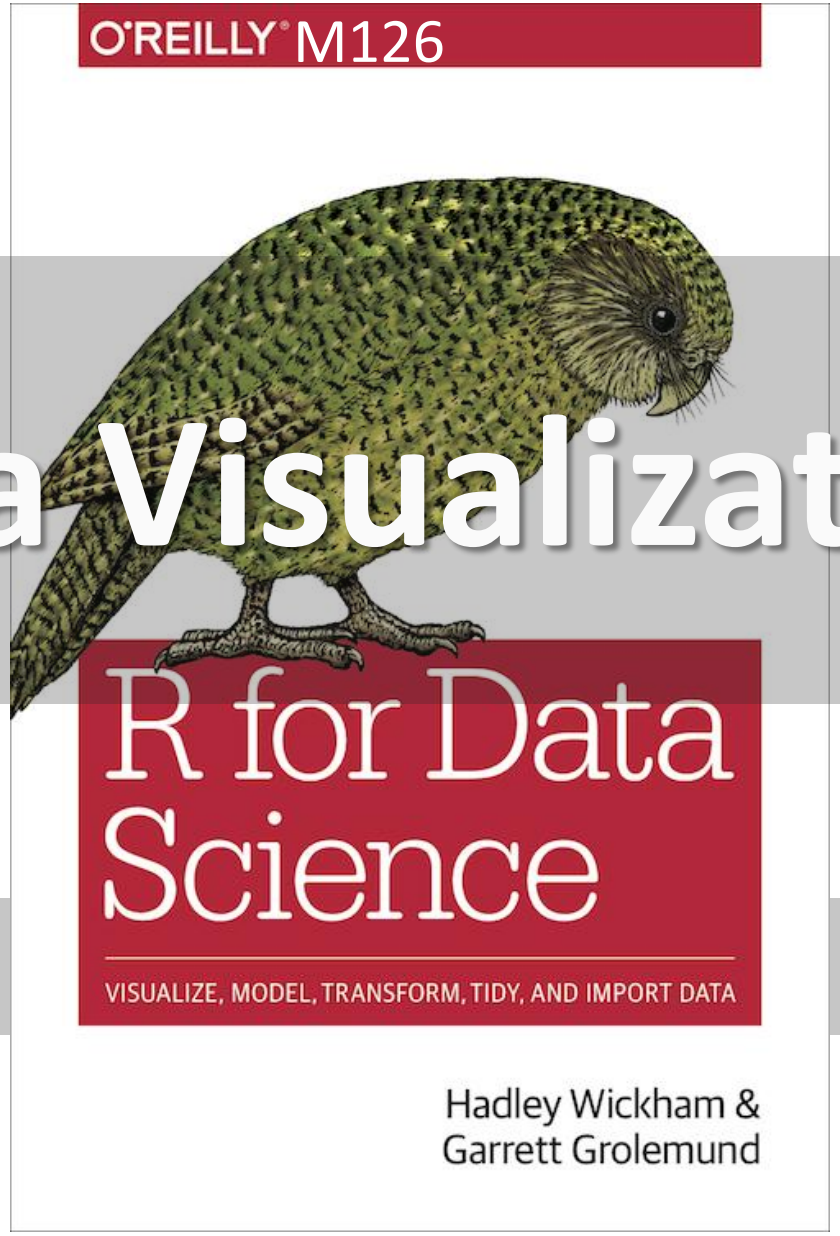
Data Visualization



https://r4ds.had.co.nz/

Why use R and its visualization packages

- R is the most popular language in the world of data science.
- R is an interpreted language. Hence, we can run code without any compiler.
- R is a vector language, so anyone can add functions to a single vector without putting in a loop. Hence, R is powerful and faster than other languages.

For data viz, R allows:

- Reproducibility
- Create many variations and iterations without having to start from scratch

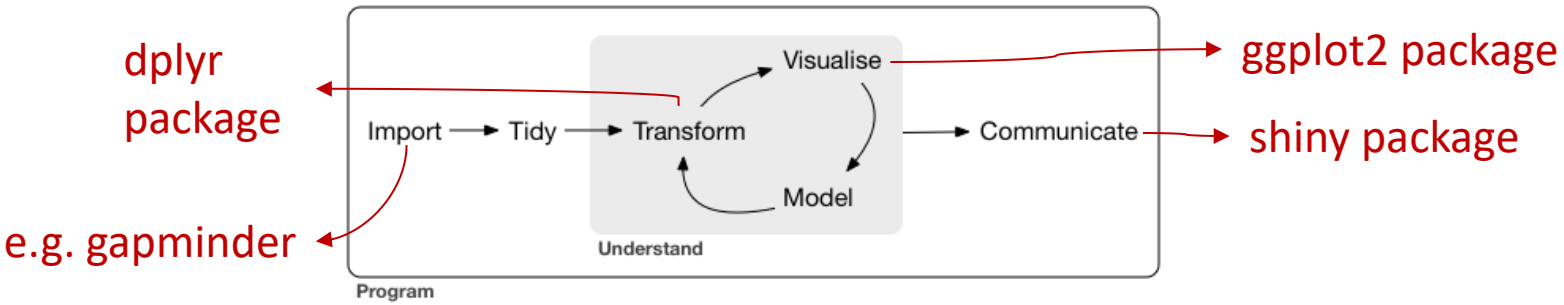
The workflow for visualizing with R

- **import** your data into R
- **tidy** your data
- **transform** your data (e.g. narrow in on observations of interest, create new variables, calculate a set of summary statistics)

wrangling

- **visualize**
- **model** (e.g. to answer questions you've made)
- **communicate**

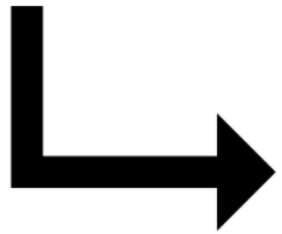
data viz



Tidy data and data manipulation

Country	2002	2007
Ecuador	12921234	13755680
Italy	57926999	58147733
Mozambique	18473780	19951656

UNTIDY



TIDY

Country	Year	Population
Ecuador	2002	12921234
Ecuador	2007	13755680
Italy	2002	57926999
Italy	2007	58147733
Mozambique	2002	18473780
Mozambique	2007	19951656

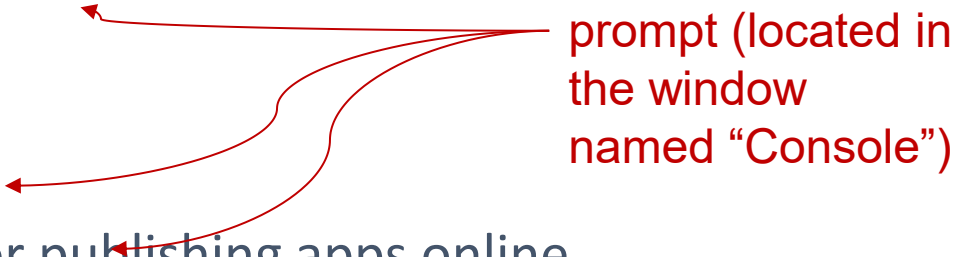
To start

- Download the most recent version of **R** https://cran.r-project.org/
- Once R is installed, download and install the IDE **R Studio** https://posit.co/download/rstudio-desktop
- In R Studio, install add-on packages for R:
  - **Tidyverse** - tidyverse.org

    install.packages("tidyverse") # includes ggplot2
  - **Shiny**

    install.packages("shiny")

    install.packages("rsconnect") # For publishing apps online
- You may also want to create a shinyapps.io account

