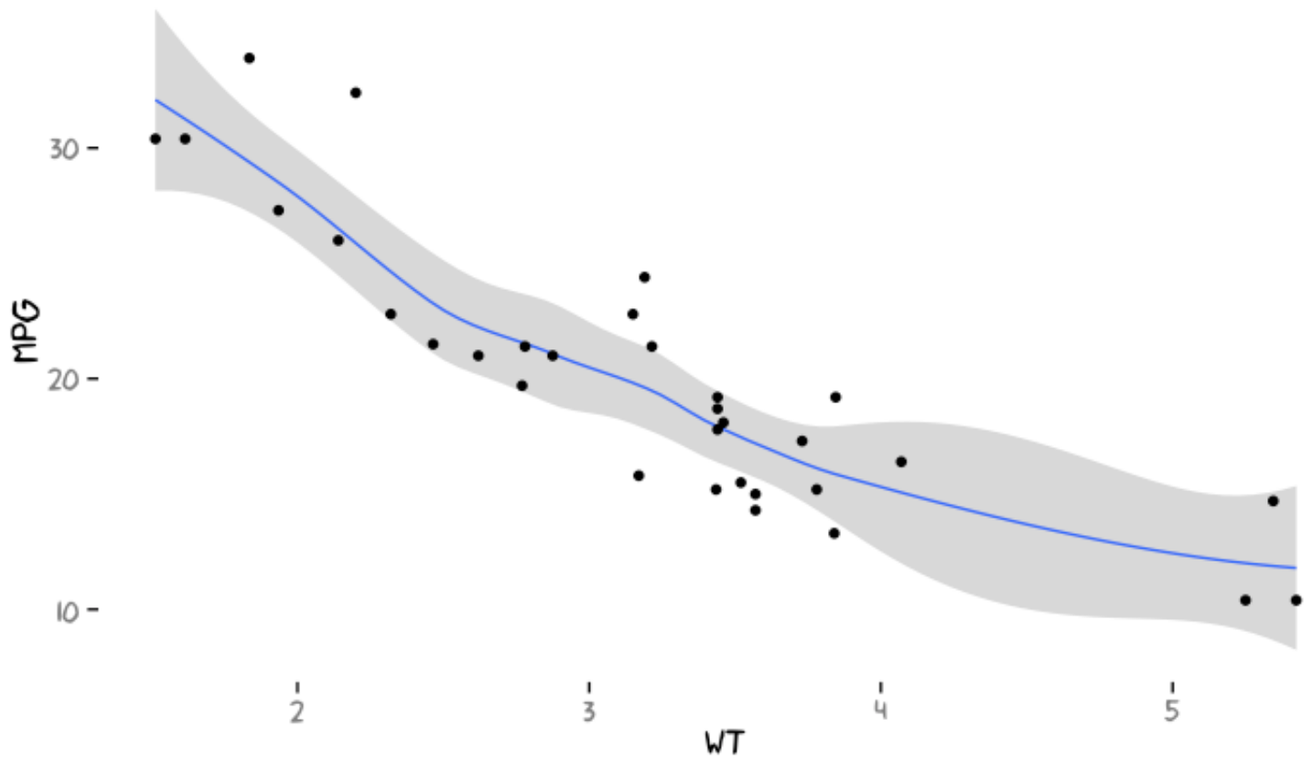


```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink"
    width="819pt" height="425pt"
    viewBox="0 0 819 425" version="1.1">
<defs>
<g>
<symbol overflow="visible" id="glyph0-0">
<path style="stroke:none;"
      d="M 0.3125 0 L 0.3125 -6.875 L 5.765625 -6.875 L 5.765625 0 Z M 4.90625 -0.859375 L 4.90
</symbol>
<symbol overflow="visible" id="glyph0-1">
<path style="stroke:none;"
      d="M 0.921875 -4.75 L 0.921875 -5.390625 C 1.523438 -5.453125 1.945312 -5.550781 2.1875 -
```

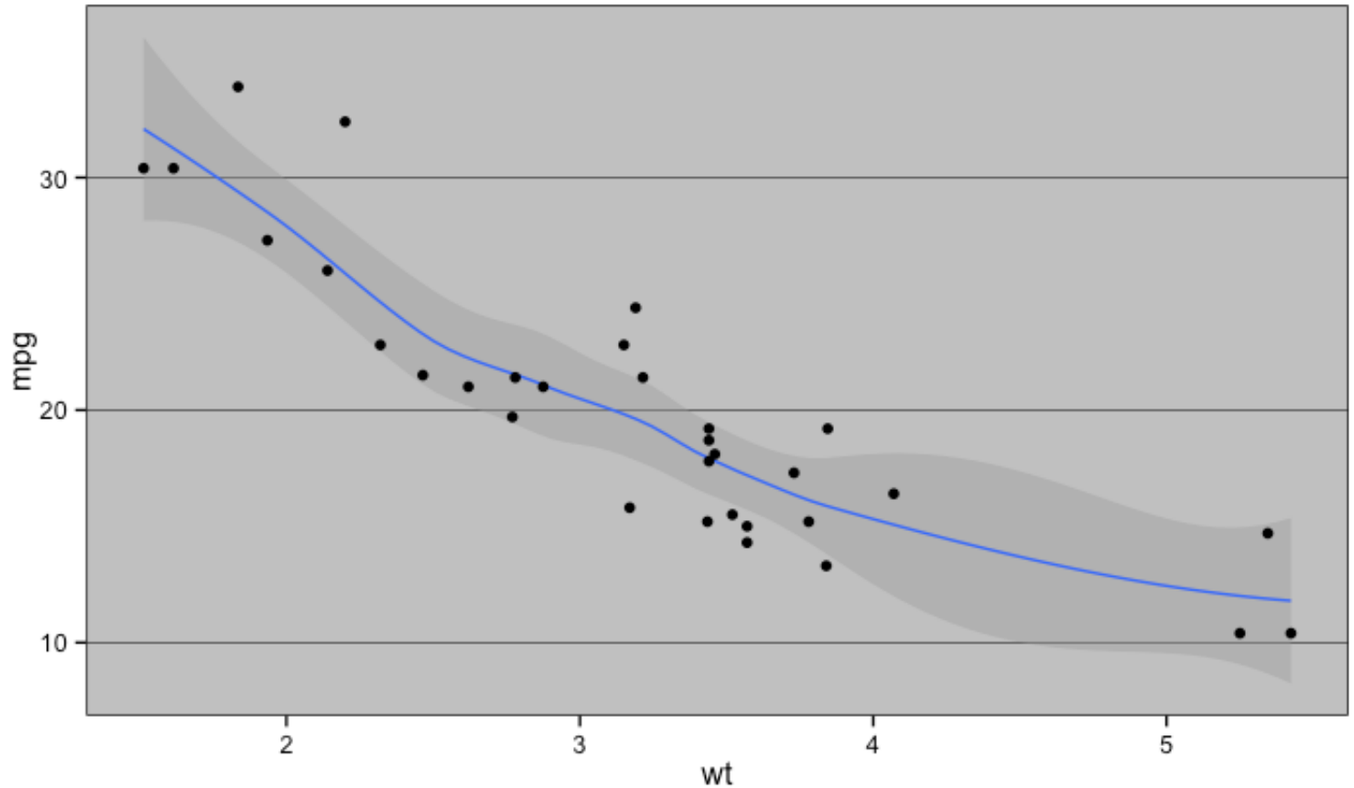
```
library(ggthemes)
g1 + theme_economist()
```



```
library(xkcd)
g1 + theme_xkcd()
```



```
g1 + theme_excel()
```



