TRUMAN STATE UNIVERSITY

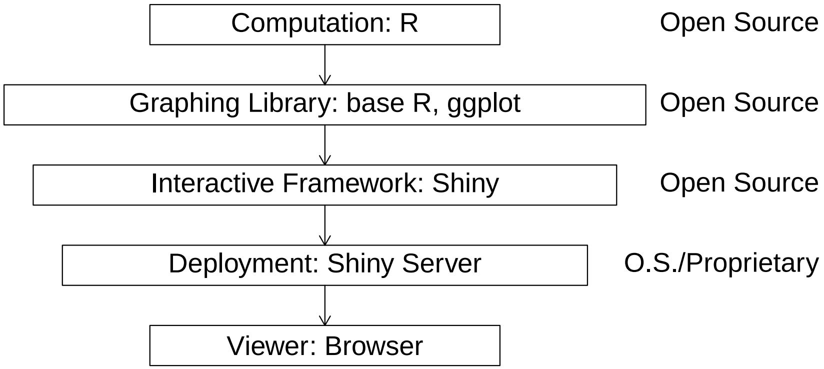
Department of STATISTICS

The Plotly Library

DISTINCT BY DESIGN

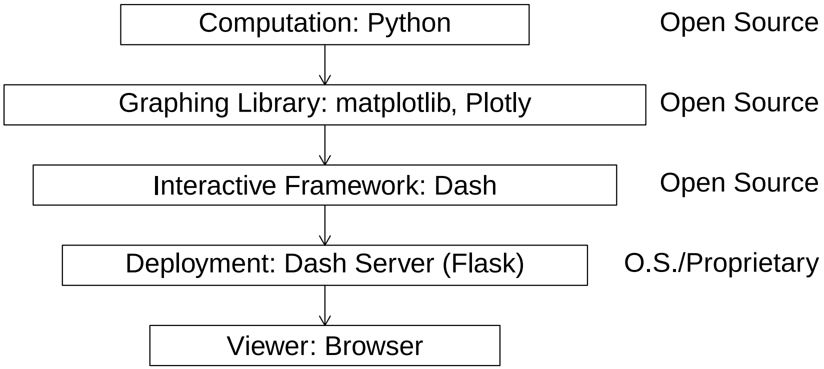
Code Ecosystems

What we've seen so far…



…is not the only game in town.

Another solution is built around Python (or R).


Computation: Python
Open Source
Graphing Library: matplotlib, Plotly
Open Source
Interactive Framework: Dash
Open Source
Deployment: Dash Server (Flask)
O.S./Proprietary
Viewer: Browser

R integration is provided by the dash package.

Concepts from one system are often transportable to new systems.

Dash coding looks similar to what you've learned about Shiny.

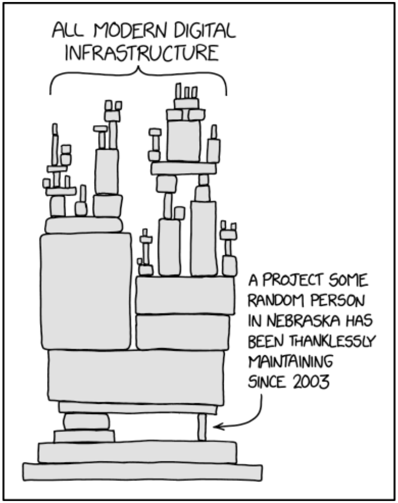
```r
# From dashr.plotly.com/basic-callbac
app <- Dash$new()

app$layout(
  htmlDiv(
    list(
      dccInput(id='my-id',
               value='initial value',
               type='text'),
      htmlDiv(id='my-div')
    )
  )
)
```

```r
app$callback(
  output=list(id='my-div',
              property='children'),
  params=list(input(id='my-id',
              property='value')),
  function(input_value) {
    sprintf("You've entered \"%s\"",
            input_value)
  })

app$run_server()
```

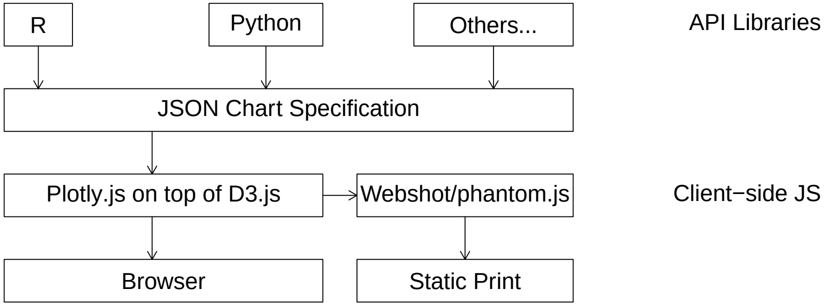
Code dependency can be a code security issue.