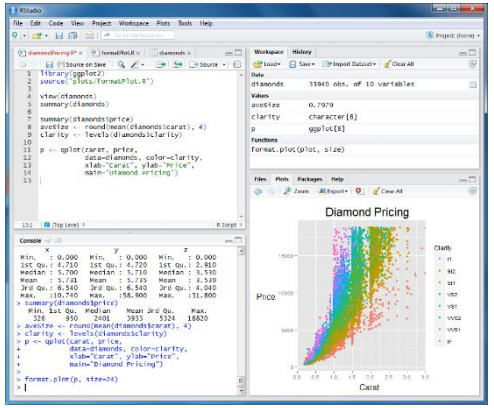


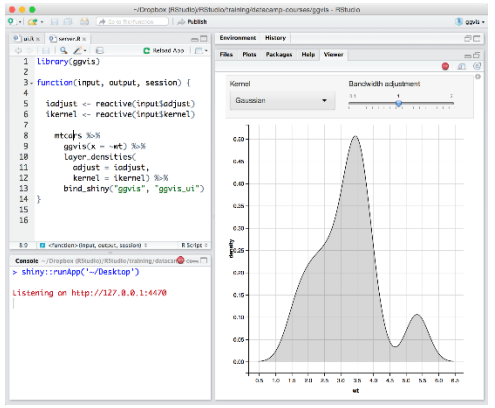
R resources

- The R Project for Statistical Computing https://www.r-project.org/
- The Comprehensive R Archive Network - https://cran.rstudio.com/
- TutorialsPoint - Learn R programming https://www.tutorialspoint.com/r/index.htm
- Swirl R package for learning R programming and data science http://swirlstats.com/
- Grolemund, G., & Wickham, H. (2017). ***R for Data Science***. O'Reilly Media. Retrieved from https://r4ds.had.co.nz/
- Codeschool R courses http://tryr.codeschool.com
- DataCamp course https://www.datacamp.com/

RStudio resources

- RStudio - https://posit.co/download/rstudio-desktop
- RStudio videos https://resources.rstudio.com/
- Rstudio's IDE features: https://rstudio.com/products/rstudio/features
- Rstudio's IDE video presentation: https://fast.wistia.net/embed/iframe/520zbd3tij?videoFoam=true
- RStudio Cheatsheets

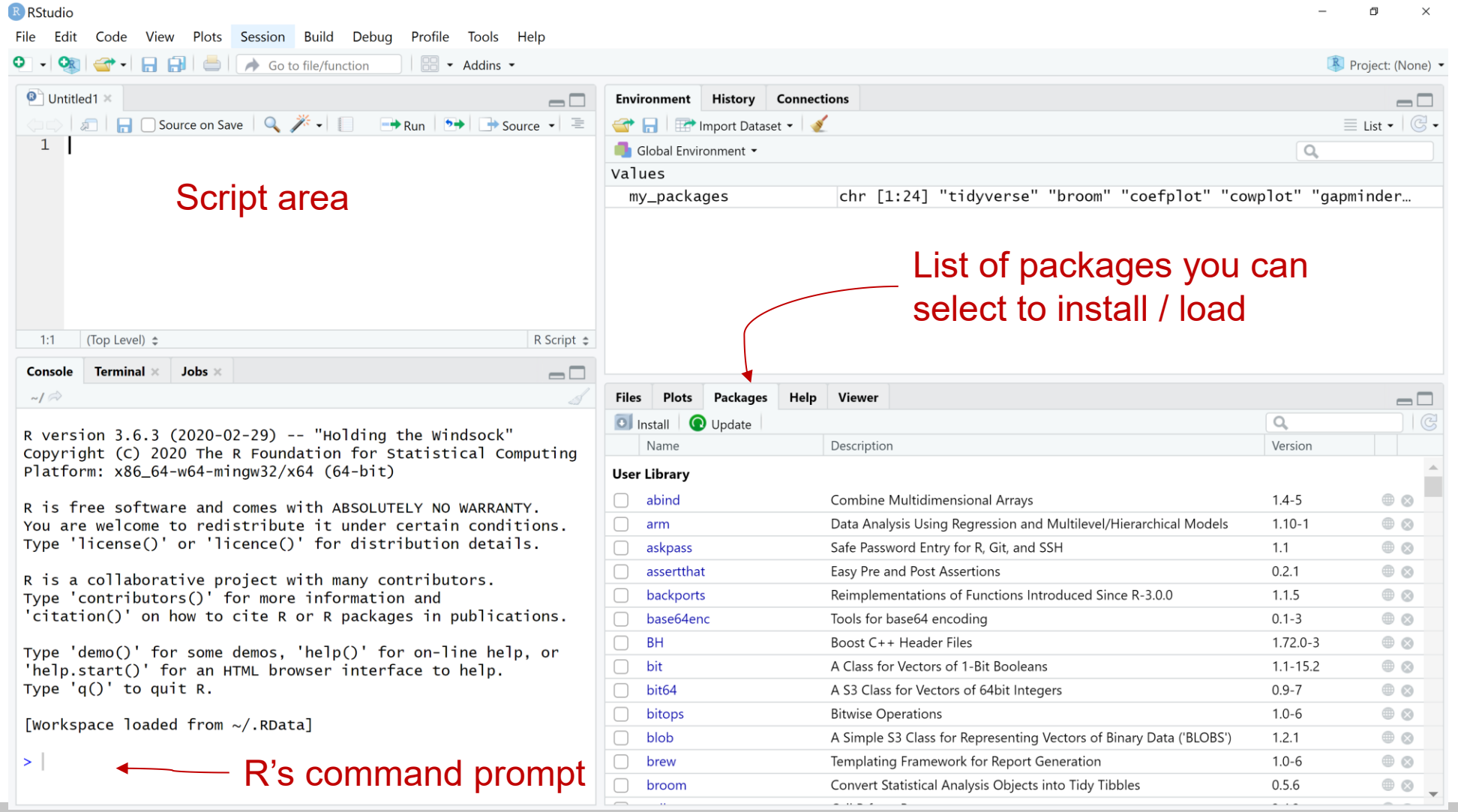




R Studio

Note: run as administrator!

- This is what RStudio looks like once installed



R Studio - installing packages

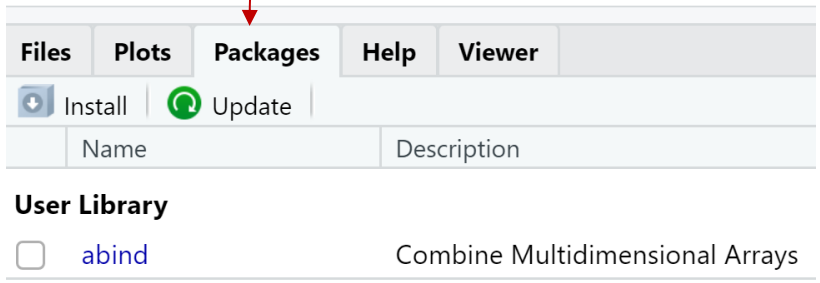
Note: R is case sensitive!

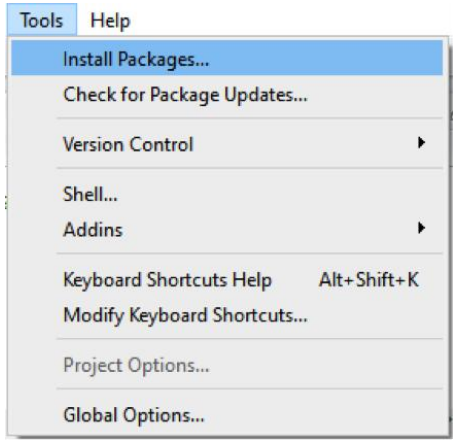
- Install at command line
  > install.packages("tidyverse")

  the package name

- Or install more than one packages at a time
  > install.packages(c("tidyverse", "ggplot2", "dplyr"))

- Or from the Packages tab, or the menu





- To view the current library path(s) location
  > .libPaths()

R Studio - loading packages

- You only need to install a package once, but you need to **reload** it every time you start a new session. E.g.

```
> library(tidyverse) # load the package tidyverse
> library(dplyr)
> library(gapminder)
```

at R's command prompt

- Or, altogether in a script file and then highlight and Ctrl + Enter

```
library(tidyverse)
library(dplyr)
library(gapminder)
```

- To see which packages are loaded in a session

```
> (.packages()) # listed by most recently loaded first
```

tidyverse

tidyverse is a collection of packages that are useful for many tasks like data management, data reshaping, data formatting, memory management, etc.
It includes:

- **ggplot2** (contains the functions used to create plots)
- dplyr (contains functions to work with data, e.g. subset, aggregate data)
- tidyr
- readr
- purrr
- tibble
- stringr
- Forcats

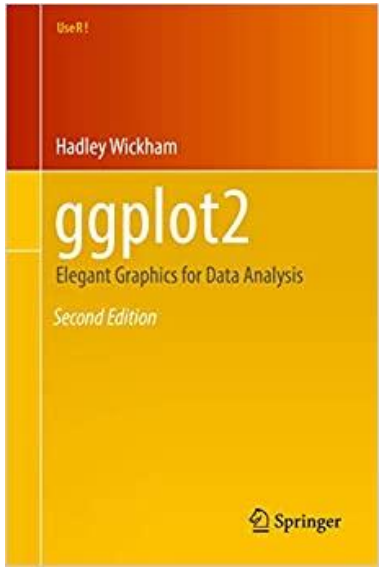
https://www.tidyverse.org/packages

ggplot2

- **ggplot** is a system for declaratively creating graphics, based on The Grammar of Graphics:

- Provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the rest.

