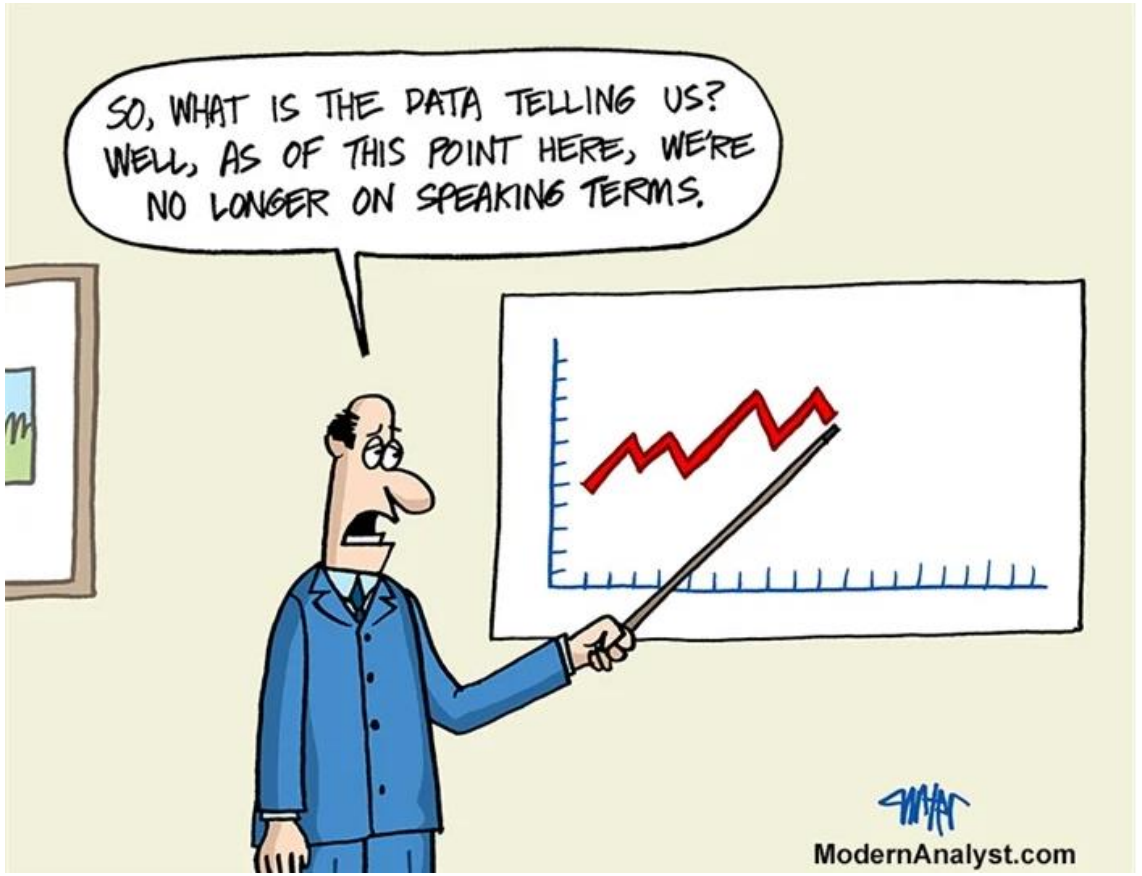
Interactive “shiny” apps, ethics, and conclusions



Overview

Shiny for interactive web applications

Brief mention/pointers to additional topics

- Ethics
- If there is time: Clustering and principal component analysis (PCA)

Conclusions

Announcement

There will be no late penalty for the final project as long it's turned in before the end of reading period

I still highly recommend you turn it in at the original deadline so that you have plenty of time to study for the final exam

- The final exam is weighted significantly more than the final project

Interactive applications

Shiny

Shiny is an R package that makes it easy to build interactive web apps

Example: k-means clustering on Fisher's Iris data set

Setosa

Virginica

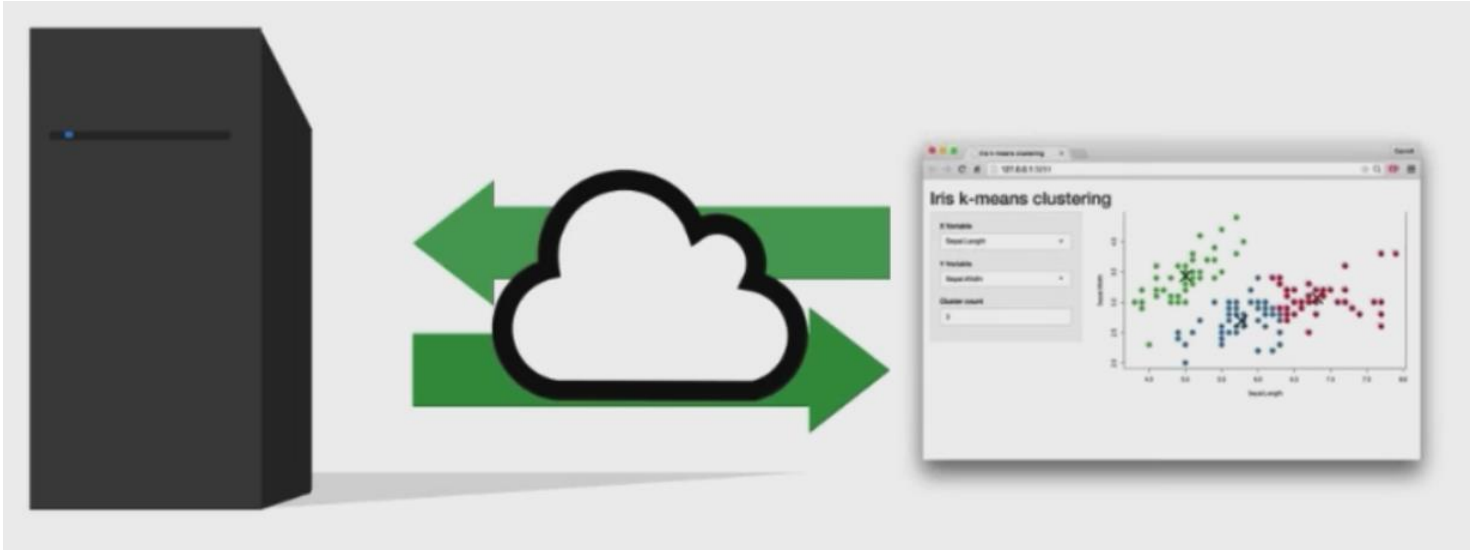
```
# Look at the original iris data frame
> View(iris)
```

Tutorial: https://shiny.rstudio.com/tutorial/

Shiny applications

Server runs R code, creates results

Client uses a web-based GUI to interact with code



You need to write 2 pieces of code to create a Shiny app:

- server: for the code that is run on the server
- ui: for the web interface shown to the user

Shiny application template

```
# include the shiny package
library(shiny)

# the function to create the user interface
ui <- fluidPage()

# the function to create the server
server <- function(input, output) {}

# putting them together to run
shinyApp(ui = ui, server = server)
```