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

- Rage-quit: Coder unpublished 17 lines of JavaScript and "broke the Internet" (arstechnica.com)
- The Internet Is Being Protected By Two Guys Named Steve (buzzfeed.com)
- Tech giants, chastened by Heartbleed, finally agree to fund OpenSSL (arstechnica.com)
- New type of supply-chain attack hit Apple, Microsoft and 33 other companies (arstechnica.com)

Plotly

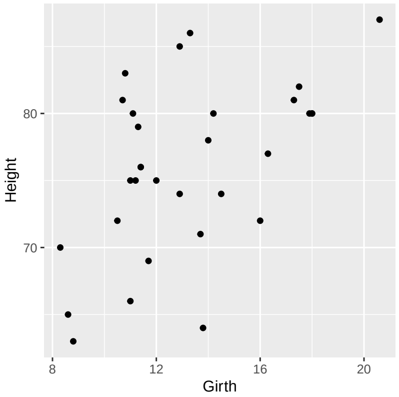
Plotly is a language-agnostic web graphics library.



See plotly.com/r/ and plotly-r.com/index.html.

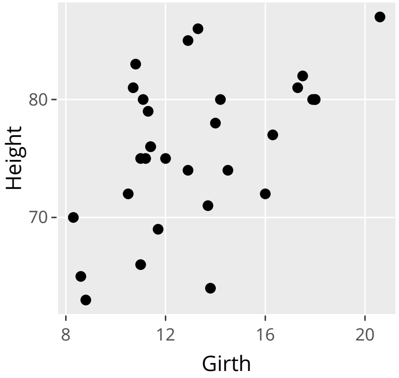
A jump-start: `ggplotly` converts ggplot graphs into plotly graphs.

```r
library(tidyverse)
library(plotly)
p <- ggplot(trees) +
  geom_point(aes(x=Girth, y=Height))
p
```



A jump-start: `ggplotly` converts ggplot graphs into plotly graphs.

```r
library(tidyverse)
library(plotly)
p <- ggplot(trees) +
  geom_point(aes(x=Girth, y=Height))
ggplotly(p)
```



Plotly “thinks” similarly to ggplot.

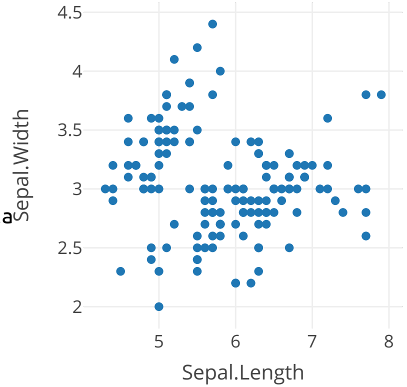
Concept	ggplot	plotly
Initiate a Plot	`ggplot()`	`plot_ly()`
Layers	`layer()` or `geom_` …	`add_trace()` or `add_` …
Mapping to a Variable	`aes(x=xvar)`	`x = ~xvar`
Mapping to a Constant	`x = 1` (outside aes)	`x = 1`
Inheritance	Mappings in `ggplot` inherited by `geom_`	Mappings in `plot_ly` inherited by `add_`
Connecting the Pieces	Use the “+” operator.	Use the “%>%” operator.

- `plot_ly()` tries to guess an appropriate graph form, similar to base-R `plot`, but unlike `ggplot`.
- `ggplot` seems to offer more flexibility, while `plotly` appears to be more limited to preset trace types.

On its own, `plot_ly` acts similarly to base-R `plot`.