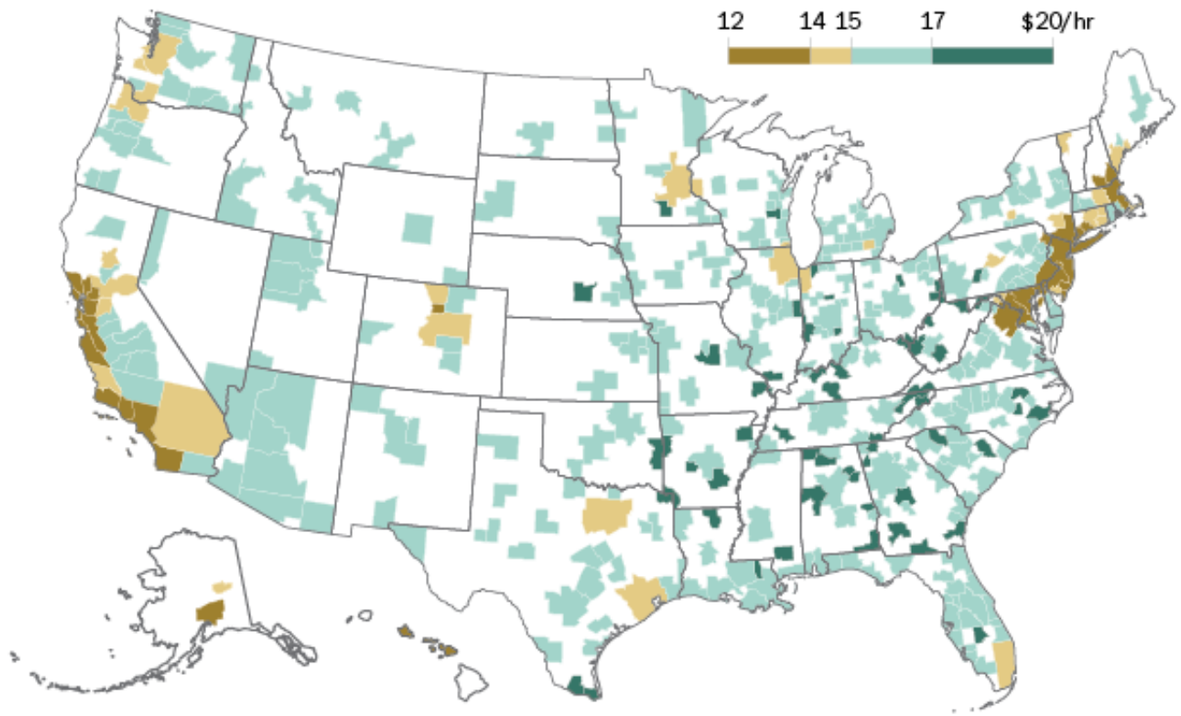
Doing Real Work

Doing Real Work

http://bit.ly/pewmapdemo+

Where Paychecks Stretch the Most, and Least

Estimated real purchasing power of a national $15 hourly wage, by metropolitan area

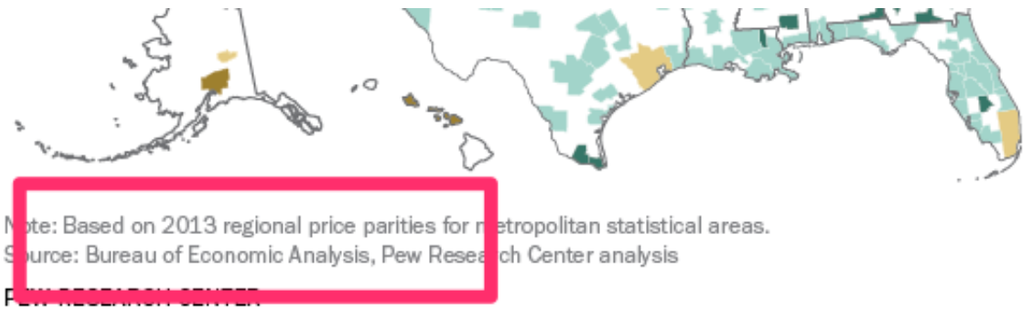


Note: Based on 2013 regional price parities for metropolitan statistical areas.
Source: Bureau of Economic Analysis, Pew Research Center analysis

PEW RESEARCH CENTER

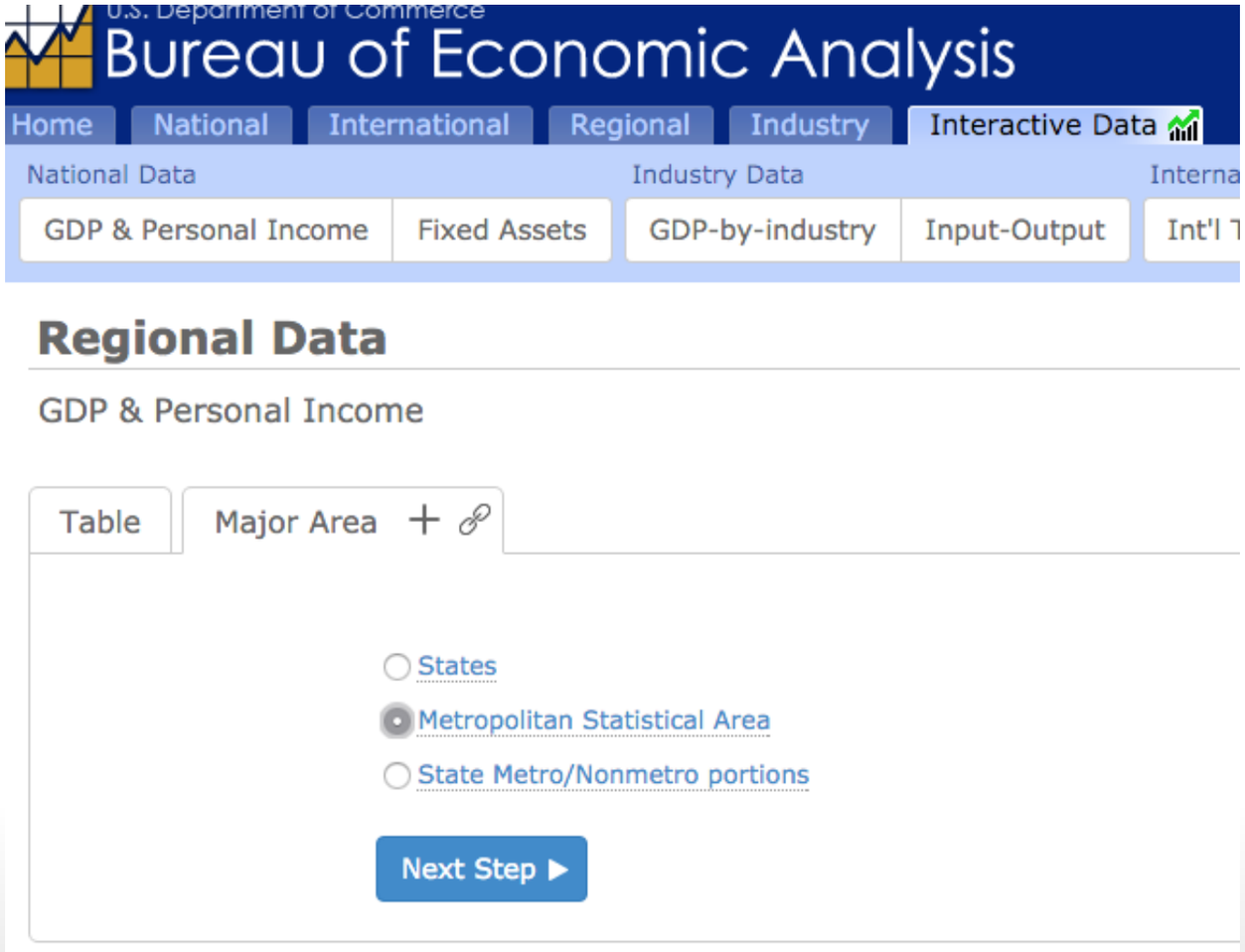
Doing Real Work

Let's find this data!



