Introduction to Data Science Using R

Lecture 2

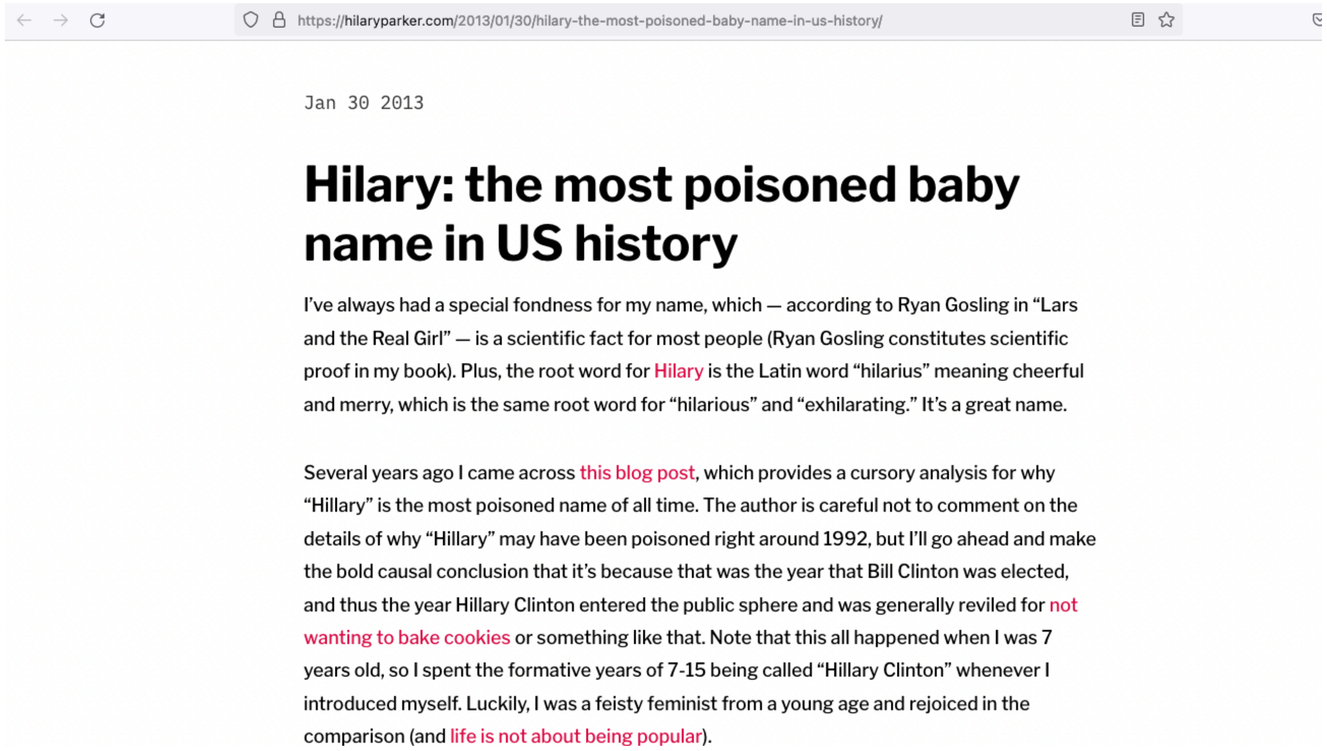
Ben Rogers

September 8, 2022

Parts of a Data Science Project

- Forming the question
- Finding or generating the data
- Analyzing the data
    - Exploring the data
    - Modeling the data
- Communicate to others

An example



https://hilaryparker.com/2013/01/30/hilary-the-most-poisoned-baby-name-in-us-history/