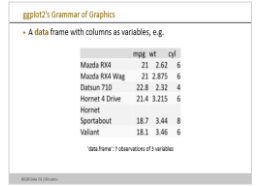
https://ggplot2.tidyverse.org/



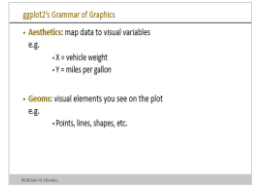
ggplot2 = the Grammar of Graphics plot

The Grammar of Graphics components:

1. **Data**: what we want to visualize. In R, data is stored in a data.frame, a rectangular collection of variables (in the columns) and observations (in the rows).



2. **Aesthetics**: visual properties of geoms e.g. x and y positions, colors, shapes, transparency, etc.



3. **Geoms**: the geometrical object that a plot uses to represent data, e.g. points, lines, areas, polygons, etc.

ggplot2's Grammar of Graphics

- A **data** frame with columns as variables, e.g.

	mpg	wt	cyl
Mazda RX4	21	2.62	6
Mazda RX4 Wag	21	2.875	6
Datsun 710	22.8	2.32	4
Hornet 4 Drive	21.4	3.215	6
Hornet Sportabout	18.7	3.44	8
Valiant	18.1	3.46	6

'data.frame': 7 observations of 3 variables

ggplot2's Grammar of Graphics

- **Aesthetics**: map data to visual variables

  e.g.

  - X = vehicle weight
  - Y = miles per gallon

- **Geoms**: visual elements you see on the plot

  e.g.

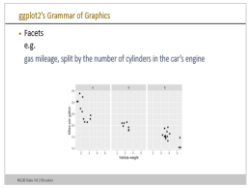
  - Points, lines, shapes, etc.

ggplot2 = the Grammar of Graphics plot

The Grammar of Graphics components (cont'd):

4. **Statistical transformations** (stats): The stats summarize data in many useful ways. E.g., binning and counting to create a histogram, regression line for regression analysis, etc.

5. **Facets**: allow to break up the data into subsets to visualize as small multiples.



6. **Scales**: Scales map values in the data space to values in the aesthetic space, whether it is color, size, or shape.

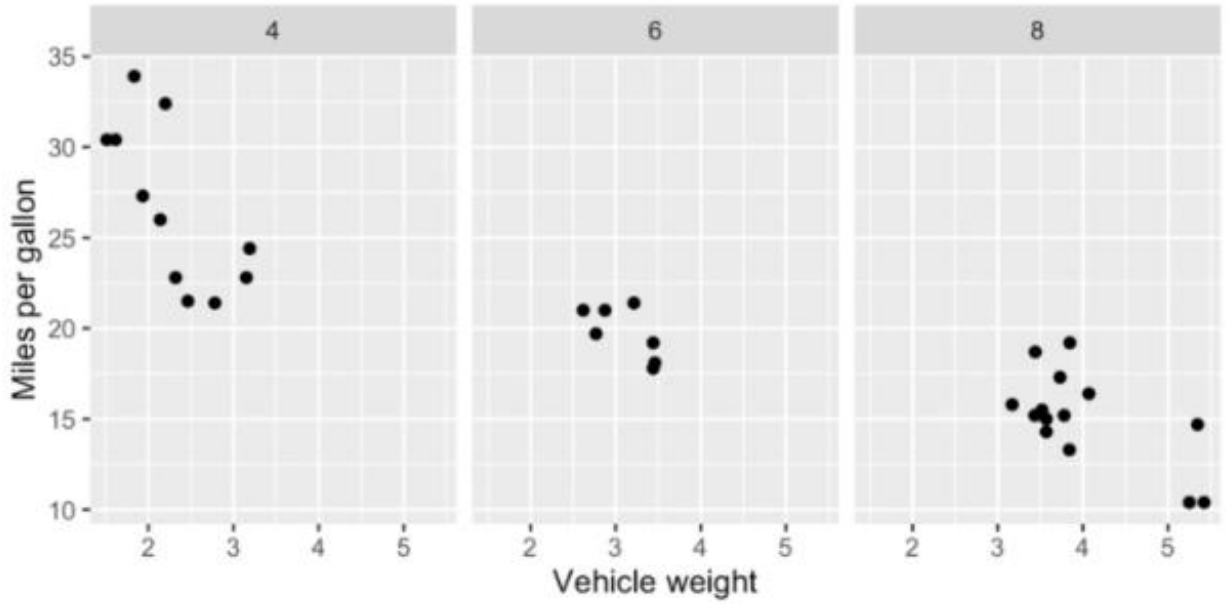
7. **Coordinates**: controls how data coordinates are mapped to the visual plane of graphics

ggplot2's Grammar of Graphics

- Facets

  e.g.

  gas mileage, split by the number of cylinders in the car's engine



ggplot2's Grammar of Graphics

- **Scales**: control how data values are mapped to aesthetics

  e.g.

  - Color scale
  - Log transform axis

Flow of action in ggplot2

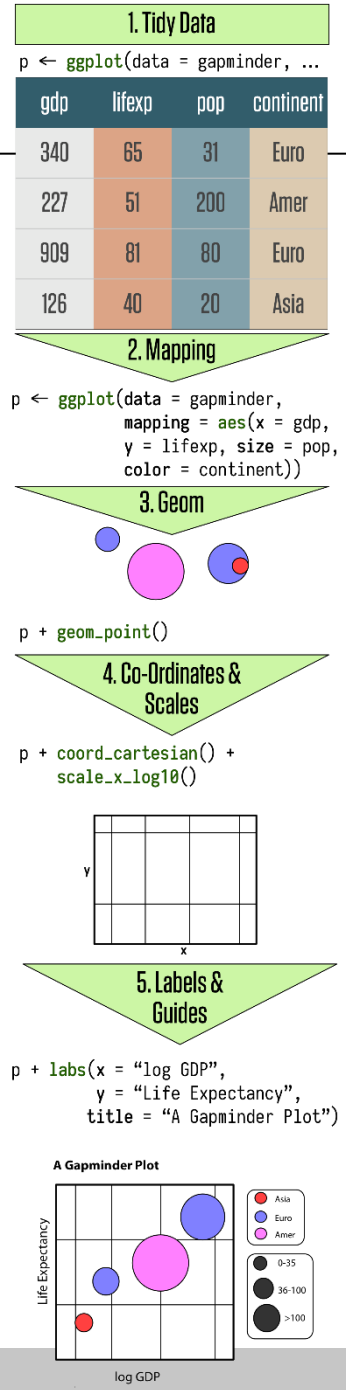
Start with a table of data →

Map the variables you want to display to aesthetics like position, color or shape →

Choose one or more geoms to draw the graph →

Define coordinates (if you don't want to use the default cartesian) and scales →

Fix your title, labels, legends, etc. →



Healy, 2019

Steps for making a plot with ggplot2

1. **Assign data**: declare the input data.frame (e.g., data=acs).
2. **Assign** a variable to aes **or set** the **aesthetics** (with aes()): assign or set the x-axis variable. E.g., to make a histogram or density curve using the x-axis variable only. To make a scatter plot, assign the y-axis variable together. You can assign a variable to color, fill, or size variable (e.g., color=sex) or set the variable(e.g., color="black").
3. **Specify the geom**(s): select various geoms layer by layer. E.g., select points, lines, areas, or polygons layer by layer.