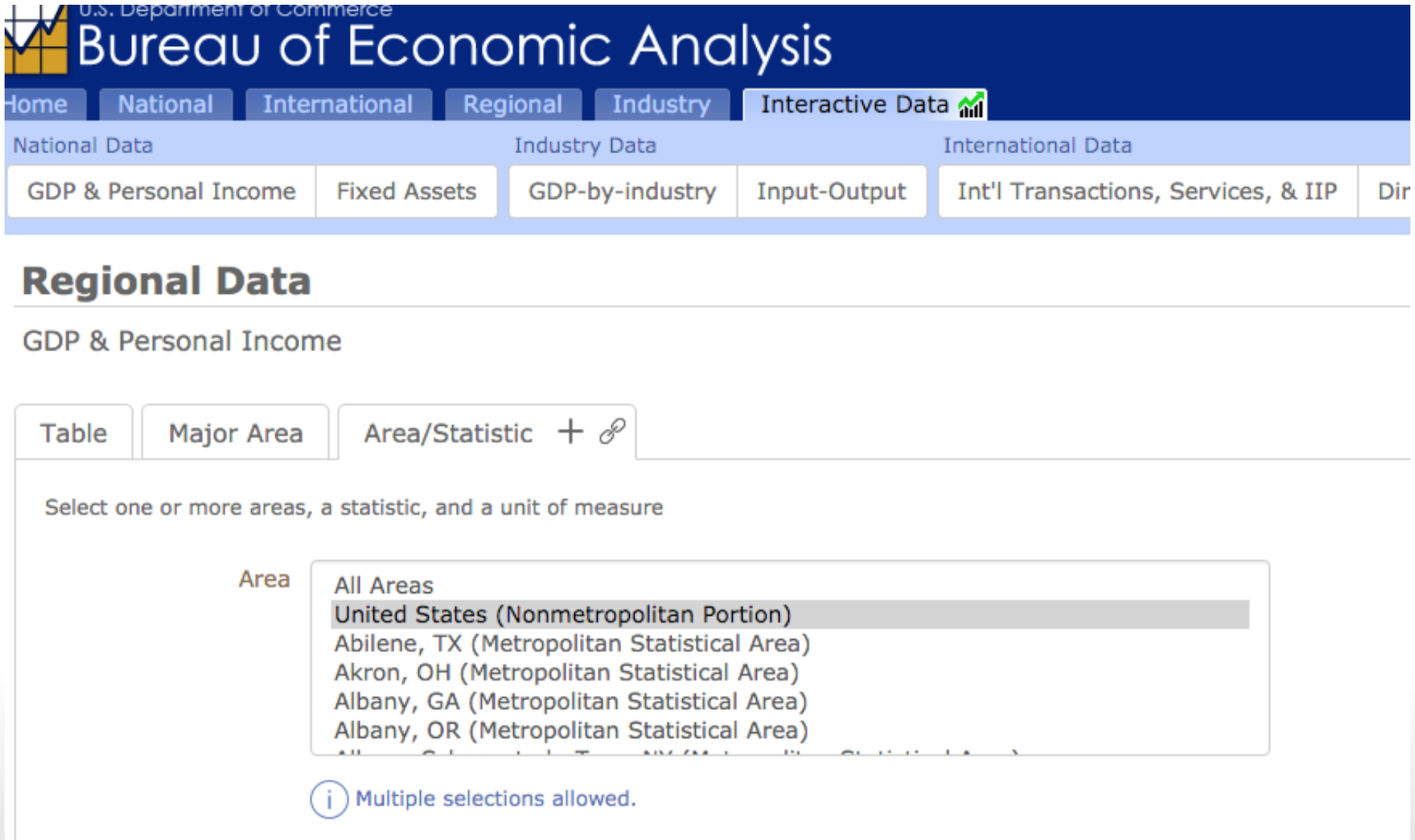
Doing Real Work

Let's find this data!