An Example

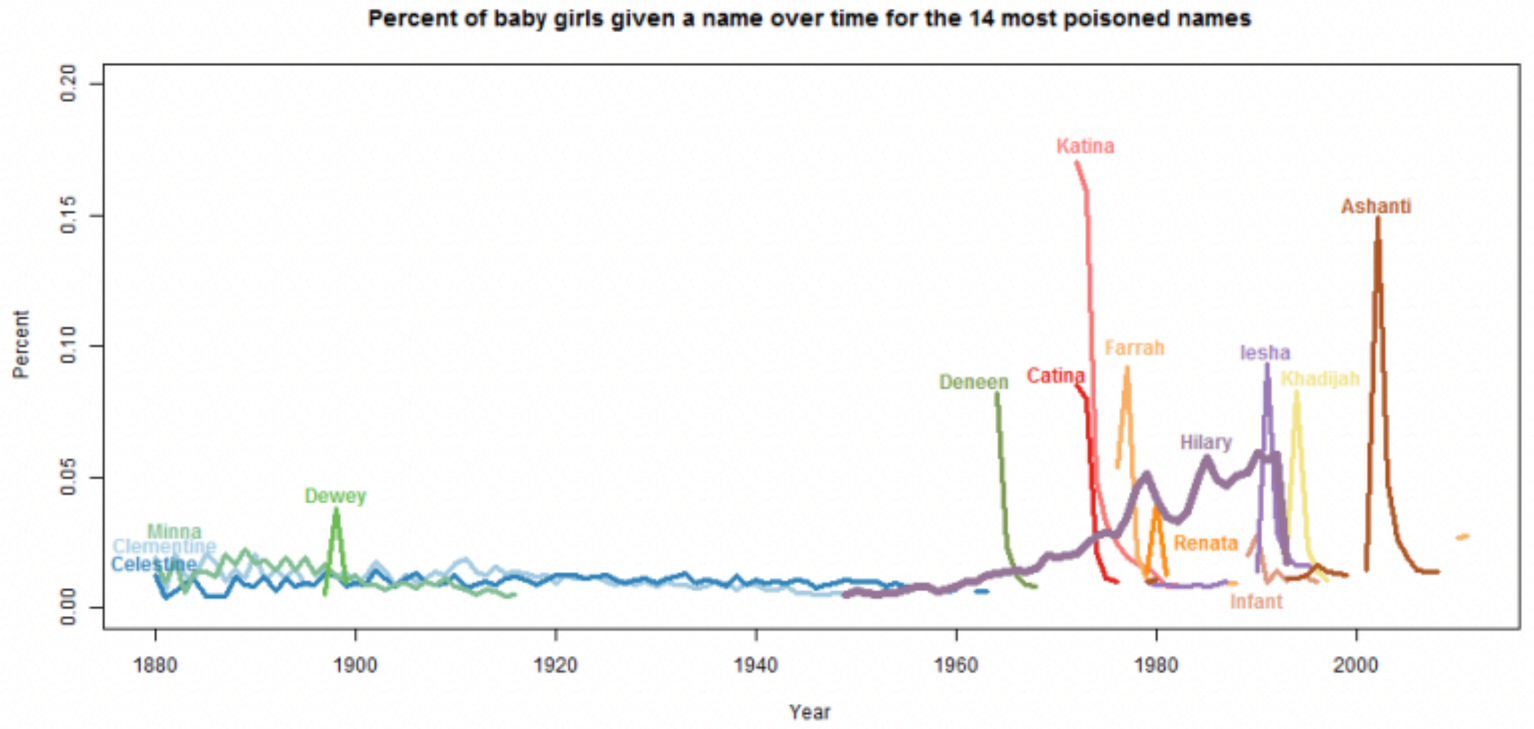
- The Question:
    - Is Hilary/Hillary the most rapidly poisoned name in recorded American history?
- The Data:
    - Collected from the Social security website: 1000 most popular baby names from 1880-2011