Because the coordinate system and scales have default values, one can draw a plot without setting them. You can change the coordinate system or scales if needed.
Stats and facets can be added as desired.

ggplot2_cheat-sheet.pdf

ggplot2 installation

```
# The easiest way to get ggplot2 is to install the whole tidyverse:
install.packages("tidyverse")

# Alternatively, install just ggplot2:
install.packages("ggplot2")

# Or the development version from GitHub:
install.packages("devtools")
devtools::install_github("tidyverse/ggplot2")
```

Example of making a plot with ggplot2

- Let’s make some plots

(see e-class > Documents > ASSIGNMENTS - PROJECTS - TOOLS > R - ggplot2 – shiny)

For the examples we will use an excerpt from the Gapminder data, which is available in R as a package for the purpose of teaching and making code examples.

ggplot2 resources

- ggplot2: Elegant Graphics for Data Analysis by Hadley Wickham, https://ggplot2-book.org/
- R Graphics Cookbook by Winston Chang

- Online documentation: https://docs.ggplot2.org/current

- https://ggplot2.tidyverse.org

plotly

- plotly is an R package to create a variety of interactive graphics
- There are two main ways to creating a plotly object:
  - by transforming a **ggplot2** object (via *ggplotly()*) into a **plotly** object
  - by directly initializing a **plotly** object with *plot_ly() / plot_geo() / plot_mapbox()*

https://plotly-r.com/overview.html

Shiny

- Shiny is merely an R package like dplyr or ggplot2. The package is used to **create web-applications**, but uses the R language, rather than javascript or HTML5, which are traditionally used for web applications.

- By using R, Shiny provides an efficient method of creating web applications designed around data presentation and analysis. Shiny apps are useful for:
  - Interactive data visualization for presentations and websites
  - Sharing results with collaborators
  - Communicating science in an accessible way
  - Bridging the gap between R users and non-R users