UI code converted to
HTML for web browsers

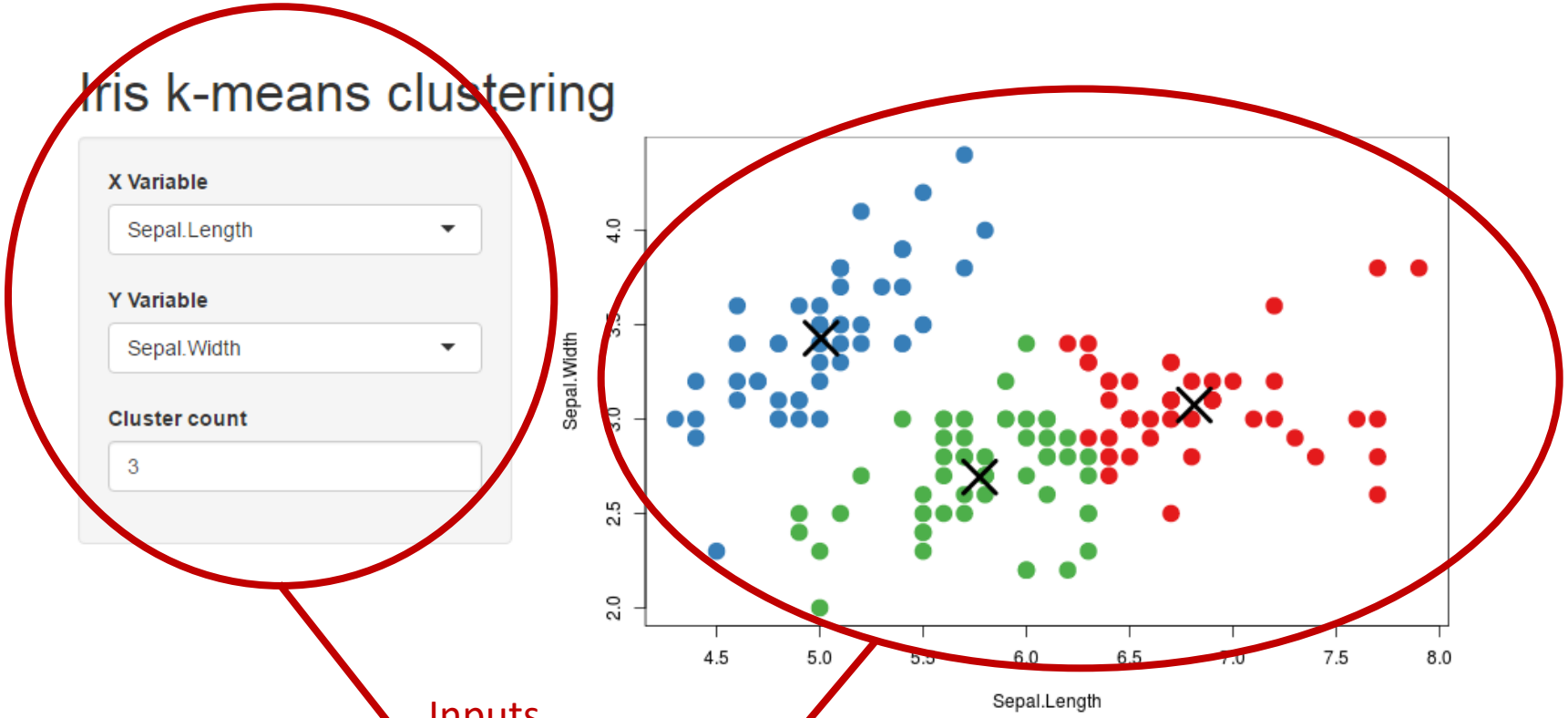
```
> SDS230::download_class_code(26)
```

Start by creating a .R script that has this code
Follow along by continually testing the code…

Shiny application template

```
# include the shiny package
library(shiny)

# the function to create the user interface
ui <- fluidPage("Hello world!")

# the function to create the server
server <- function(input, output) {}

# putting them together to run
shinyApp(ui = ui, server = server)
```

Think in terms of inputs and outputs



Inputs

Outputs

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

Building a UI

```
# adding a slider…
ui <- fluidPage(

        sliderInput(inputId = "num",
              label = "Choose a number",
              val = 25, min = 1, max = 100)

)    # notice the closing parenthesis!
```

Indentation is important to stay organized!

Input functions:

Buttons

Action

Submit

`actionButton()`
`submitButton()`

Single checkbox

Choice A

`checkboxInput()`

Checkbox group

Choice 1
Choice 2
Choice 3

`checkboxGroupInput()`

Date input

2014-01-01

`dateInput()`

Date range

2014-01-24 to 2014-01-24

`dateRangeInput()`

File input

Choose File No file chosen

`fileInput()`

Numeric input

1

`numericInput()`

Password Input

`passwordInput()`

Radio buttons

Choice 1
Choice 2
Choice 3

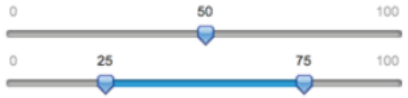
`radioButtons()`

Select box

Choice 1

`selectInput()`

Sliders



`sliderInput()`

Text input

Enter text...

`textInput()`

See Cheat sheet!

Input functions!

Input functions all have a similar form:

```
sliderInput(inputId = "num",
            label = "Choose a number",
            val = 25, min = 1, max = 100)
```

Name to refer to returned value

Label for user to see

Input type specific arguments

Use help page to learn about input arguments:

```
> ? sliderInput
```

Output functions

Function	Inserts
`dataTableOutput()`	an interactive table
`htmlOutput()`	raw HTML
`imageOutput()`	image
`plotOutput()`	plot
`tableOutput()`	table
`textOutput()`	text
`uiOutput()`	a Shiny UI element
`verbatimTextOutput()`	text

Always need to give outputs a name

Example: plotOutput(outputId = "my_plot")

Building a UI

```
ui <- fluidPage(
        sliderInput(inputId = "num",
           label = "Choose a number",
           val = 25, min = 1, max = 100),
        plotOutput("my_plot")

)
```

Don’t forget the comma!

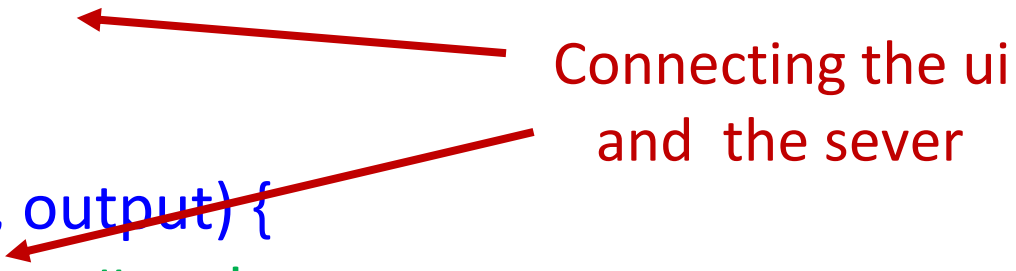
Server function

The server function connects inputs to outputs

```
server <- function(input, output) {

        output$my_plot <-   # code

}
```

Connecting the ui and the server

```
ui <- fluidPage(
        sliderInput(inputId = "num",
           label = "Choose a number",
           val = 25, min = 1, max = 100),
        plotOutput("my_plot")
)

server <- function(input, output) {
        output$my_plot <-   # code
}
```



Sever function

Sever function connects inputs to outputs

```
server <- function(input, output) {

        output$my_plot <-   renderPlot({

                # add your plot here!
                # e.g., hist(rnorm(100))   # boring

        })

}
```

Connecting the ui and the server

```
ui <- fluidPage(
        sliderInput(inputId = "num",
           label = "Choose a number",
           val = 25, min = 1, max = 100),
        plotOutput("my_plot")
)

server <- function(input, output) {
        output$my_plot <-   renderPlot({

        })
}
```

Usually a pairing of xOutput and renderX

See Shiny Cheat Sheet for more pairs

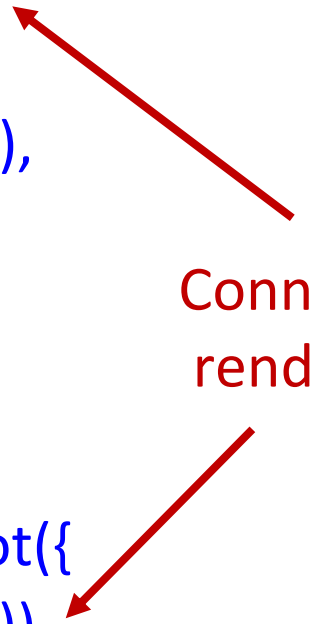
Render functions

Render functions take R output and place it in an HTML page in the UI

function	creates
`renderDataTable()`	An interactive table (from a data frame, matrix, or other table-like structure)
`renderImage()`	An image (saved as a link to a source file)
`renderPlot()`	A plot
`renderPrint()`	A code block of printed output
`renderTable()`	A table (from a data frame, matrix, or other table-like structure)
`renderText()`	A character string
`renderUI()`	a Shiny UI element

Connecting the ui and the server

```
ui <- fluidPage(
        sliderInput(inputId = "num",
           label = "Choose a number",
           val = 25, min = 1, max = 100),
        plotOutput("my_plot")
)

server <- function(input, output) {
        output$my_plot <-   renderPlot({
                hist(rnorm(input$num))
        })
}
```

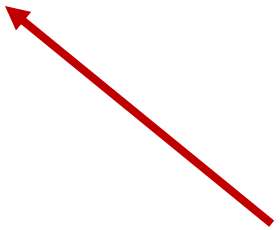


Connect UI input to render output

Connecting the ui and the server

```
ui <- fluidPage(
        sliderInput(inputId = "num",
          label = "Choose a number",
          val = 25, min = 1, max = 100),
        plotOutput("my_plot")
)

server <- function(input, output) {
        output$my_plot <-   renderPlot({
                hist(rnorm(input$num))
        })
}
```

i.e., it is ‘reactive’.
The plot is redrawn every time slider value changes



This function is called every time the input$num value changes!