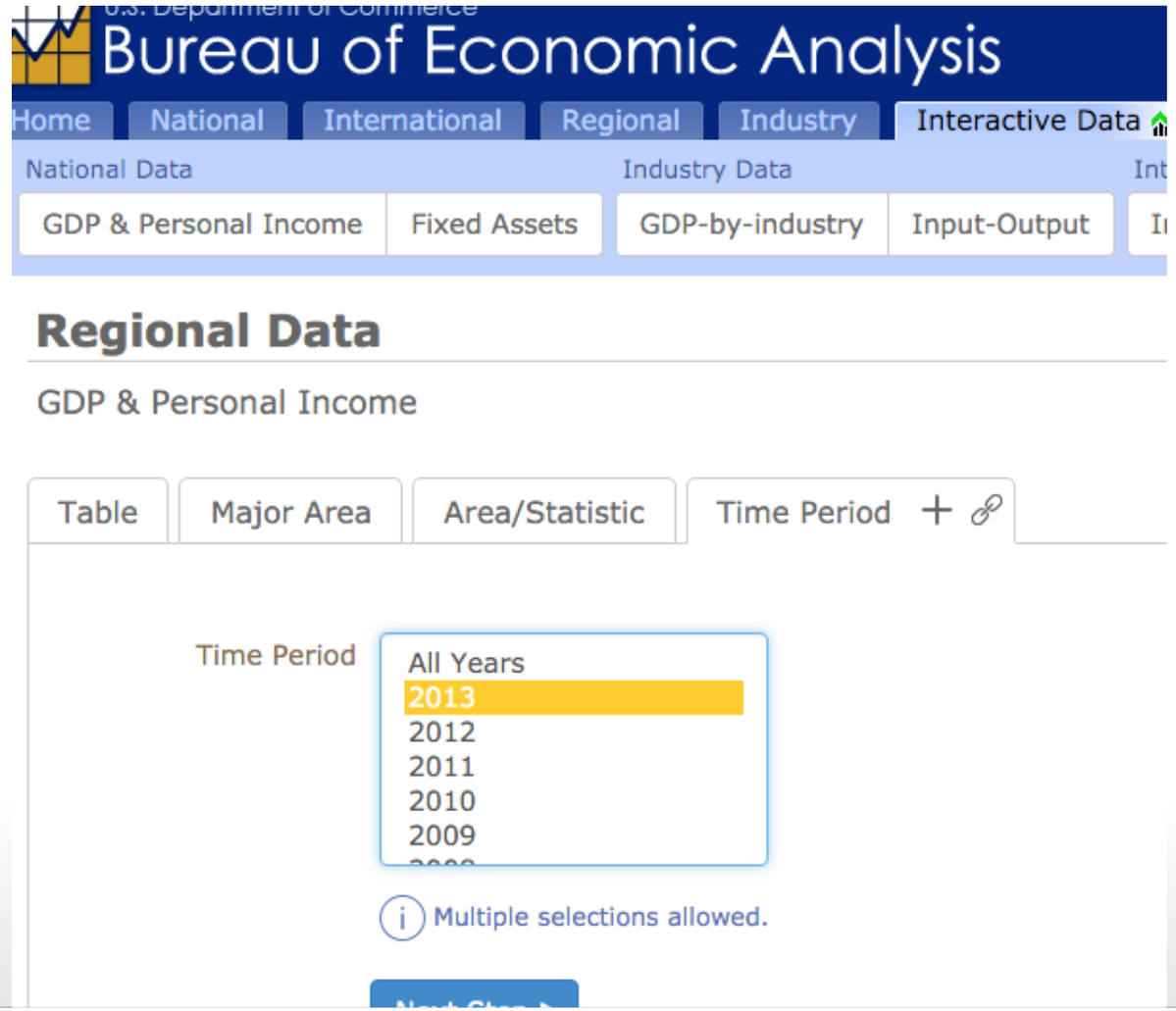
Doing Real Work

Let's find this data!



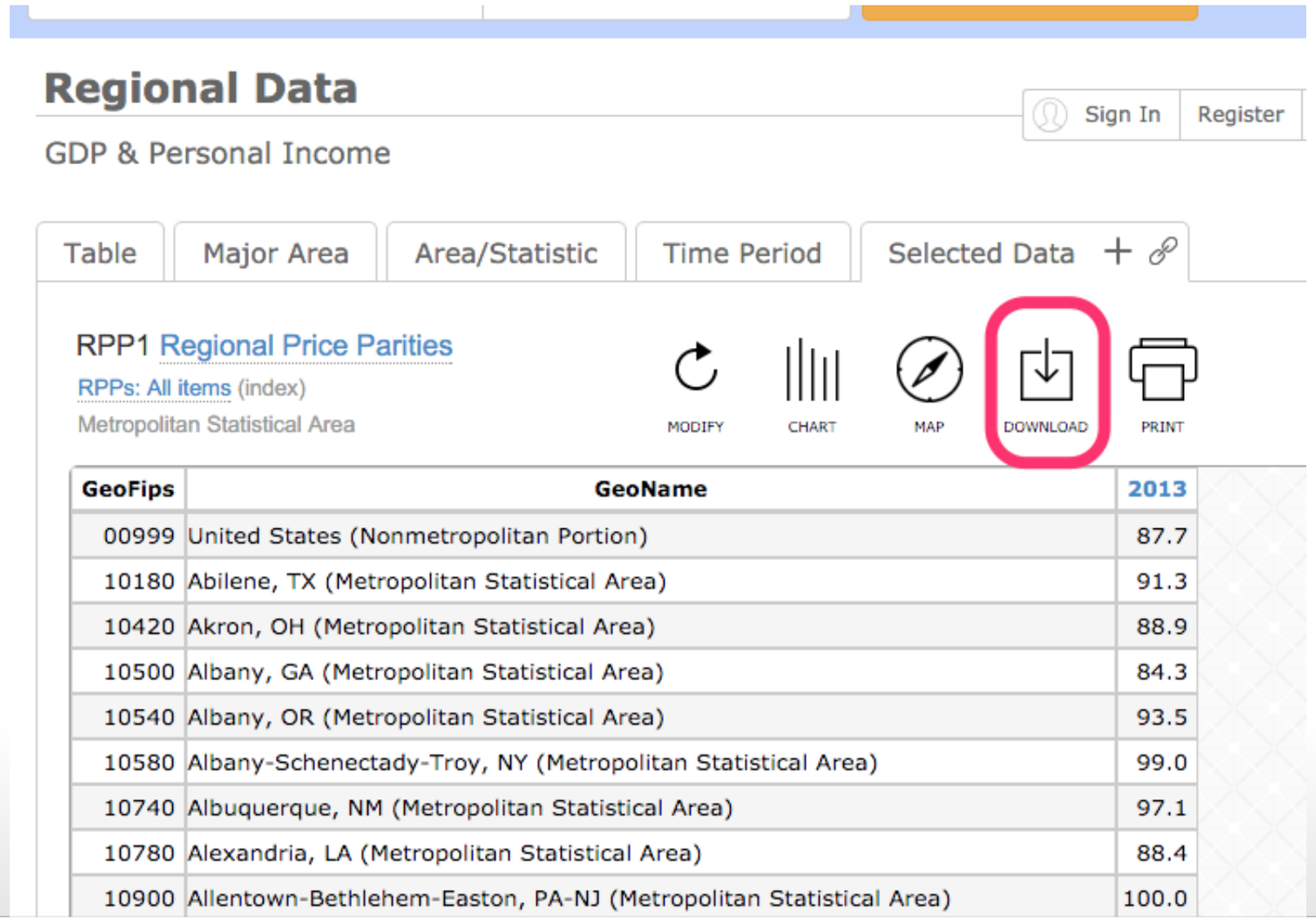
Doing Real Work

Let's find this data!



Doing Real Work

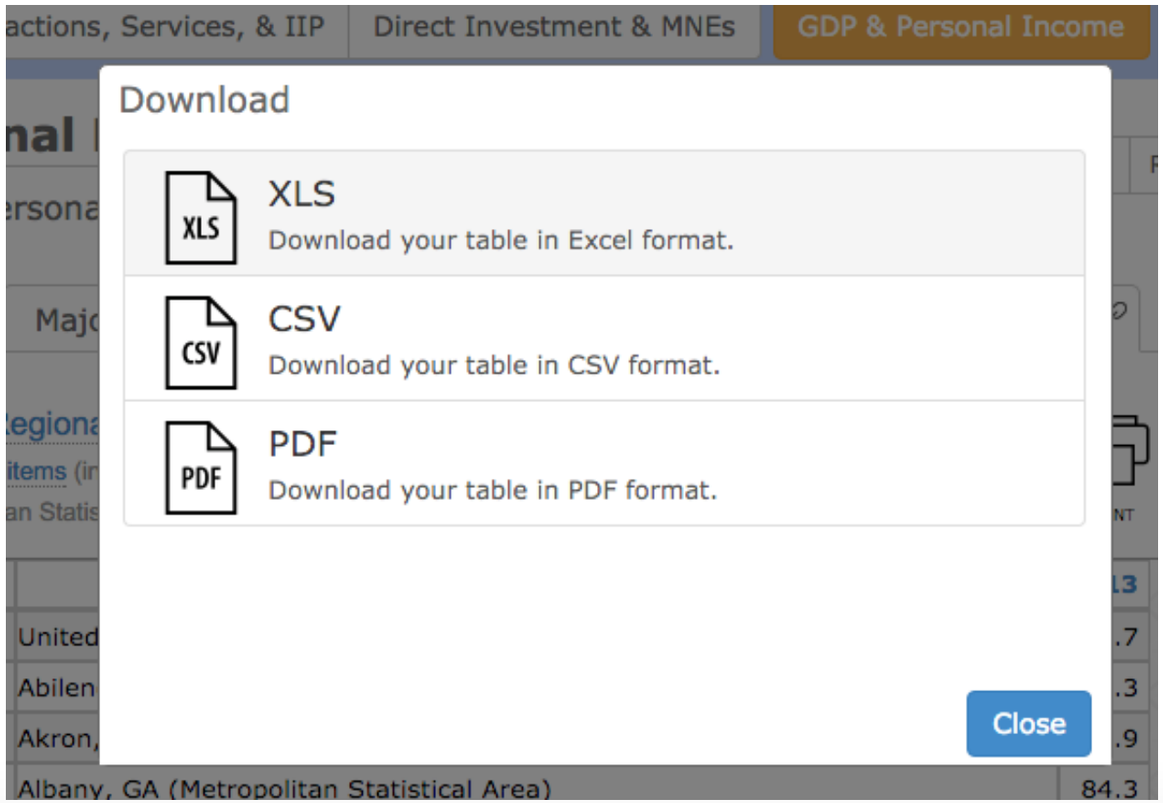
Almost done!