Example of a Shiny app

Iris k-means clustering

X Variable

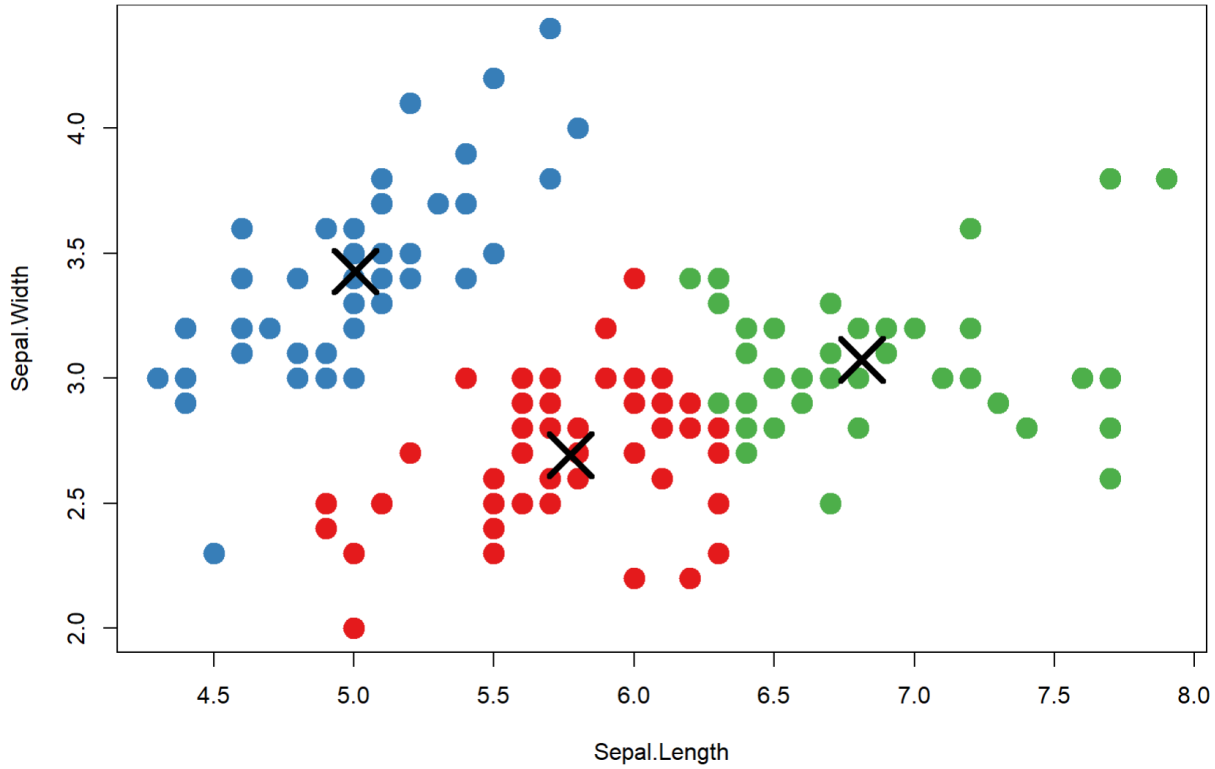
Sepal.Length

Y Variable

Sepal.Width

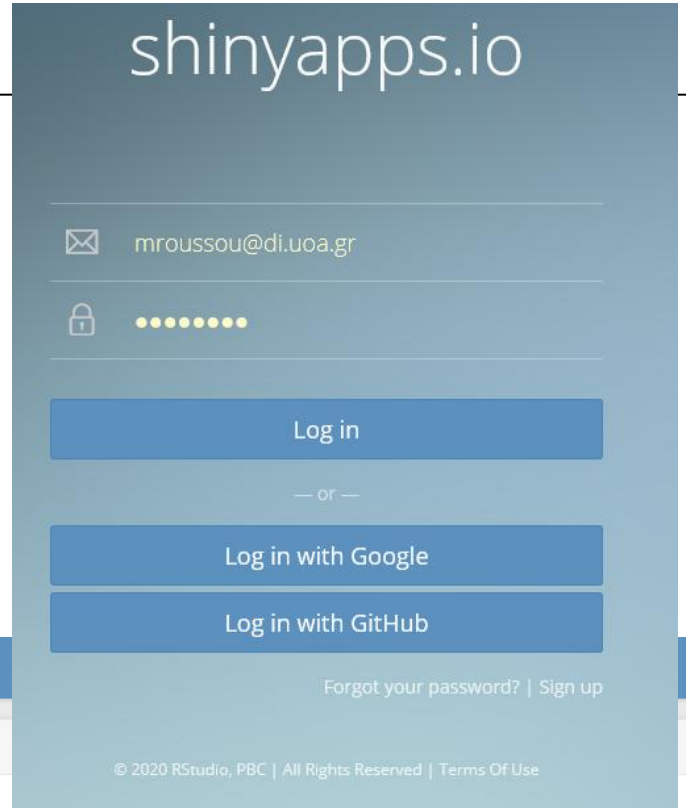
Cluster count

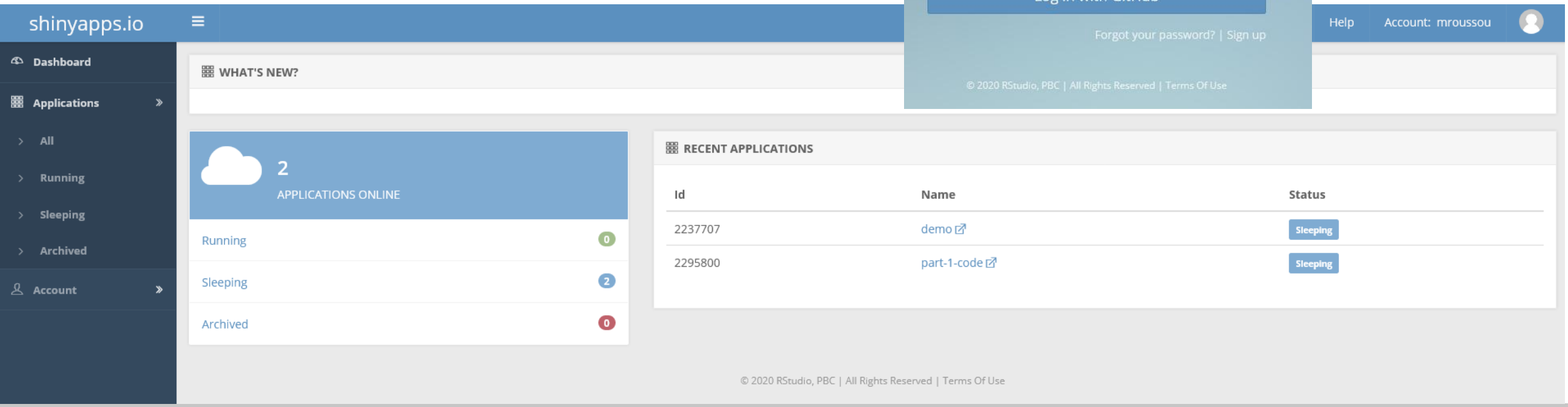
3



shinyapps.io

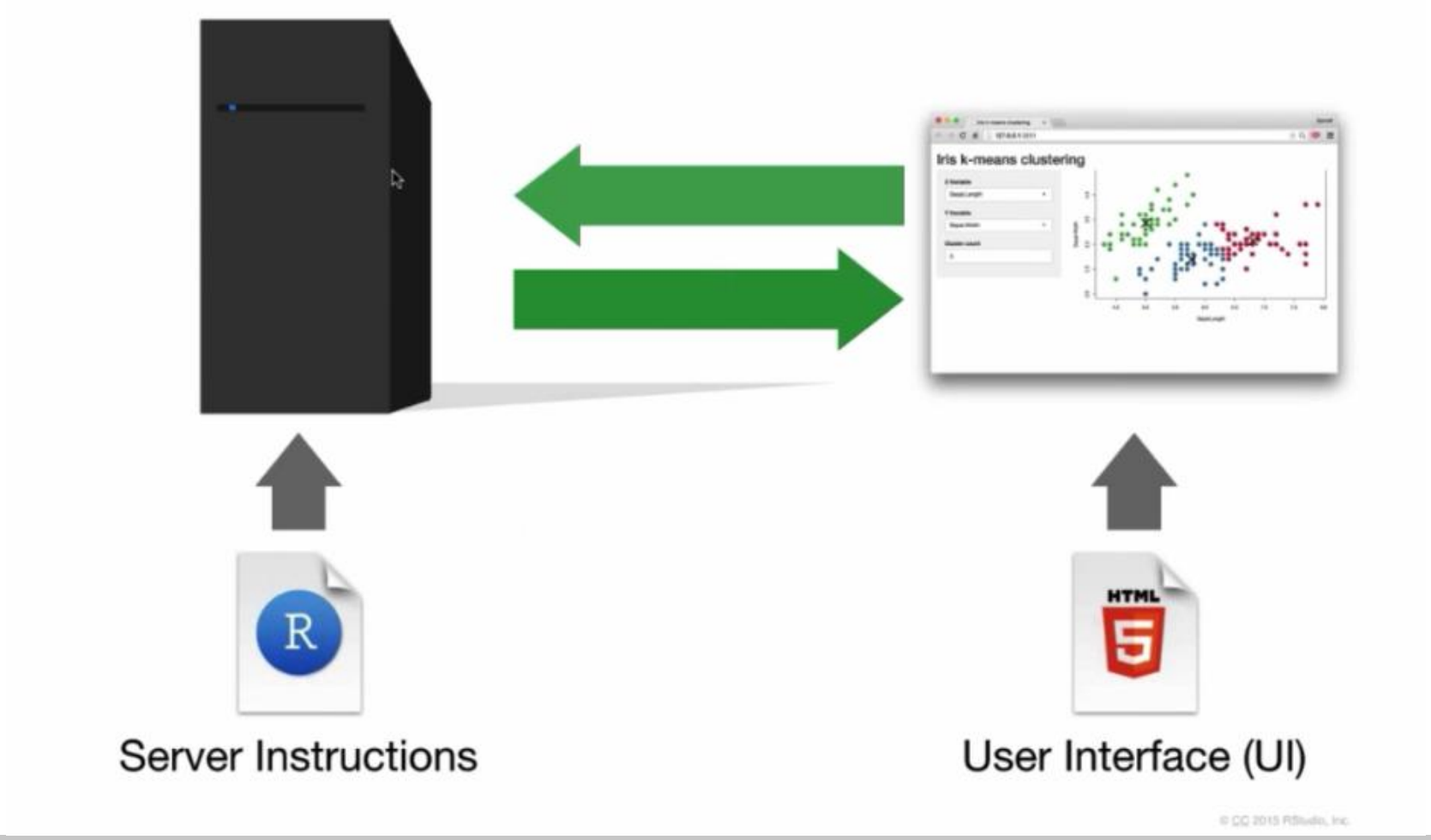
- www.shinyapps.io



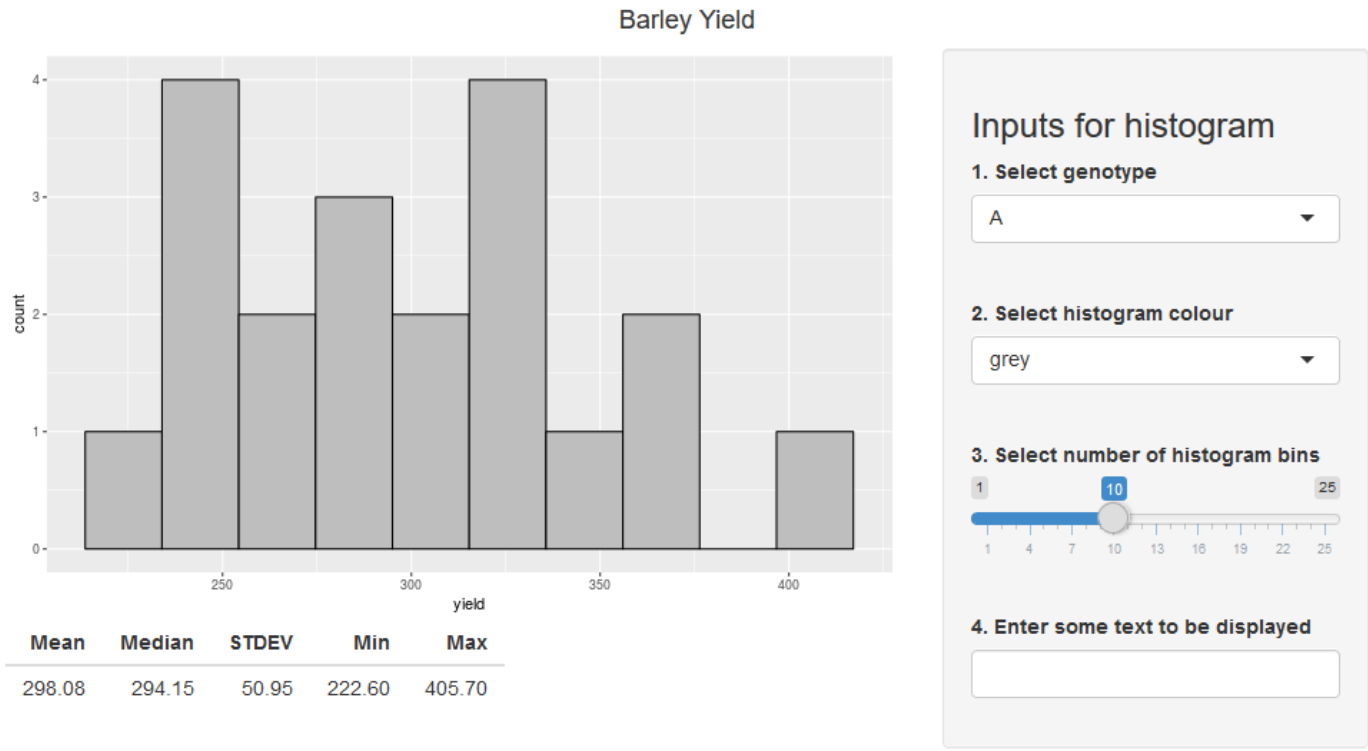


Shiny app architecture

- 2 components: a UI and a set of instructions for the server



Example of making a Shiny app



see e-class > Documents > ASSIGNMENTS - PROJECTS - TOOLS > R - ggplot2 – shiny > R - RStudio - Shiny app example tutorial