Sharing Shiny apps

Create a directory

Either:

- Save your file as app.R with both sever and ui functions
- Save as two files: server.R and ui.R
- Embed code in an R Markdown file

If you host this directory on a Shiny server you can access this over the web

- Can host with RStudio (shinyapps.io)

Layout managers

Layout managers allow you to better position items on the web page (i.e., better ui)

```
ui <- fluidPage(

    sidebarLayout(

        sidebarPanel(  # add controls here),
        mainPanel(   # add plots here )

    ) # end sidebarLayout

)   # end ui
```

Don’t forget the comma!

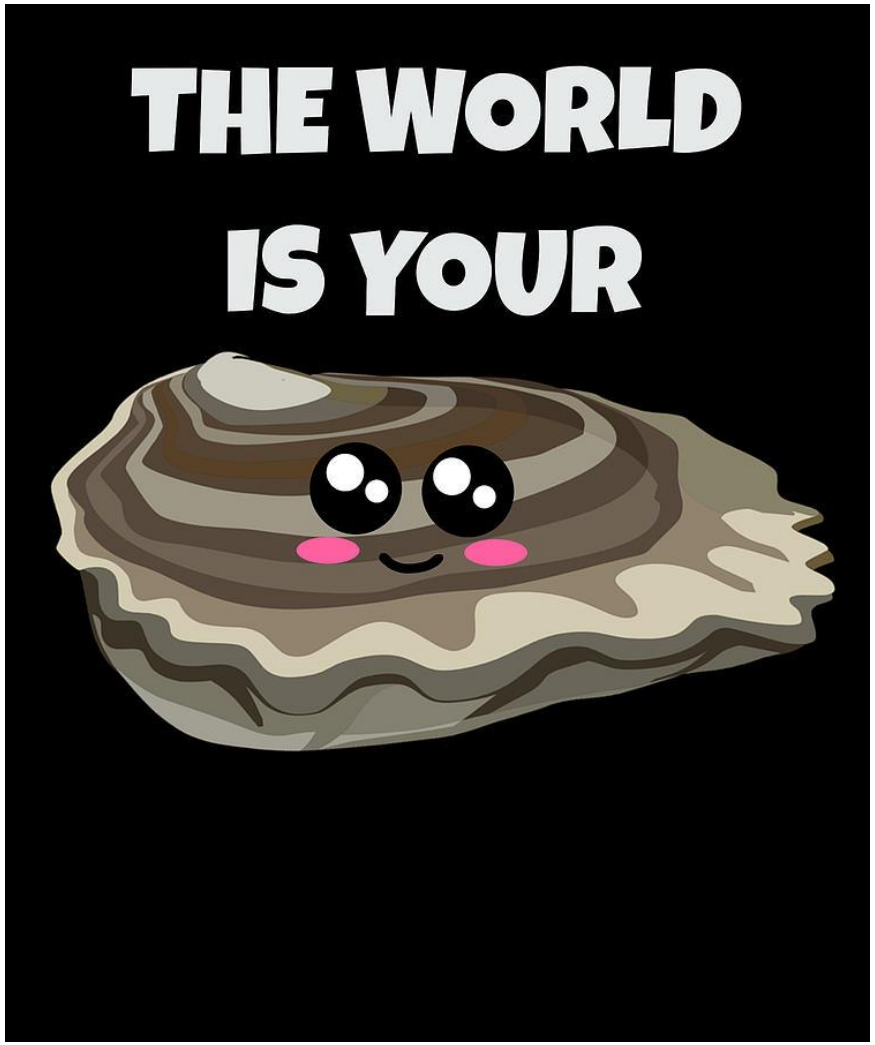
Layout managers

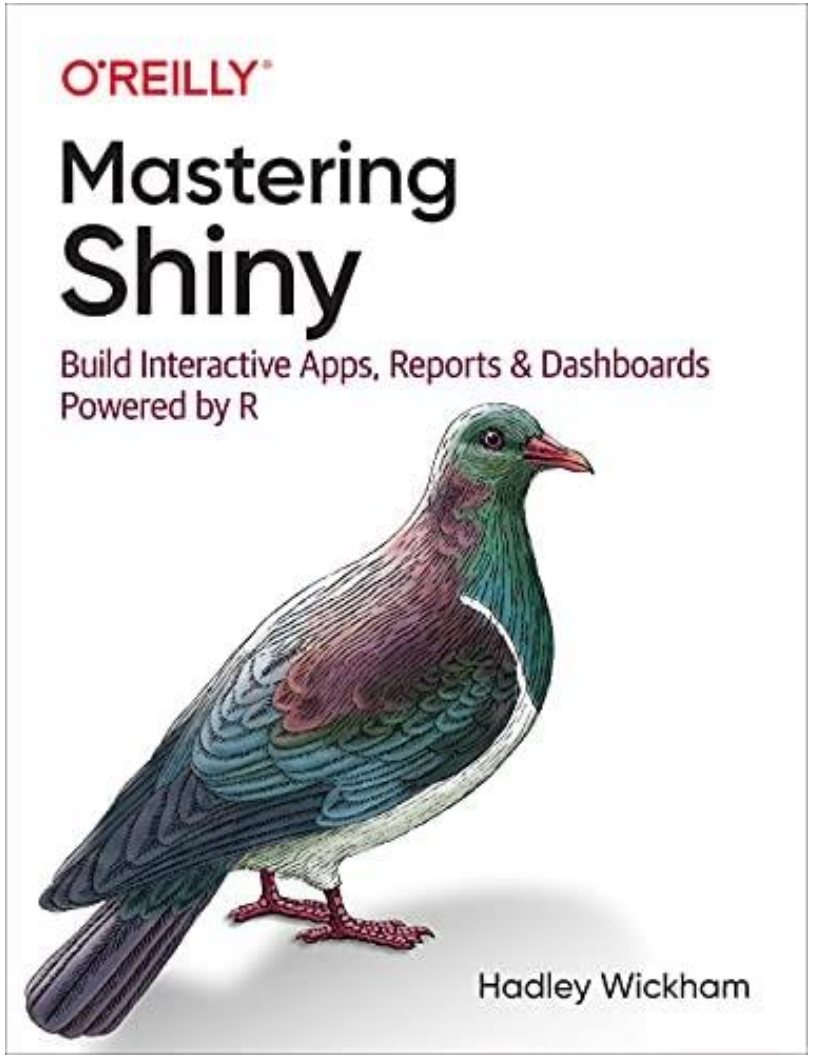
```
ui <- fluidPage(

        sidebarLayout(
            sidebarPanel(
                    sliderInput(inputId = "num",
                        label = "Choose a number",
                        val = 25, min = 1, max = 100)
            ),

            mainPanel(
                    plotOutput("my_plot")
            )

      ) # end sidebarLayout
)   # end ui
```

Application title

```
ui <- fluidPage(

    titlePanel("My cool Shiny app!"),

     sidebarLayout(
          sidebarPanel(
               sliderInput(inputId = "num",
                    label = "Choose a number",
                    val = 25, min = 1, max = 100)
          ),
          mainPanel(
               plotOutput("my_plot")
          )

    ) # end sidebarLayout
)  # end ui
```