GeoFips	GeoName	2013
00999	United States (Nonmetropolitan Portion)	87.7
10180	Abilene, TX (Metropolitan Statistical Area)	91.3
10420	Akron, OH (Metropolitan Statistical Area)	88.9
10500	Albany, GA (Metropolitan Statistical Area)	84.3
10540	Albany, OR (Metropolitan Statistical Area)	93.5
10580	Albany-Schenectady-Troy, NY (Metropolitan Statistical Area)	99.0
10740	Albuquerque, NM (Metropolitan Statistical Area)	97.1
10780	Alexandria, LA (Metropolitan Statistical Area)	88.4
10900	Allentown-Bethlehem-Easton, PA-NJ (Metropolitan Statistical Area)	100.0

Doing Real Work

Done! *(kinda)*



Doing Real Work

Done! *(kinda)*

RPP1 Regional Price Parities		
RPPs: All items (index)		
Bureau of Economic Analysis		
Metropolitan Statistical Area		
GeoFips	GeoName	2013
00999	United States (Nonmetropolitan Portion)	87.7
10180	Abilene, TX (Metropolitan Statistical Area)	91.3
10420	Akron, OH (Metropolitan Statistical Area)	88.9
10500	Albany, GA (Metropolitan Statistical Area)	84.3
10540	Albany, OR (Metropolitan Statistical Area)	93.5
10580	Albany-Schenectady-Troy, NY (Metropolitan Statistical Area)	99.0

```r
dat <- read.csv("data/download.csv", skip=4, header=TRUE, stringsAsFactors=FALSE)
dat <- head(dat, -2)
head(dat)
```

```
##   GeoFips                                                  GeoName
## 1   00999                     United States (Nonmetropolitan Portion)
## 2   10180                  Abilene, TX (Metropolitan Statistical Area)
## 3   10420                    Akron, OH (Metropolitan Statistical Area)
## 4   10500                   Albany, GA (Metropolitan Statistical Area)
## 5   10540                   Albany, OR (Metropolitan Statistical Area)
## 6   10580 Albany-Schenectady-Troy, NY (Metropolitan Statistical Area)
##   X2013
## 1  87.7
## 2  91.3
## 3  88.9
## 4  84.3
## 5  93.5
## 6  99.0
```

```r
dat$GeoName <- gsub(" \\(Metropolitan Statistical Area\\)", "",
                    dat$GeoName)
dat$GeoFips <- sprintf("%05d", as.numeric(dat$GeoFips))
head(dat)
```

```
##   GeoFips                                GeoName X2013
## 1   00999 United States (Nonmetropolitan Portion)  87.7
## 2   10180                            Abilene, TX  91.3
## 3   10420                              Akron, OH  88.9
## 4   10500                             Albany, GA  84.3
## 5   10540                             Albany, OR  93.5
## 6   10580             Albany-Schenectady-Troy, NY  99.0
```

Doing Real Work

www.bea.gov/API/bea_web_service_api_user_guide.htm#tabs-1

Appendix A - RegionalData (statistics by state, county, and MSA)

The new datasets RegionalIncome and RegionalProduct have more statistics and industry detail than the RegionalData dataset. See Appendices I and J. Although RegionalData is still valid, we encourage users to switch to the more comprenhensive datasets RegionalIncome and RegionalProduct.

The RegionalData dataset contains estimates from the Regional Economic Accounts. These include estimates of GDP by state and metropolitan area; estimates of personal income and employment by state, metropolitan area, and county; and regional price parities by state and MSA.

RegionalData Request Parameters

Parameter Name	Type	Description	Required	Multiple Values Accepted	"All" value	Default
KeyCode	String	The code for the statistic requested	Yes	No		
GeoFips	String	The state, county or MSA code	No	Yes	STATE or COUNTY or MSA	STATE
Year	String	Year requested	No	Yes	ALL	ALL

Doing Real Work

http://bea.gov/api/data/?UserID=xxxxxx-xxxx-xxxx-xxxx-xxxxxxxx&
method=GetData&datasetname=RegionalData&KeyCode=RPPALL_MI&
Year=2013&&ResultFormat=json"


```
64          "Ordinal": "7",
65          "Name": "DataValue",
66          "DataType": "numeric",
67          "IsValue": "1"
68        }],
69        "Data": [{
70          "GeoFips": "10180",
71          "GeoName": "Abilene, TX (Metropolitan Statistical Area)",
72          "Code": "RPPALL_MI",
73          "TimePeriod": "2013",
74          "CL_UNIT": "IDX",
75          "UNIT_MULT": "0",
76          "DataValue": "91.3"
77        }, {
78          "GeoFips": "10420",
79          "GeoName": "Akron, OH (Metropolitan Statistical Area)",
80          "Code": "RPPALL_MI",
81          "TimePeriod": "2013",
82          "CL_UNIT": "IDX",
83          "UNIT_MULT": "0",
84          "DataValue": "88.9"
85        }, {
86          "GeoFips": "10500",
87          "GeoName": "Albany, GA (Metropolitan Statistical Area)",
88          "Code": "RPPALL_MI",
89          "TimePeriod": "2013",
90          "CL_UNIT": "IDX",
```

Doing Real Work

```
library(jsonlite)
dat <- readJSON("that horrible URL")
```

```
dat <- dat$BEAAPI$Results$Data
dat$X2013 <- as.numeric(dat$DataValue)
dat$GeoName <- gsub(" \\(Metropolitan Statistical Area\\)", "",
                    dat$GeoName)
dat$GeoFips <- sprintf("%05d", as.numeric(dat$GeoFips))
head(dat[,c(1,2,8)])
```

```
##   GeoFips                     GeoName X2013
## 1   10180                Abilene, TX  91.3
## 2   10420                  Akron, OH  88.9
## 3   10500                 Albany, GA  84.3
## 4   10540                 Albany, OR  93.5
## 5   10580 Albany-Schenectady-Troy, NY  99.0
## 6   10740             Albuquerque, NM  97.1
```

Doing Real Work

```
library(httr)
response <- GET("http://bea.gov/api/data/",
                query=list(
                  UserID=Sys.getenv("BEA_API_TOKEN"),
                  method="GetData",
                  datasetname="RegionalData",
                  KeyCode="RPPALL_MI",
                  Year="2013",
                  ResultFormat="json"
                ))
dat <- fromJSON(content(response, as="text"))
```