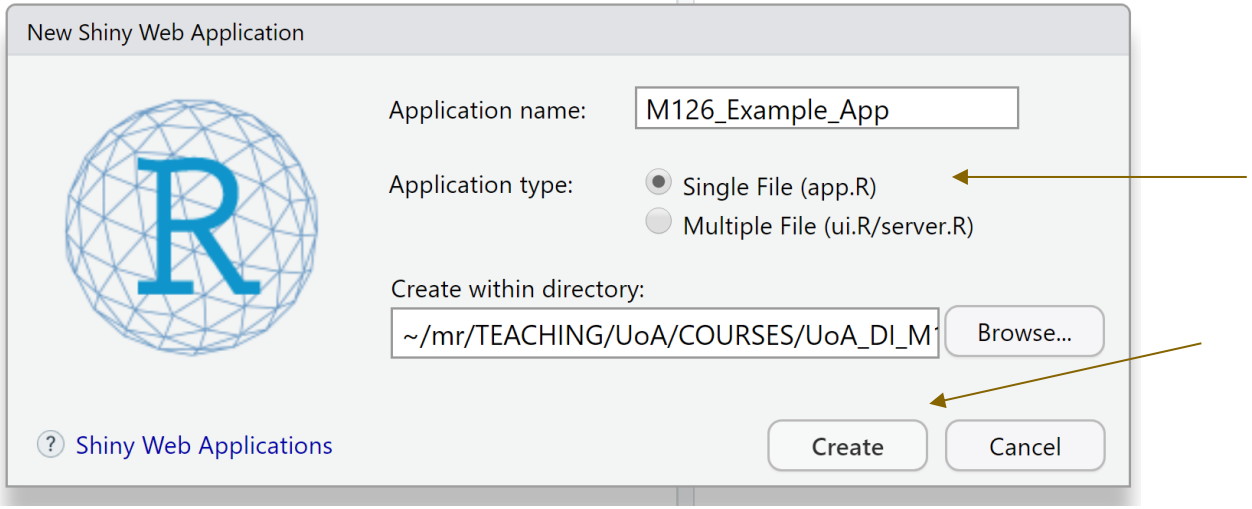
(CC-11-Shiny-master.zip)

Downloading Shiny and tutorial resources

```
install.packages("shiny")

# For publishing apps online
install.packages("rsconnect")

# For the dataset in today's tutorial
install.packages("agridat")
```

Create a new app file in R Studio

- In Rstudio's menu File > New File > Shiny Web App



- Add a folder called **Data** and a folder called **www**

```
Test_App
├── app.R
├── Data
│   └── data.csv
└── www
    └── A.jpg
```

Shiny app template

- Use the following template to make any Shiny app (app.R):

```
library(shiny)

#sets up a UI object
ui <- fluidPage()

#a server object
server <- function(input, output) {}

#brings them together
shinyApp(ui = ui, server = server)
```

Shiny app template

- For this example we will load some packages and data:

```
library(shiny)
library(ggplot2)
library(dplyr)
library(agridat)

#loading data
Barley <- as.data.frame(beaven.barley)

#sets up a UI object
ui <- fluidPage()

#a server object
server <- function(input, output) {}

#brings them together
shinyApp(ui = ui, server = server)
```

Shiny app UI element

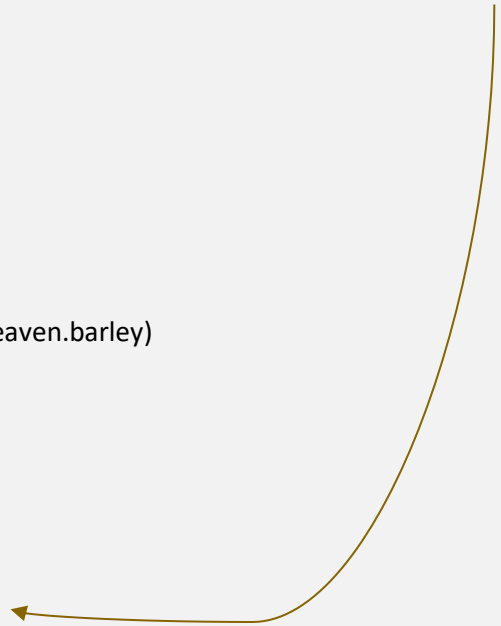
- Add elements to your app as arguments to fluidPage()

```
library(shiny)
ui <- fluidPage("Hello World")

server <- function(input, output) {}

shinyApp(ui = ui, server = server)
```

Shiny app layout

Now let's make the panel layout:

```
library(shiny)
library(ggplot2)
library(dplyr)
library(agridat)

#loading data
Barley <- as.data.frame(beaven.barley)

#sets up a UI object
ui <- fluidPage(
      titlePanel(""),
      sidebarLayout(
        sidebarPanel(),
        mainPanel()
      )
   )

#a server object
server <- function(input, output) {}

#brings them together
shinyApp(ui = ui, server = server)
```



Shiny app: inputs and outputs

- Build your app using **inputs** and **outputs**

Shiny app: inputs and outputs

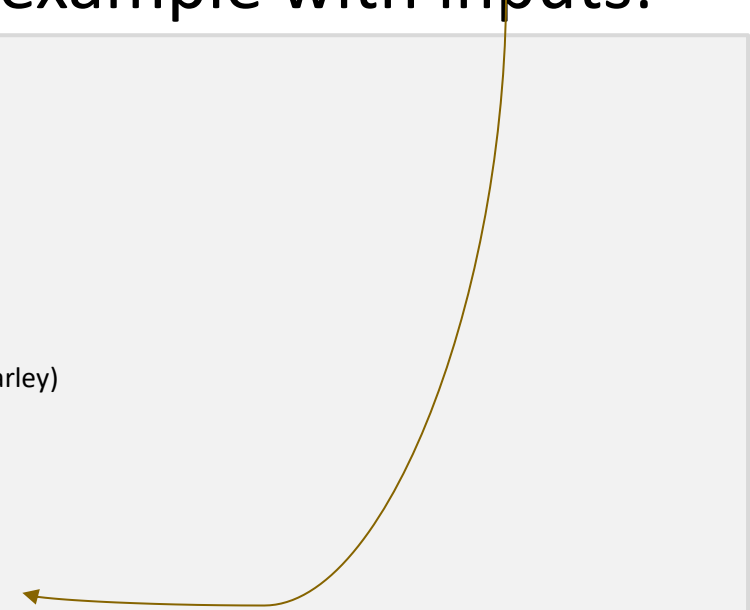
- Add elements to your app as arguments to fluidPage()

```
ui <- fluidPage(
  # *Input() functions,
  # *Output() functions
)
```

Shiny app input widgets

Now let's fill our example with inputs:

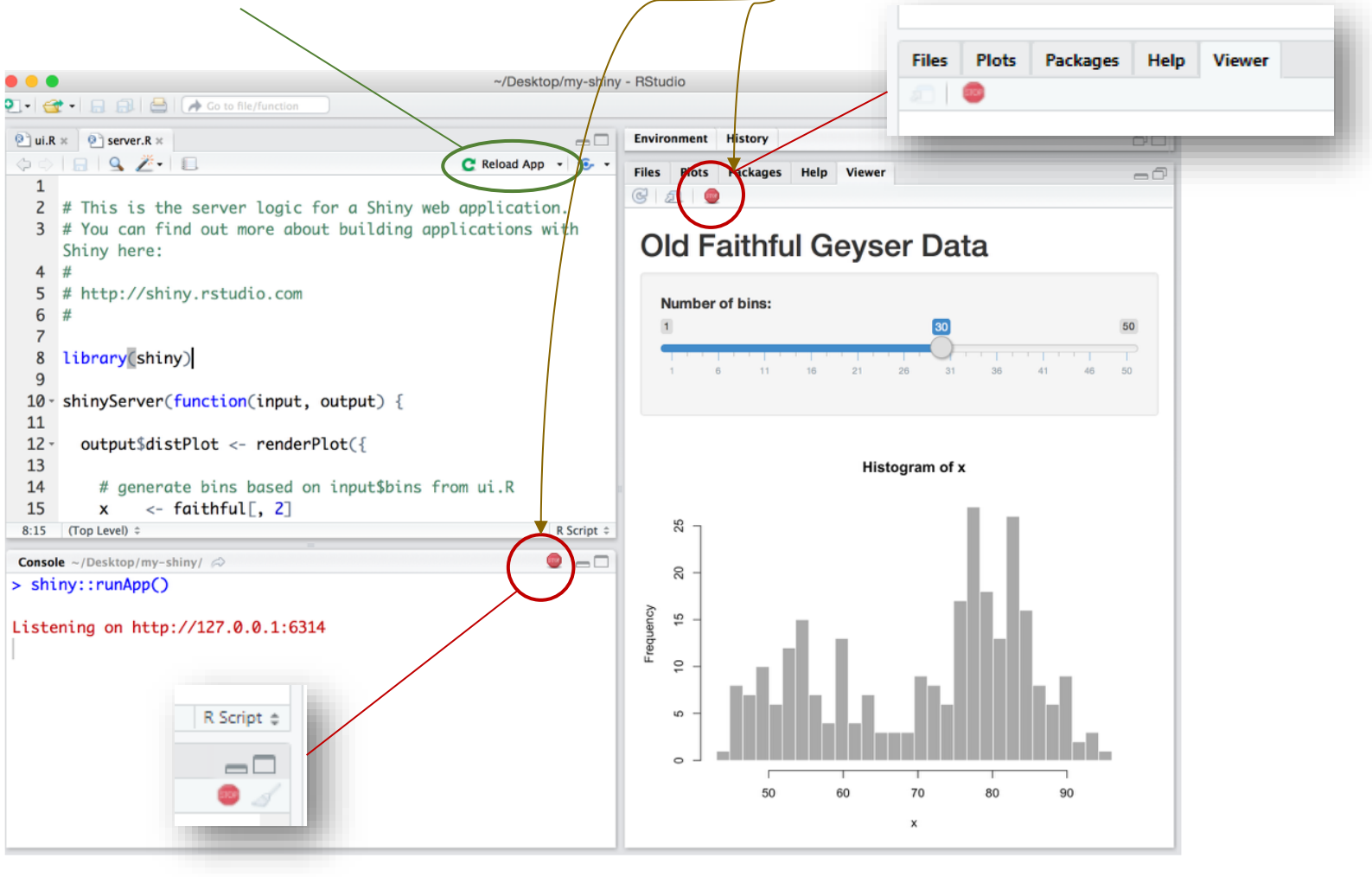
```
library(shiny)
library(ggplot2)
library(dplyr)
library(agridat)

#loading data
Barley <- as.data.frame(beaven.barley)

#sets up a UI object
ui <- fluidPage(
      titlePanel(""),
      sidebarLayout(
        sidebarPanel(),
        mainPanel()
      )
   )

#a server object
server <- function(input, output) {}

#brings them together
shinyApp(ui = ui, server = server)
```



Shiny app: close an app

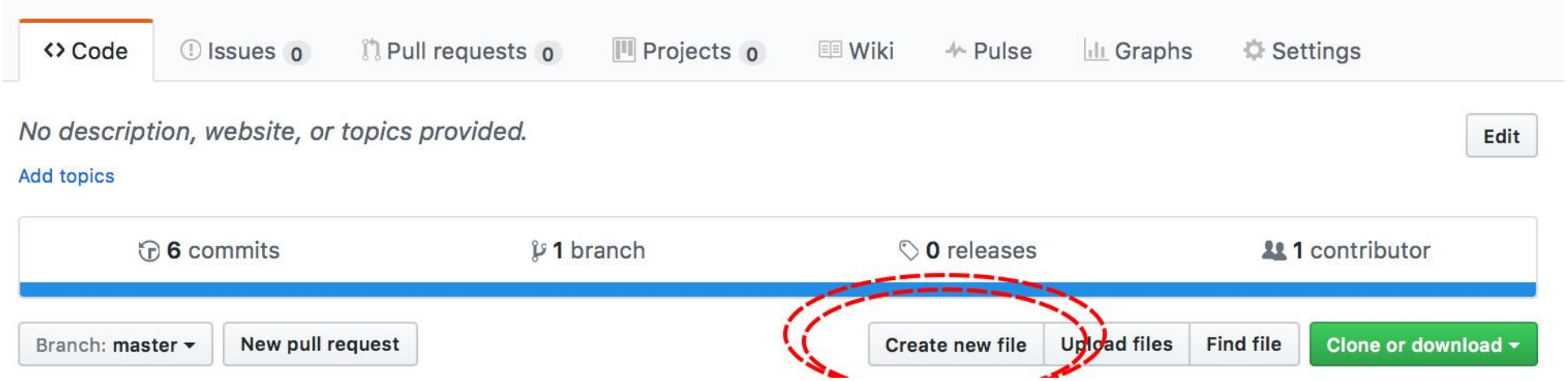
To run app: Run App

To stop app:



Exporting a finished Shiny app

- using Github
  - sign in to Github with your account details, create a repository and upload everything from your app folder, including any Data and www folders.

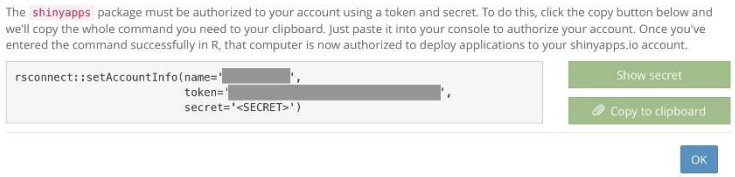


  - to send the app to another person, give them your Github username and the name of the app repo and ask them to run runGithub() in R, like this:

```
runGitHub(repo = "repo_name", username = "user_name")
```

Exporting a finished Shiny app

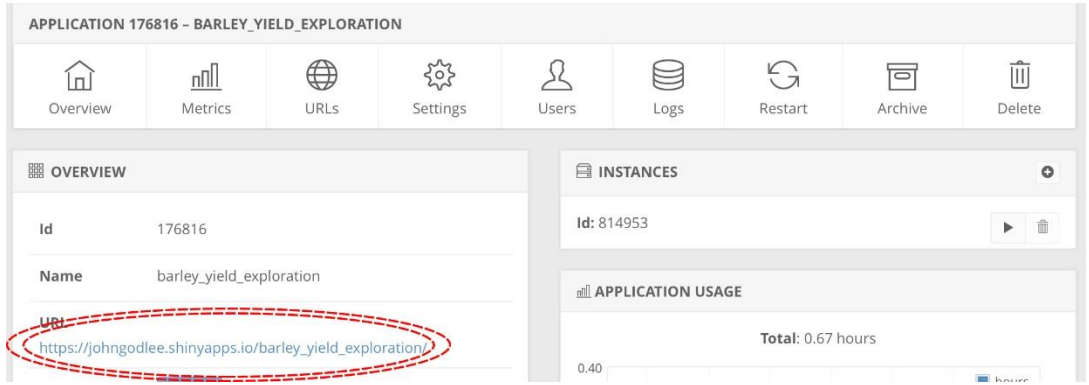
- host on www.shinyapps.io
  - Create an account
  - Go to www.shinyapps.io/admin/#/tokens, click *Show secret* and copy the rsconnect account info:



  - Then open up an R session and run the copied material to link shinyapps.io with R Studio.
  - To upload the app, open the app.R and click the publish button. Select a name for the app (**no spaces**) and click *Publish*.