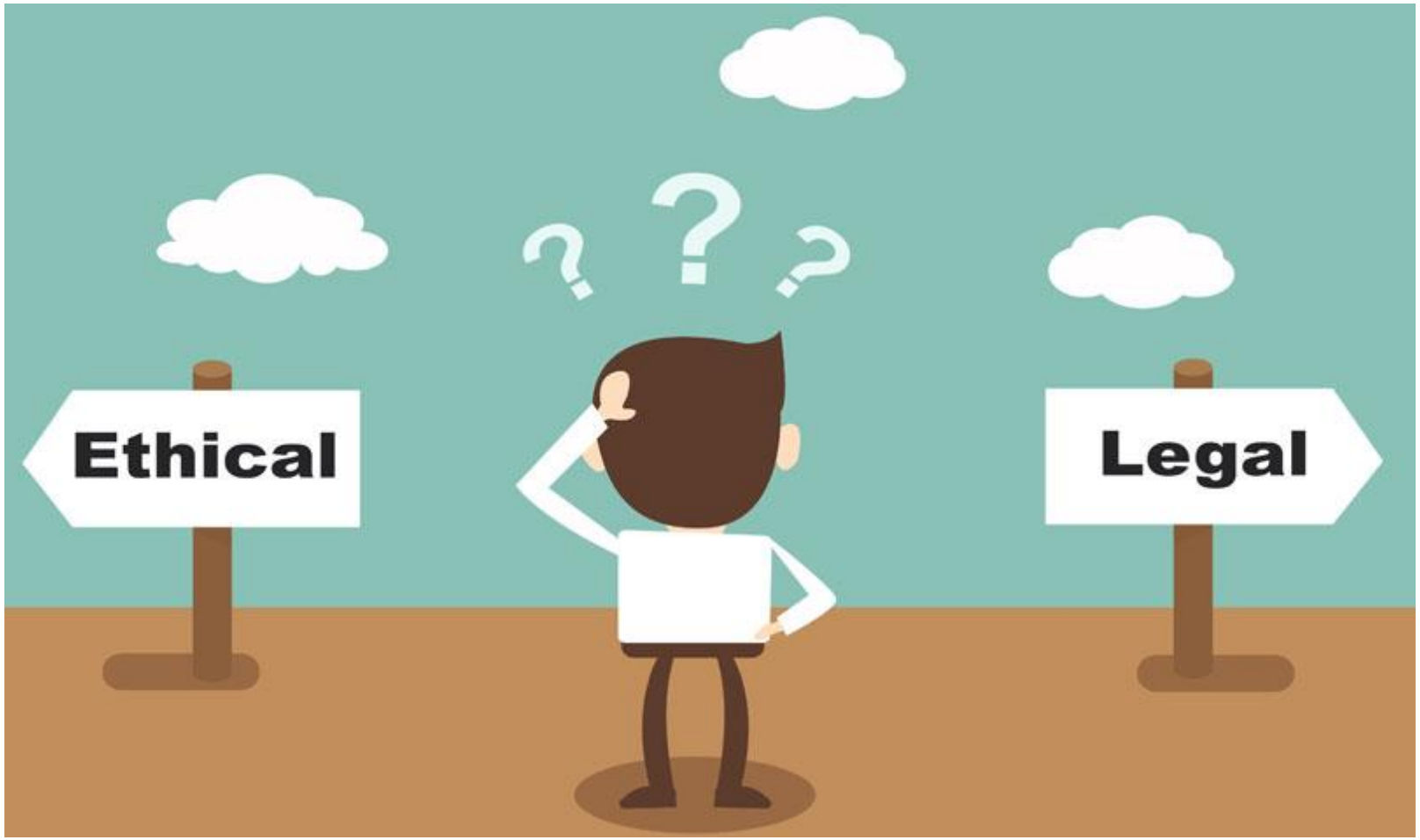
You can add a title to your app





https://mastering-shiny.org/

Ethics in Statistics and Data Science



Ethics in Data Science

Ethics of:

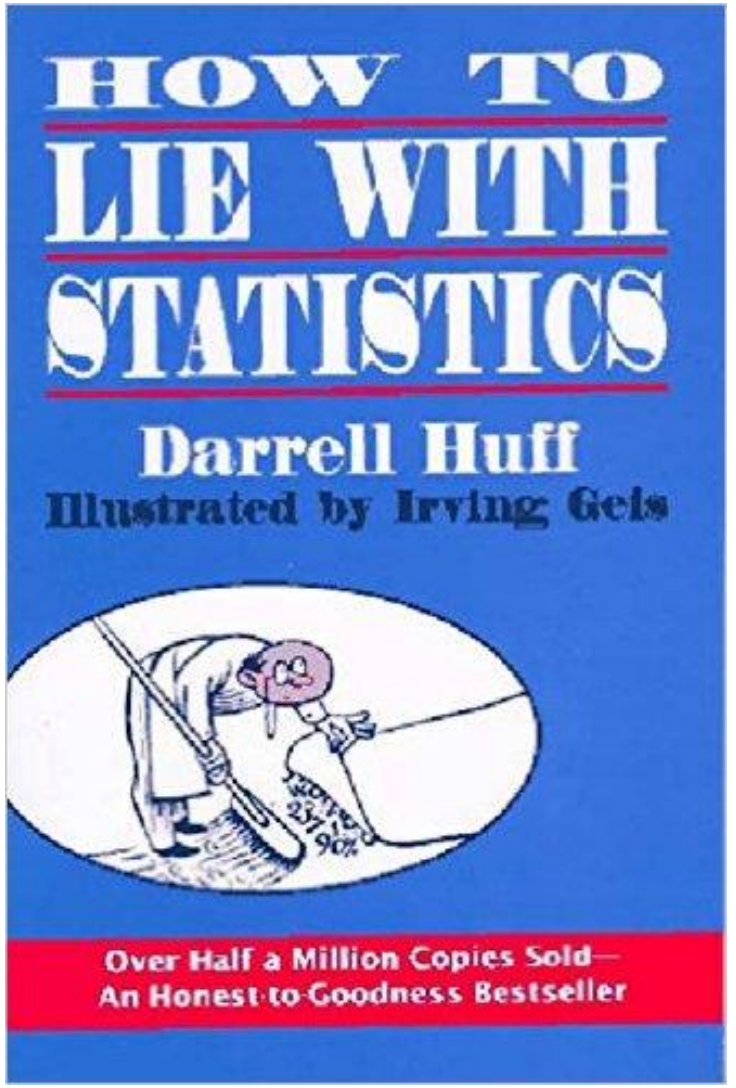
1. Data presentation
2. Using valid data
3. Data scraping TOS and privacy
4. Reproducibility
5. Citations/peer review
6. Disclosure
7. Ethics in Statistical analyses
8. Ethics of creating powerful tools

1. Ethics of data presentation

Data should be displayed in an honest way that gives an accurate picture of trends

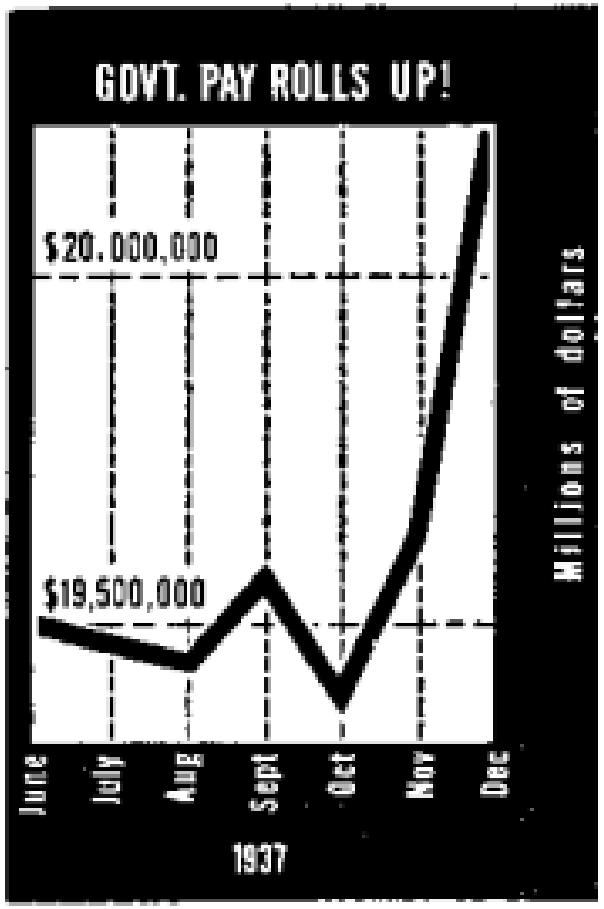
Darrell Huff wrote a classic book in the 1950's pointing out ways that people lie with statistics