Same cleanup as we did in the raw URL version

```
dat <- dat$BEAAPI$Results$Data
dat$X2013 <- as.numeric(dat$DataValue)
dat$GeoName <- gsub(" \\(Metropolitan Statistical Area\\)", "",
                    dat$GeoName)
dat$GeoFips <- sprintf("%05d", as.numeric(dat$GeoFips))
head(dat[,c(1,2,8)])
```

```
##   GeoFips                     GeoName X2013
## 1   10180                Abilene, TX  91.3
## 2   10420                  Akron, OH  88.9
## 3   10500                 Albany, GA  84.3
## 4   10540                 Albany, OR  93.5
## 5   10580 Albany-Schenectady-Troy, NY  99.0
## 6   10740             Albuquerque, NM  97.1
```

Doing Real Work

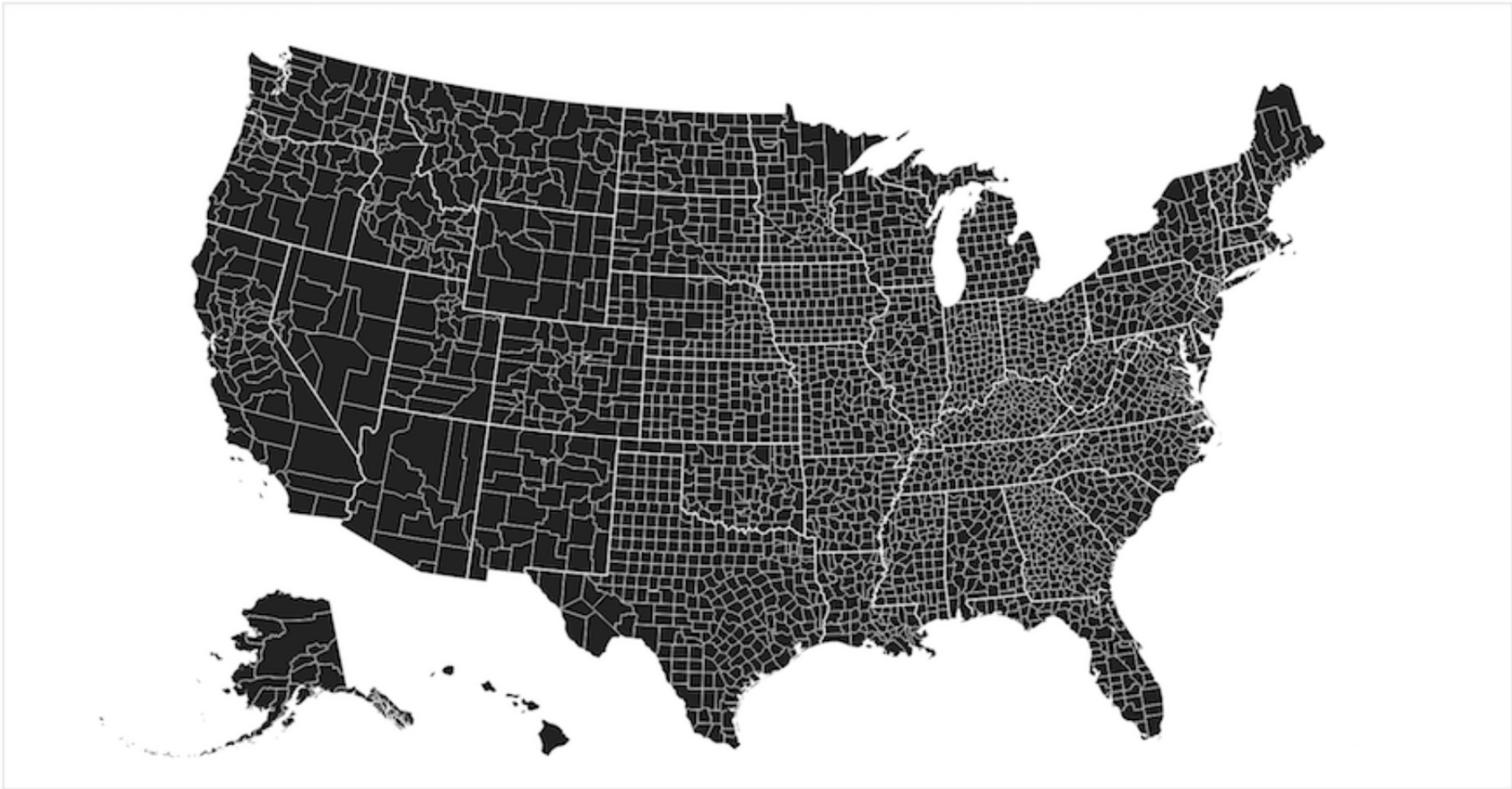
```
library(noncensus)
data(counties)
xlate <- data.frame(fipscounty=sprintf("%s%s", counties$state_fips,
                                        counties$county_fips),
                    cbsa=counties$CBSA,
                    stringsAsFactors=FALSE)
dat <- merge(dat[,c(1,2,8)], xlate[,c("cbsa", "fipscounty")],
             by.x="GeoFips", by.y="cbsa")
head(dat)
```

```
##   GeoFips     GeoName X2013 fipscounty
## 1   10180 Abilene, TX  91.3      48441
## 2   10180 Abilene, TX  91.3      48253
## 3   10180 Abilene, TX  91.3      48059
## 4   10420   Akron, OH  88.9      39133
## 5   10420   Akron, OH  88.9      39153
## 6   10500  Albany, GA  84.3      13177
```

Doing Real Work

mbostock's block #4090848 November 16, 2012

U.S. States TopoJSON



What's the Plan?

- Display a US map county choropleth with the counties filled according the RPP value
- The counties are not all represented and we don't need a billion small polygons left over so we'll outline the states for context
- We'd like to add contextual information via popup to show the discrete data value and the name of the metro area
- A legend would be good

Options (without R)

- Raw JS or jQuery + CSS
- Straight D3
- Kartograph
- Datamaps
- Leaflet [Ex: http://leafletjs.com/examples/choropleth-example.html]
  (Perfect data. ~170 lines)

Getting Work Done : R + Leaflet

Get map data

```
library(rgdal)
URL <- "http://bl.ocks.org/mbostock/raw/4090846/us.json"
fil <- sprintf("data/%s", basename(URL))
if (!file.exists(fil)) download.file(URL, fil)

# read state borders from the file
states <- readOGR(fil, "states", stringsAsFactors=FALSE,
                  verbose=FALSE)

# read county borders from the file
county <- readOGR(fil, "counties", stringsAsFactors=FALSE,
                  verbose=FALSE)
```

Getting Work Done : R + Leaflet

We don't want to display all the counties, so we'll subtract out the ones that aren't in our data set.

```
rpp_counties <- subset(county, id %in% dat$fipscounty)
rpp_counties <- merge(rpp_counties, dat,
                      by.x="id",
                      by.y="fipscounty",
                      all.x=TRUE)
```

Getting Work Done : R + Leaflet

We need to setup the color scale (really similar to how you'd do it in JS)

```
library(leaflet)
pal <- colorBin("BrBG", range(rpp_counties$X2013), bins=5)
rpp_counties$color <- pal(rpp_counties$X2013)
```

Getting Work Done : R + Leaflet

```
leaflet() %>%
  addProviderTiles("Acetate.terrain") %>%
  addPolygons(data=rpp_counties, weight=0.25,
              fillColor=~color, color="black", fillOpacity=1,
              popup=~sprintf("In %s, <span style='font-weight:700'>%s</span> has the purchasing
                             htmlEscape(GeoName),
                             htmlEscape(dollar(X2013)))) %>%
  addPolygons(data=states, weight=0.5, fillColor="white", fillOpacity=0, fill=FALSE, color="#52
  addLegend(position="bottomright", pal=pal, values=rpp_counties$X2013, labFormat=labelFormat("
  setView(-74.0059, 40.7127, 6)
```

Where Paychecks Stretch The Most/Least

Doing Real Work

~170 lines pure leaflet/javascript

vs

~60 lines of R

...and the R version can be instantly used to get new BEA data sets where the leaflet one "cheated" and merged the data prior to the HTML example.