Exporting a finished Shiny app

- host on www.shinyapps.io (*cont'd*)
  - The app URL is in the app info in the dashboard

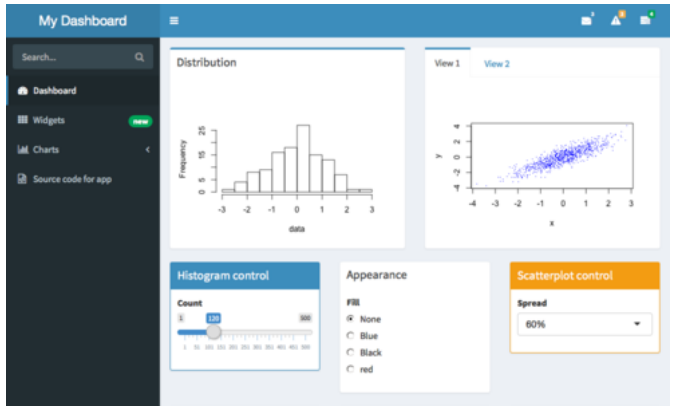


  - To embed an app in your own website use an iframe, replacing the URL with your own app URL and altering the style arguments to your own desire:

```
<iframe src=https://johngodlee.shinyapps.io/barley_yield_exploration/
style="border:none;width:1000px;height:500px;"></iframe>
```

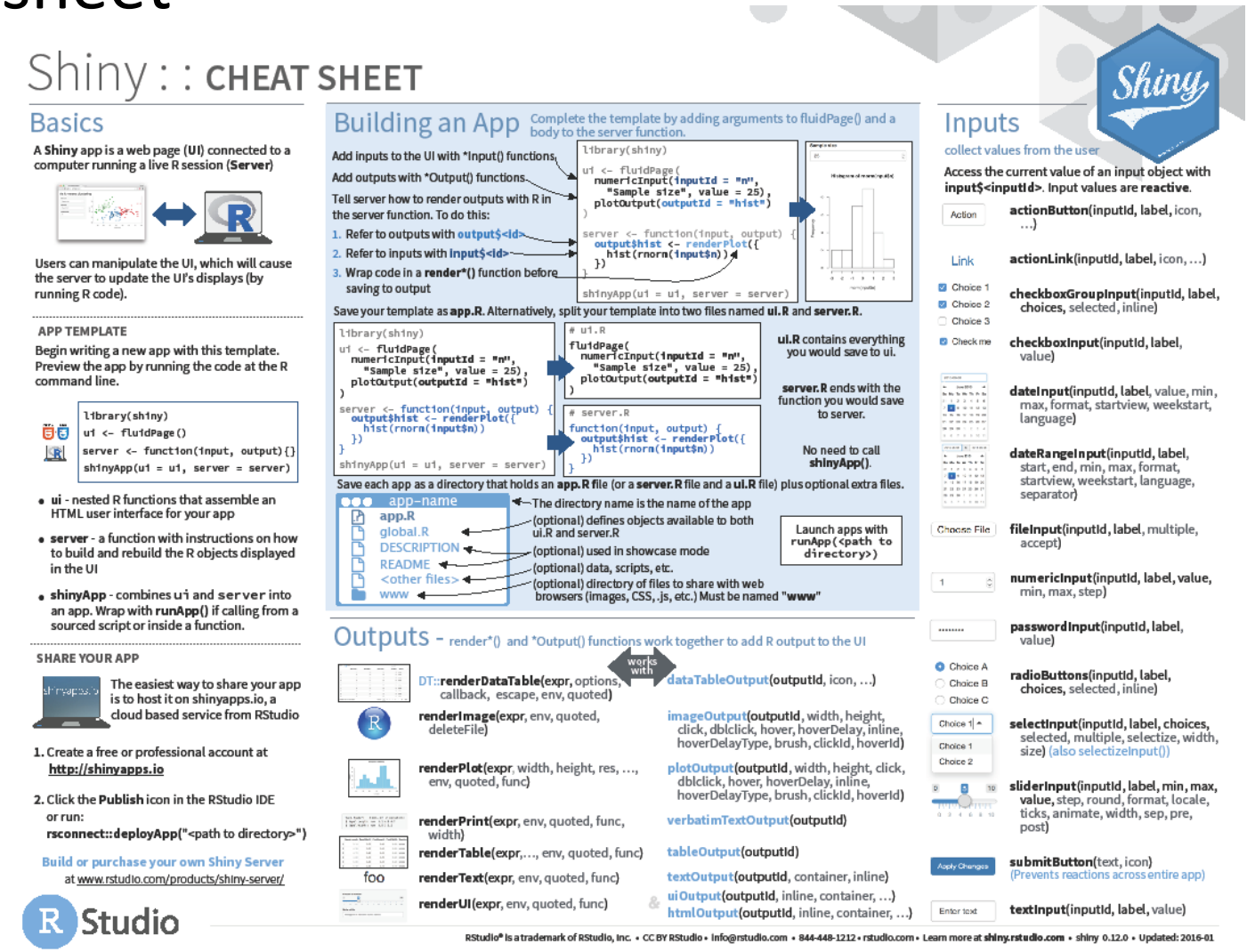
Shiny resources

- Shiny gallery: https://shiny.rstudio.com/gallery/
- 2-3 hour Shiny Tutorial: https://shiny.rstudio.com/tutorial/
- shinyApps.io platform: www.shinyapps.io
- Shinydashboard https://rstudio.github.io/shinydashboard/



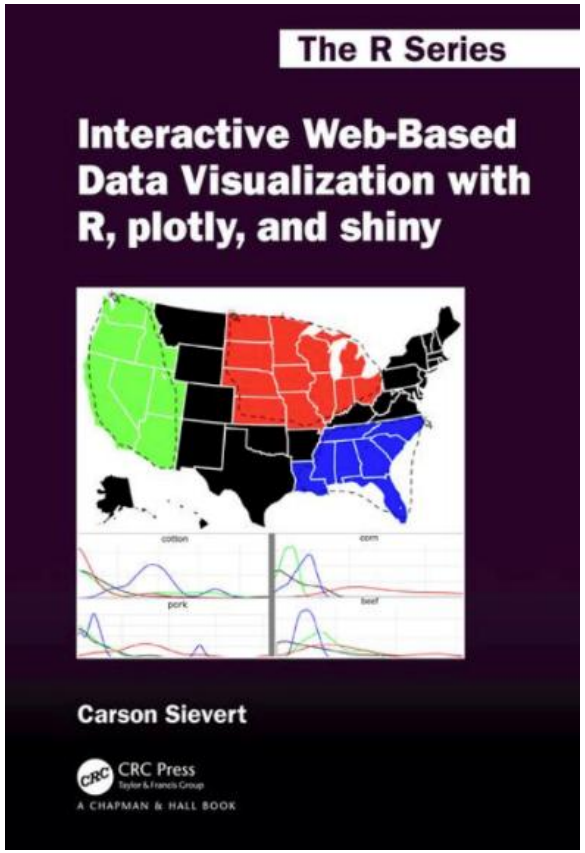
Shiny resources

- Shiny cheat sheet



Shiny resources

- Interactive web-based data visualization with R, plotly, and shiny: https://plotly-r.com/



Learn ggplot2 using Shiny App

- Moon, K.-W. (2016). *Learn ggplot2 Using Shiny App*. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-53019-2

- http://r-graph.com/
- Online app: https://cardiomoon.shinyapps.io/ggplot2new/
- *or*, in Rstudio: shiny::runGitHub("cardiomoon/ggplot2new")

Other R stuff

- Leaflet: an open-source JavaScript library for mobile-friendly interactive maps https://leafletjs.com/
- Leaflet R package to integrate and control Leaflet maps in R: https://rstudio.github.io/leaflet/

Thank you!

mroussou@di.uoa.gr

http://eclass.uoa.gr/courses/DI411/