[The book was banned as training material at the VA]



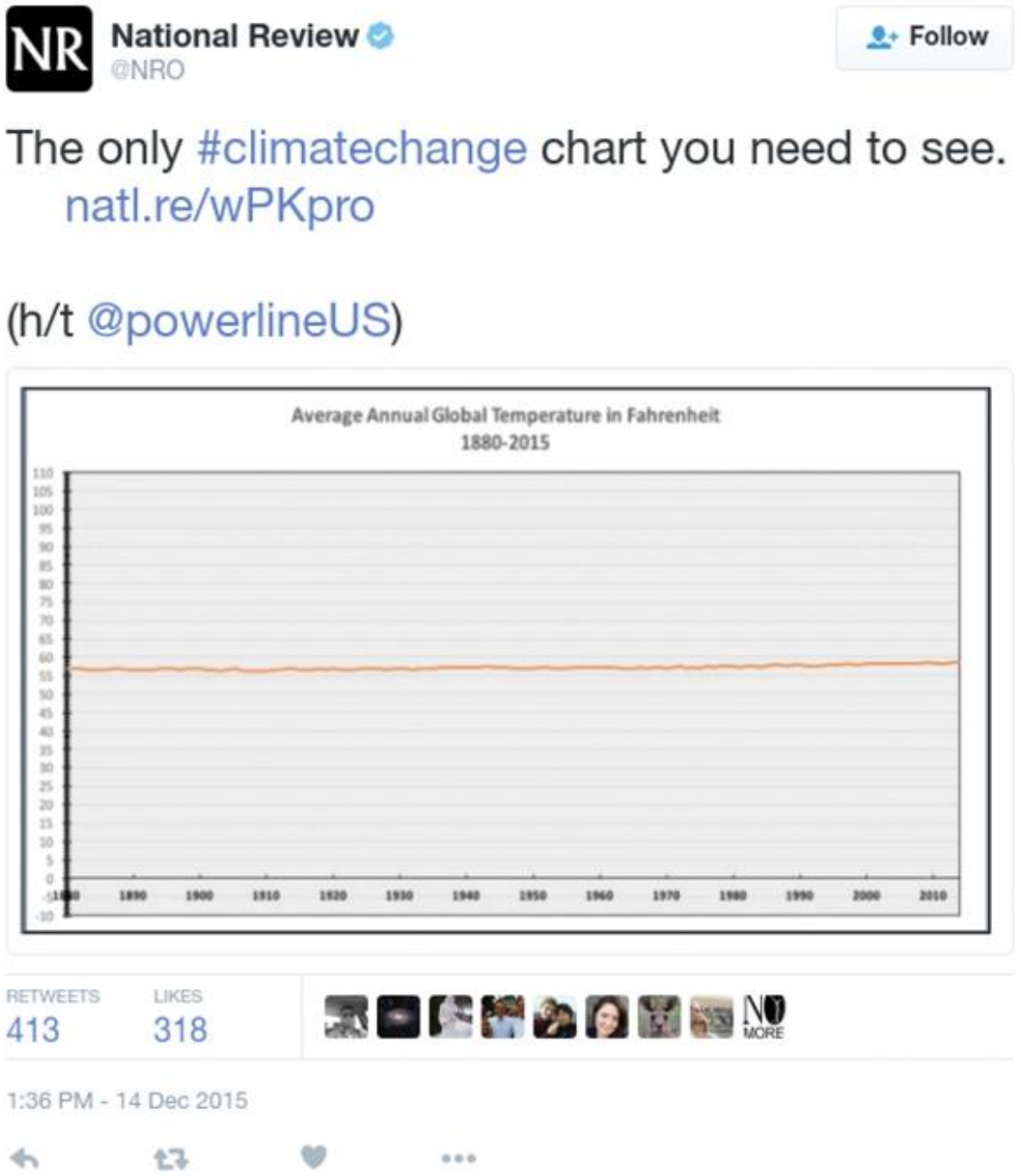
Ethics of data presentation

What is potentially misleading with this figure?



From a 1938 article in Dun's Review titled 'GOVERNMENT PAY ROLLS UP!'

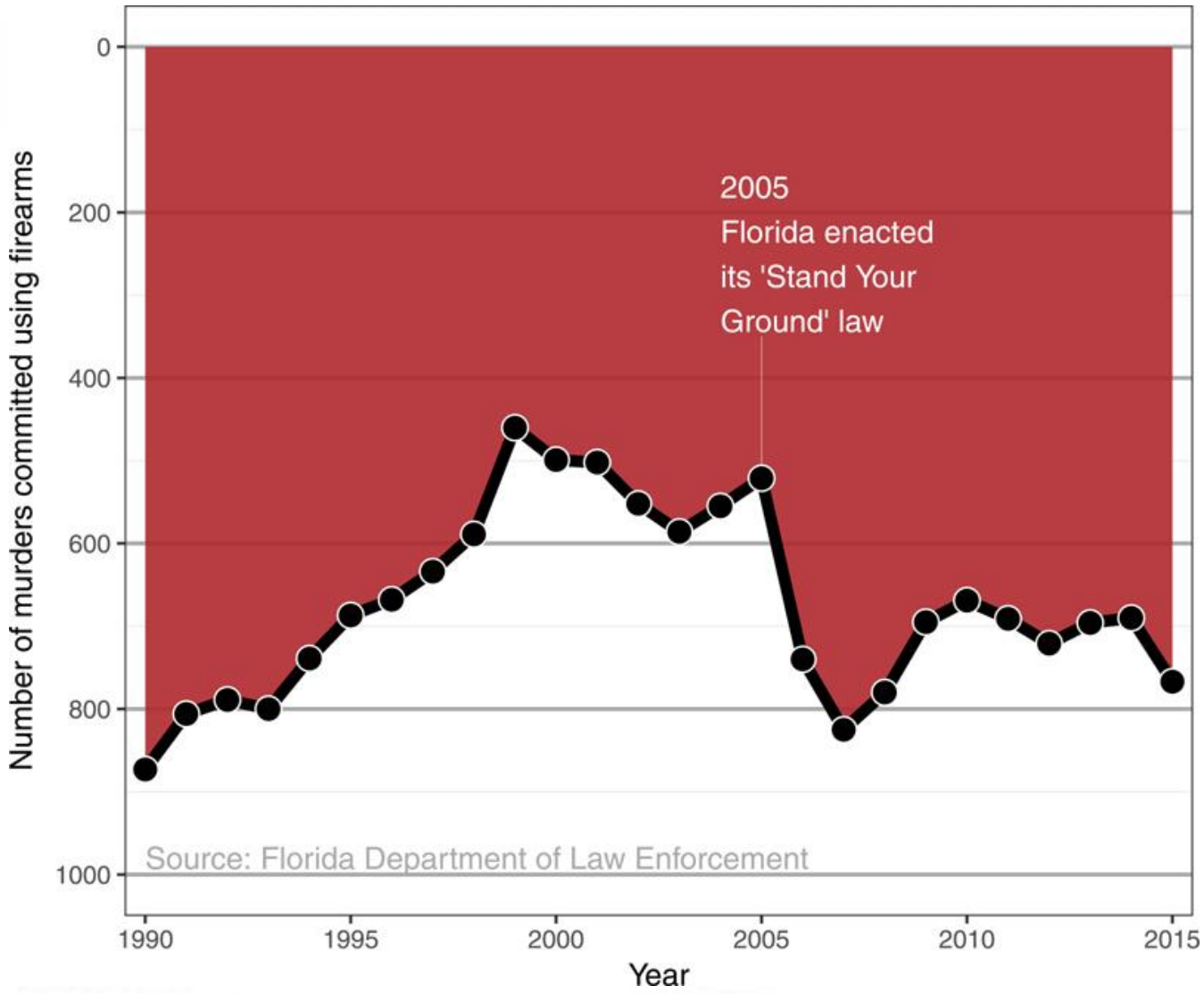
How much has the climate changed?