A Bit More About Getting Data In

Getting Data Into R

General

- built-in support for CSV/TSV/general delimited & fixed-width
- `readr` / `rio` faster & more robust compatibility
- `readxl` (and others) for raw Excel reading
- `googlesheets`
- `data.table` (large data)
- numerous packages to read statistical data files

Getting Data Into R

Web Scraping / API

- `httr` (like curl command line but better)
- `rvest` (more structured web page scraping)
- `jsonlite` (JSON)
- `XML` / `xml2` (XML)
- `Rselenium` (headless browser & DOM scraping)
- `V8` (the V8 engine in R)

Getting Data Into R

Database

- `dplyr`
- `RPostgreSQL`
- `RMySQL`
- `rredis`
- `mongolite`
- `RSQLite`

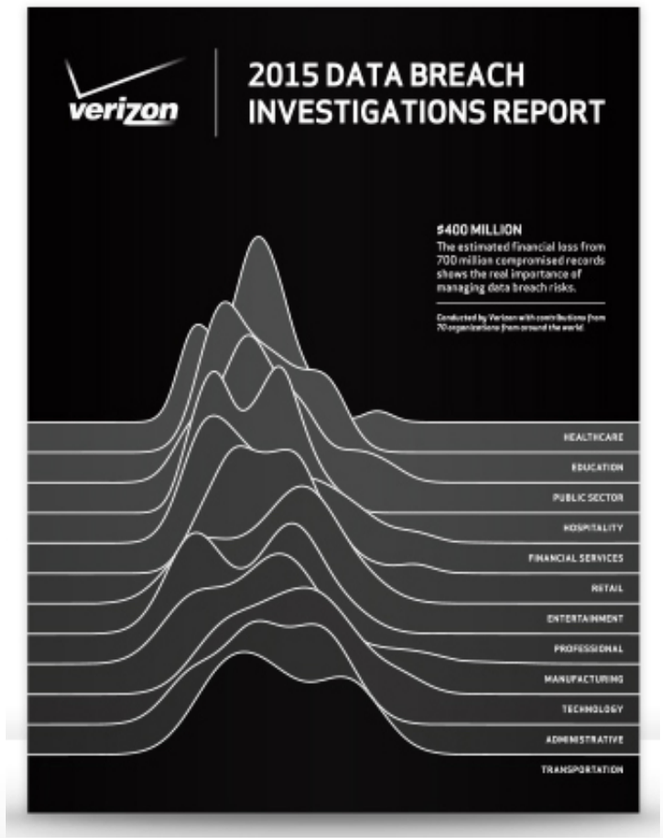
Getting Data Into R

- Almost every useful public API covered
- Virtually every "big data" store including AWS/S3
- If something is missing, complain on Twitter and there'll be a package in a week

Crunching Stats

Real World R : Crunching Stats

http://www.verizonenterprise.com/DBIR/2015/



Real World R : Crunching Stats

200,000 incidents/breaches

~3,000 data elements per record

~150 lines of statistical analysis

Real World R : Crunching Stats

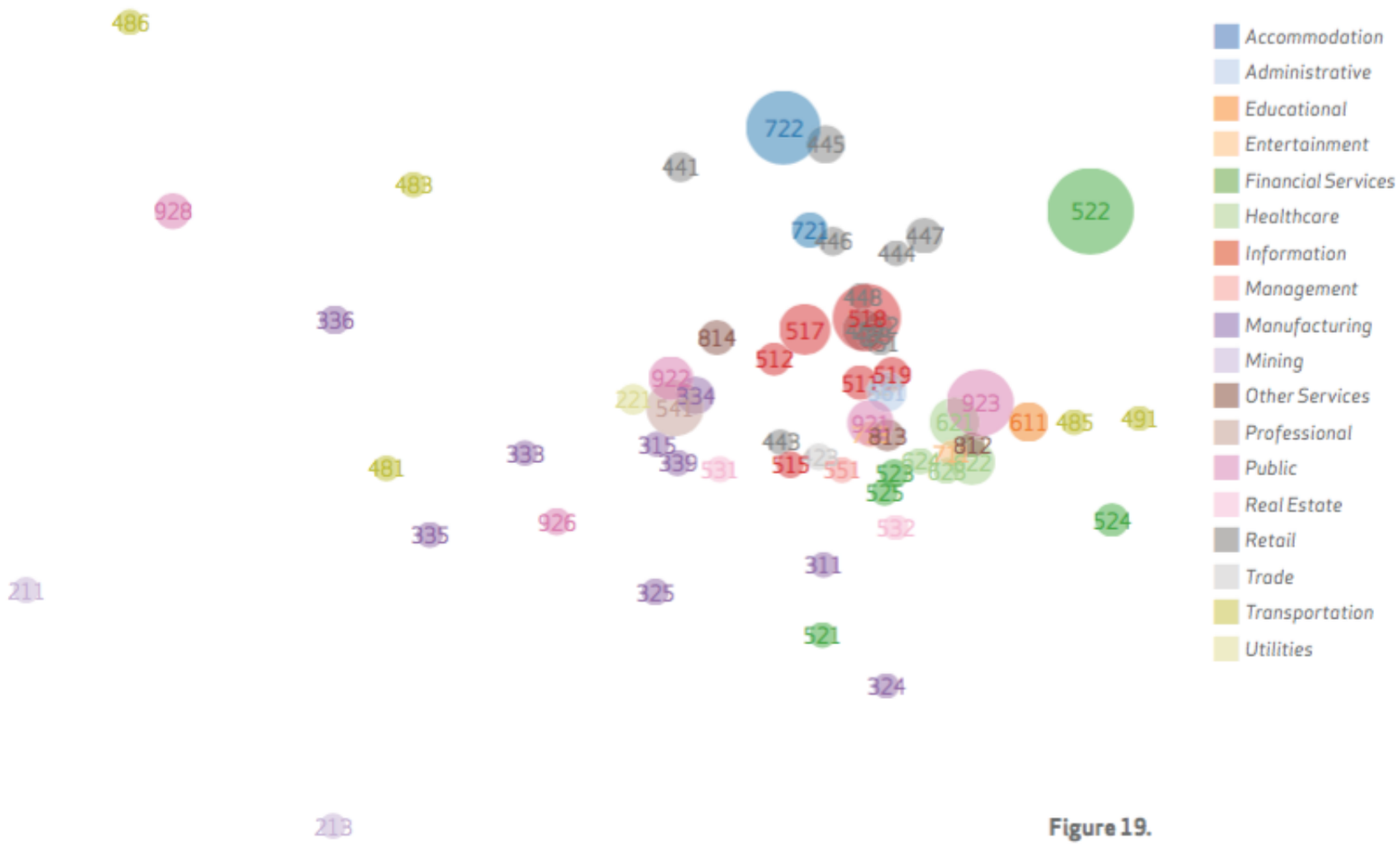


Figure 19.