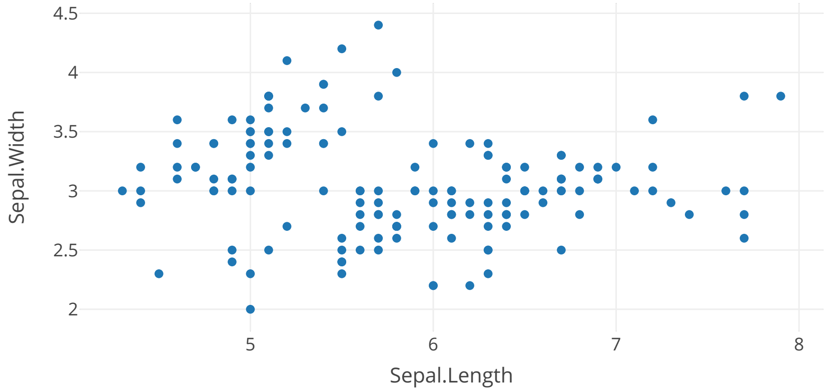
```r
# The `plot_ly` command creates
# a new graph.
plot_ly(data=iris, x=~Sepal.Length,
        y=~Sepal.Width)
```

```
## No trace type specified:
##   Based on info supplied,
##   a 'scatter' trace seems appropria
## No scatter mode specifed:
##   Setting the mode to markers
```



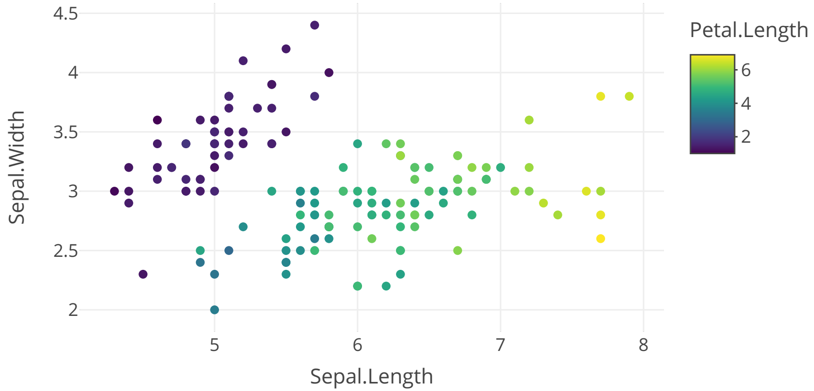
Graphs can be assigned to an object, and printed.

```r
p <- plot_ly(data=iris, x=~Sepal.Length, y=~Sepal.Width)
# print(p) also works.
p
```



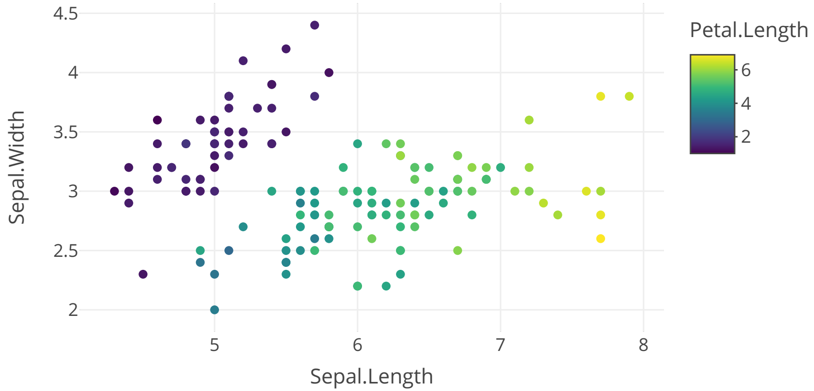
The add_ commands take a plotly object and return an ammended plotly object.

```r
p <- plot_ly(data=iris, x=~Sepal.Length, y=~Sepal.Width)
p <- add_markers(p, color=~Petal.Length)
p
```



It’s easier to think in terms of pipes.

```r
p <- plot_ly(data=iris, x=~Sepal.Length, y=~Sepal.Width) %>%
       add_markers(color=~Petal.Length, text=~Species)
p
```


Petal.Length
Sepal.Width
Sepal.Length