Did ‘Stand Your Ground’ decrease murder by firearms?

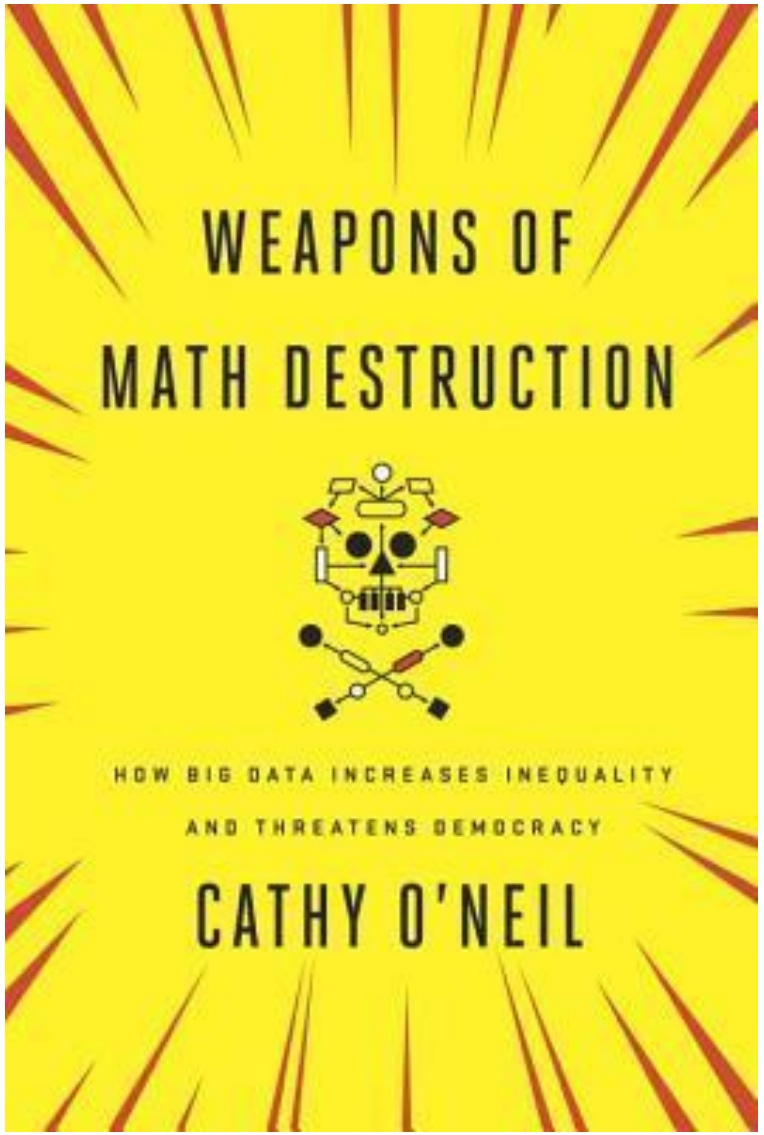
What is misleading with this figure?

The axes are going in the wrong direction



To learn more…

Take S&DS 150: Data Science Ethics



What are PCA and clustering?

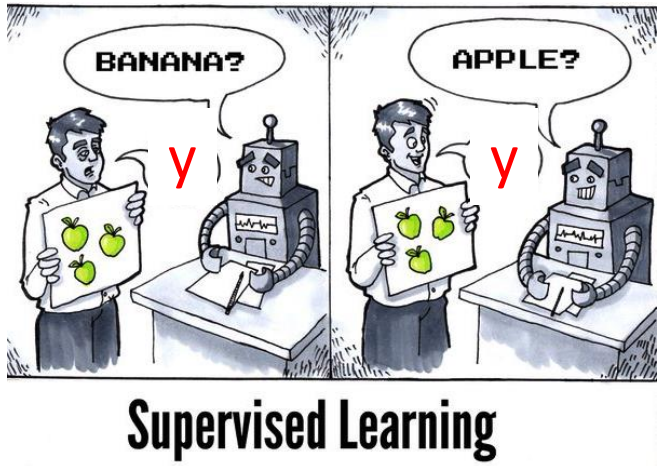
Unsupervised learning

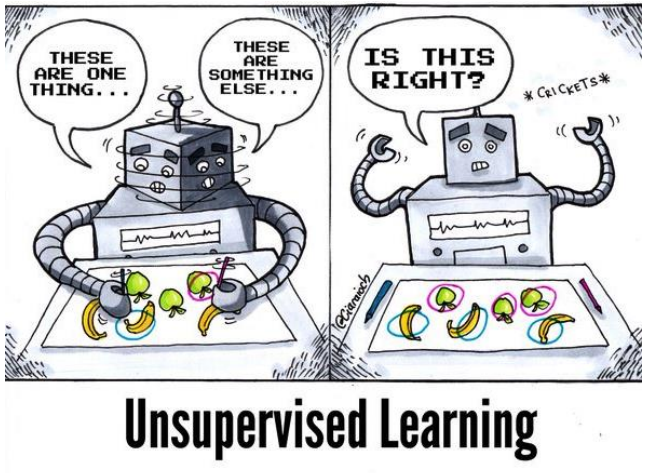
In unsupervised learning we have explanatory variables x_i's, but no response variable y:

Examples:

1. **Dimensionality reduction** where we try to find a smaller set of explanatory variables that captures most of the variability in the data
   - Principal component analysis (PCA)

2. **Clustering** where we try to group similar data points together





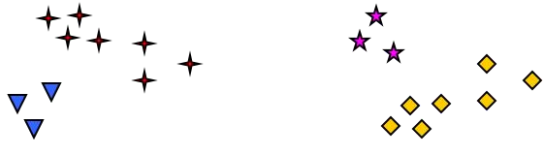
Clustering

So tell me how many clusters do you see?

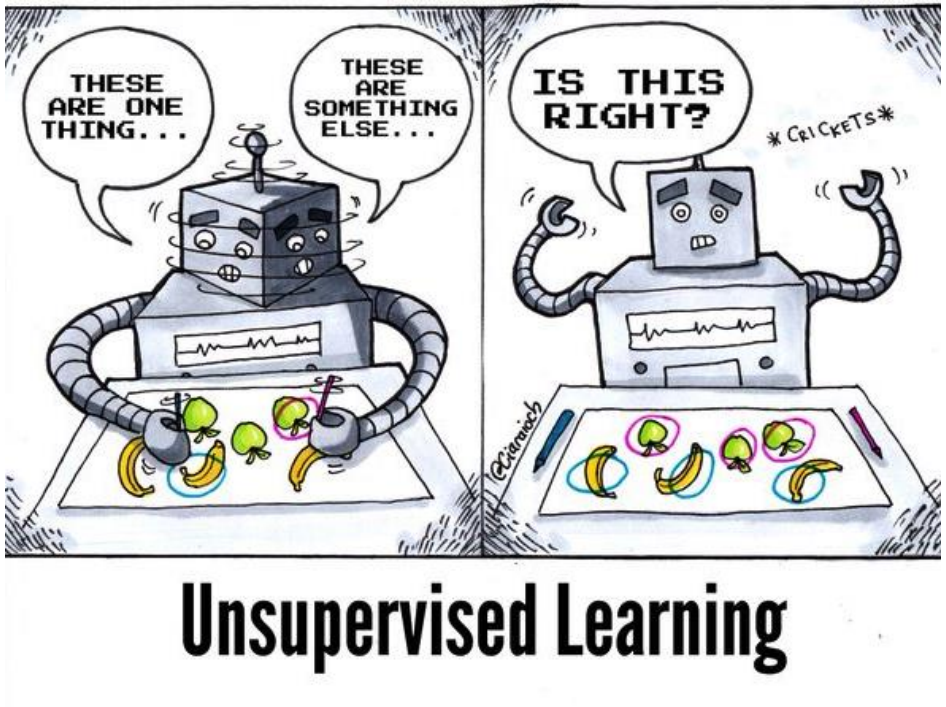
How many clusters?