Clustering on breach data across industries

28 To look up the three-digit NAICS codes, visit: census.gov/eos/www/naics/index.html

Real World R : Crunching Stats

http://vz-risk.github.io/dbir/2015/19/

Getting Data Out of R

Getting Data Out of R

- files (CSV/JSON/XML)
- database write
- S3 upload
- API "put"

Getting Data Out of R

OpenCPU

Hello World! Basic JSON RPC

```
curl https://public.opencpu.org/ocpu/library/stats/R/rnorm/json \
-H "Content-Type: application/json" -d '{"n":3, "mean": 10, "sd":10}'

[4.9829, 6.3104, 11.411]
```

This maps to the following request

```
#library(jsonlite)
args <- fromJSON('{"n":3, "mean": 10, "sd":10}')
output <- do.call(stats::rnorm, args)
toJSON(output)
```

Which is equivalent to this function call

```
rnorm(n=3, mean=10, sd=10)
```

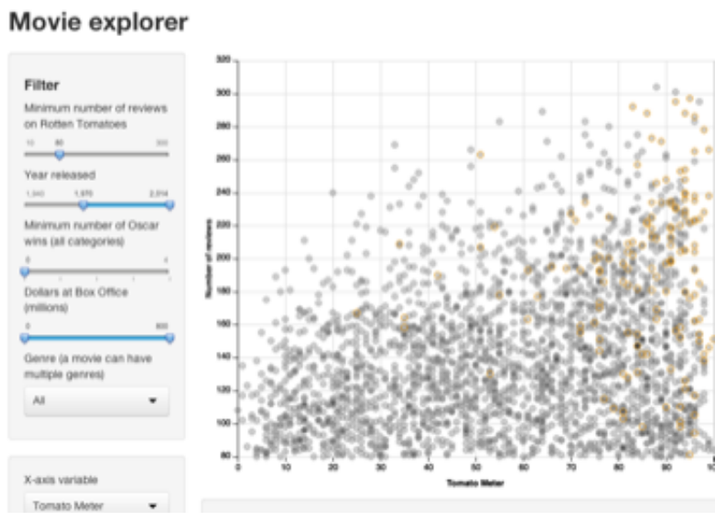
Getting Data Out of R

- `plumber` http://plumber.trestletech.com/ (think "Flask")
- `httpuv` https://github.com/rstudio/httpuv (basic web & websocket server)

Getting Data and Visualizations Out of R

Getting Data and Visualizations Out of R

Shiny http://shiny.rstudio.com/



http://shiny.rstudio.com/gallery/movie-explorer.html

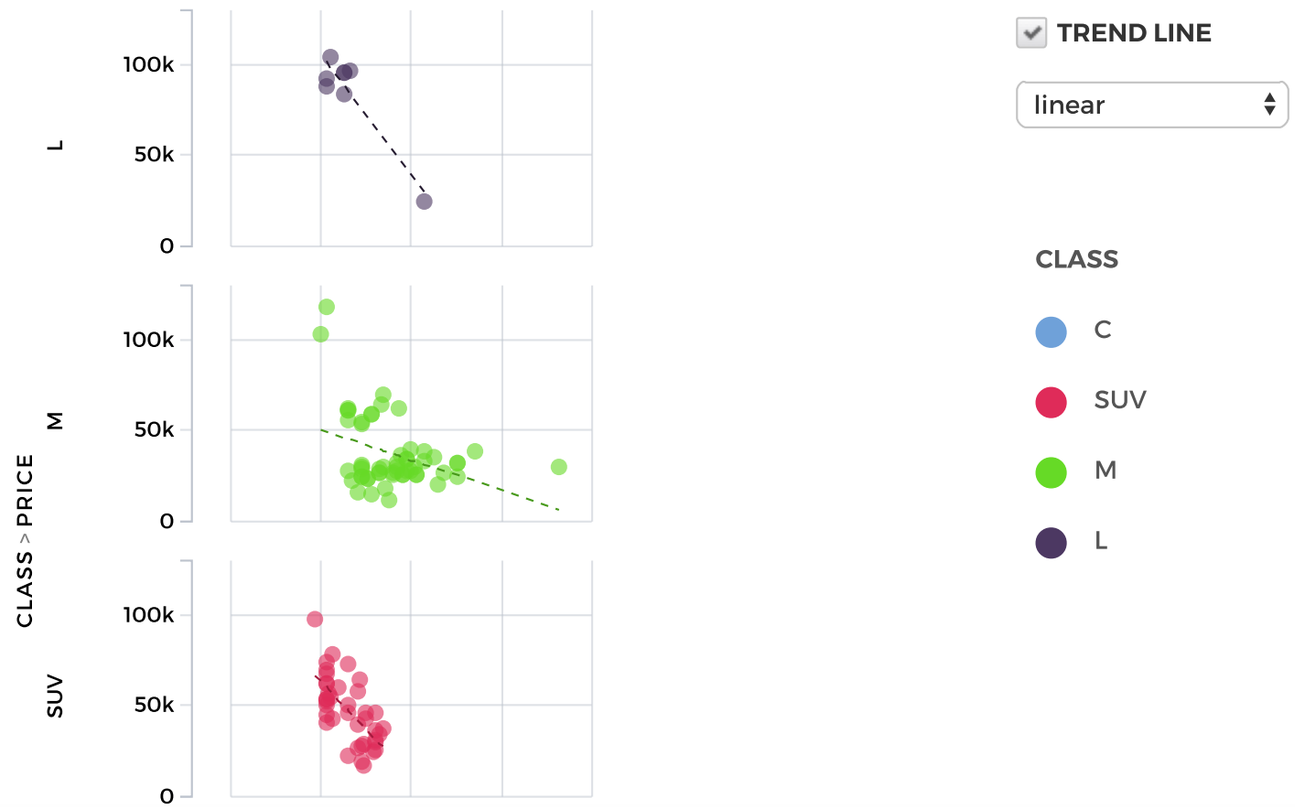
HTML Widgets

- `htmlwidgets` http://www.htmlwidgets.org/
- The widget gallery http://hafen.github.io/htmlwidgetsgallery/ (i've got 3!)
- You've already seen one! (`leaflet`)

htmlwidgets

```
library(taucharts)
data(cars_data)
tauchart(cars_data) %>%
  tau_point("milespergallon", c("class", "price"), color="class") %>%
  tau_trendline() %>% tau_legend()
```

htmlwidgets



htmlwidgets

- `devtools::create("/path/to/new/package")`
- `setwd("/path/to/new/package")`
- **or use RStudio**
- `htmlwidgets::scaffoldWidget()`

htmlwidgets

- `devtools::build()`
- `devtools::install()`
- **or use RStudio**

htmlwidgets

```
library(widgety)
widgety("Hello, world!")
```