Six Clusters

Two Clusters

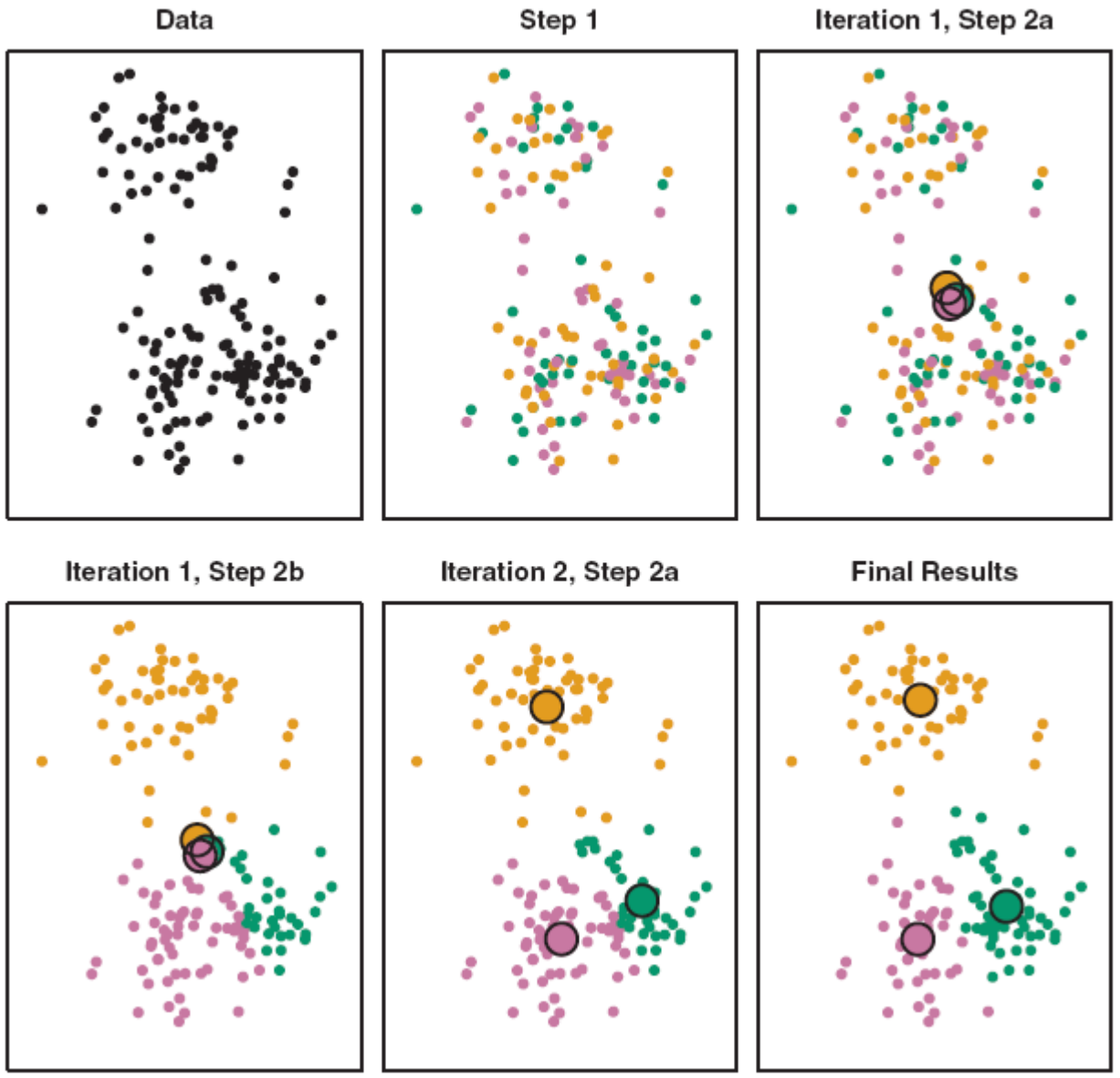


Four Clusters



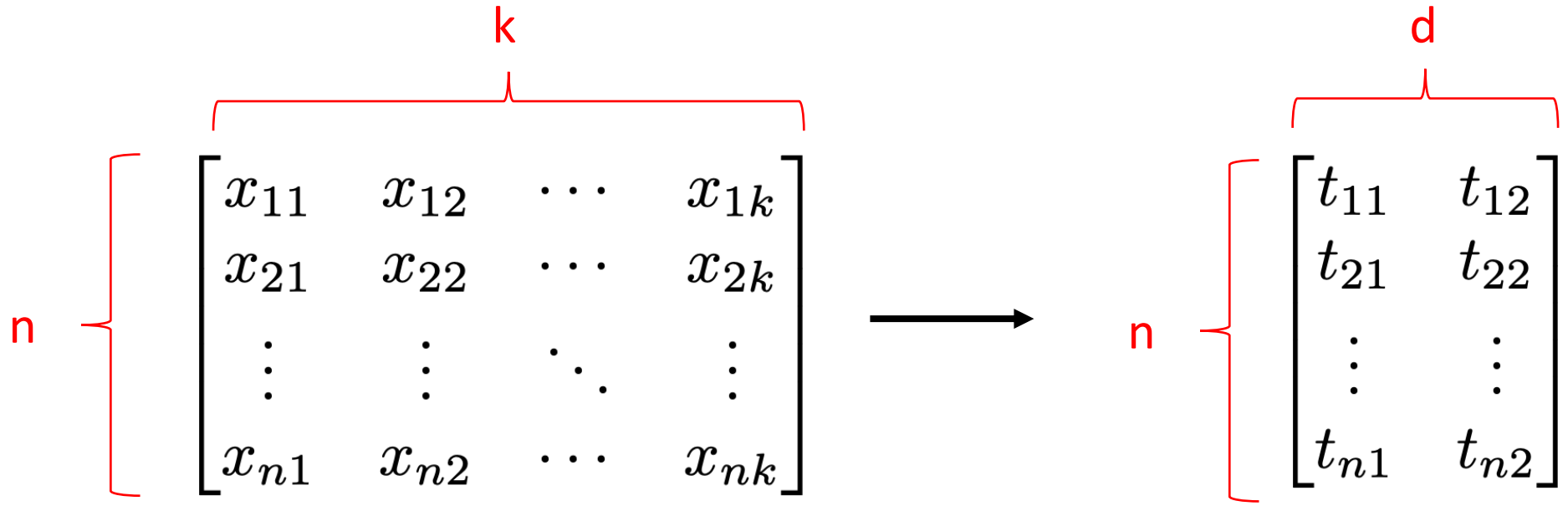
K-means clustering

1. Randomly assign points to clusters C_k
2. Calculate cluster centers as means of points in each cluster
3. Assign points to the closest cluster center
4. Recalculate cluster center as the mean of points in each cluster
5. Repeat steps 3 and 4 until convergence

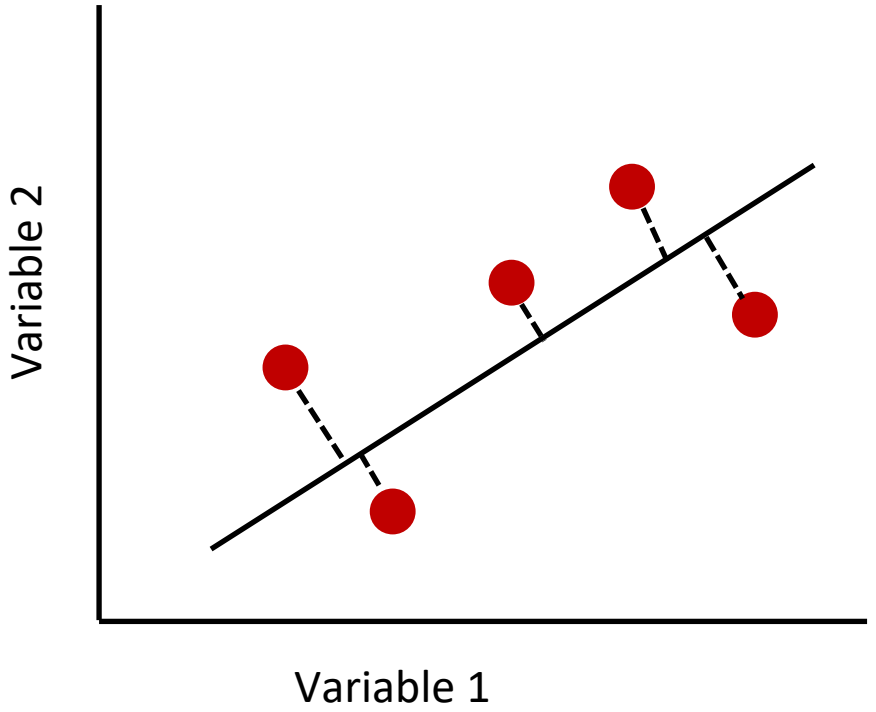


Principal Component Analysis

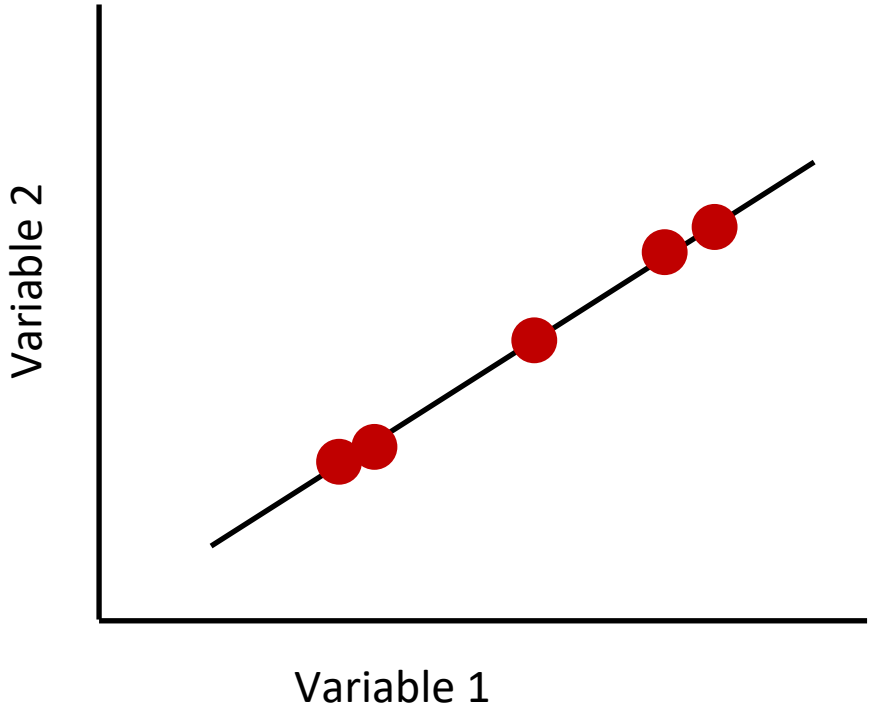
Principal Component Analysis is a dimensionality method that tries to capture most of the variability in the original data



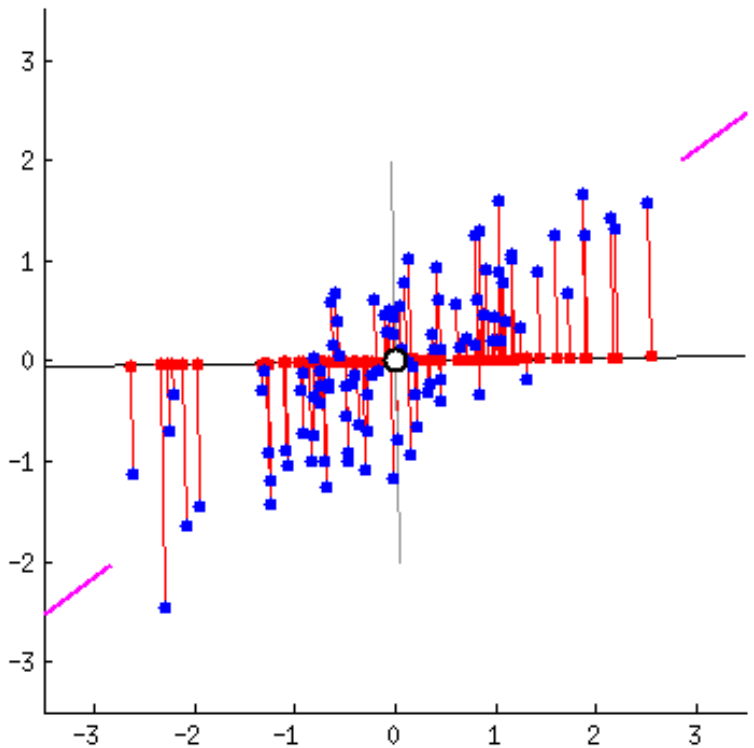
Principal Component Analysis

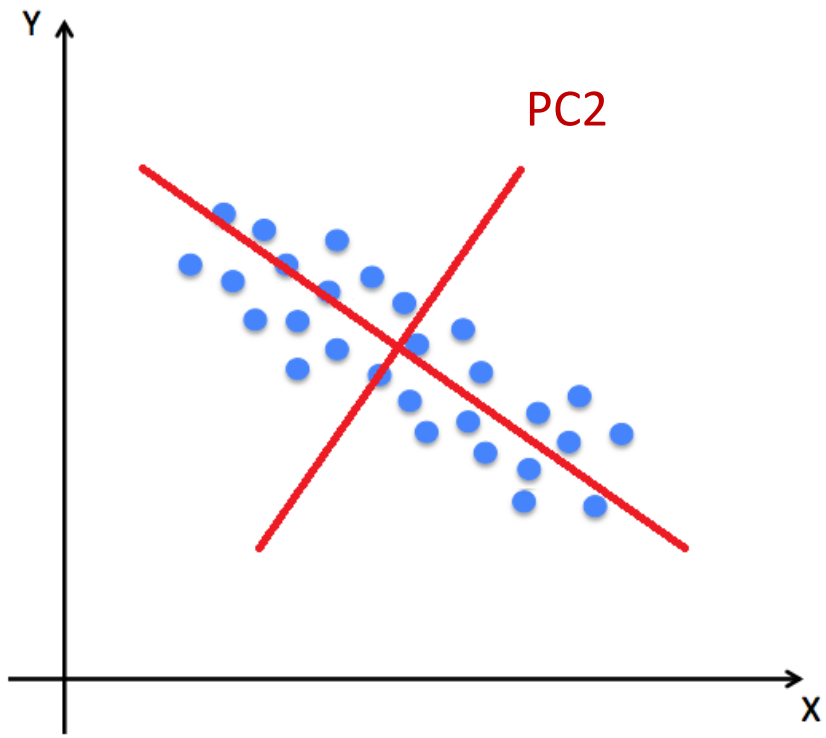


Principal Component Analysis



Principal Component Analysis





Questions?

Wrap up and conclusions

Topics we will cover

R and descriptive statistics/plots: Base R, fundamental concepts in Statistics

Review confidence intervals: Sampling and bootstrap distributions

Review of hypothesis tests: Permutation and parametric tests, theories of testing

Data wrangling: filtering and summarizing data, joining data sets, reshaping data

Data visualization: grammar of graphics, mapping

Regression: simple/multiple, non-linear terms, logistic regression

ANOVA: one-way/factorial, interactions

Statistical learning: cross-validation, logistic regression, PCA, clustering

Course objectives

Extend and solidify concepts and method learned in intro stats

Learn how to use the R programming language to analyze, visualize and wrangle data

Gain experience extracting insights from real data

Learn how to find patterns in a large noisy data sets and convincingly convey the results to others!

Next steps

Take more advanced Statistics and Data Science classes offered at Yale!

There are many good online resources to learn more R

Good luck with the end of the semester!

Good luck finishing your final projects!

The final exam is on Monday December 19 at 7pm in LC 101