The Analysis

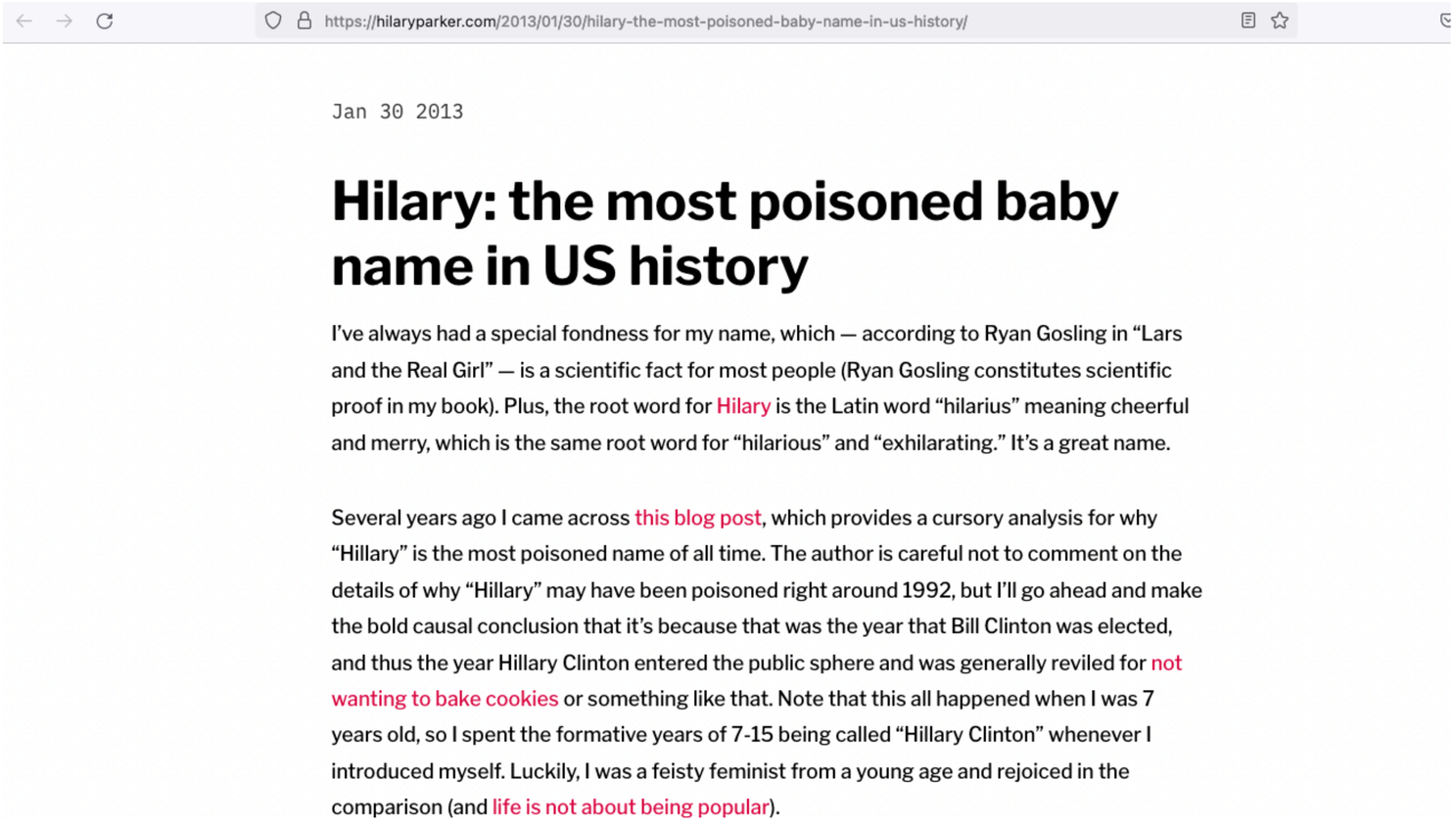
- Calculated relative risk for each year versus the year before.

Name	Loss (%)	Year
Farrah	78	1978
Dewey	74	1899
Catina	74	1974
Deneen	72	1965
Khadijah	72	1995
Hilary	70	1993
Clementine	69	1881
Katina	69	1974
Renata	69	1981
Iesha	69	1992
Minna	68	1883
Ashanti	68	2003
Celestine	67	1881
Infant	67	1991

Analysis


Percent of baby girls given a name over time for the 14 most poisoned names
Percent
0.00
0.05
0.10
0.15
0.20
1880
1900
1920
1940
1960
1980
2000
Year
Minna
Clementine
Celestine
Dewey
Deneen
Katina
Catina
Farrah
Hilary
Iesha
Khadijah
Renata
Infant
Ashanti

Communicate



https://hilaryparker.com/2013/01/30/hilary-the-most-poisoned-baby-name-in-us-history/

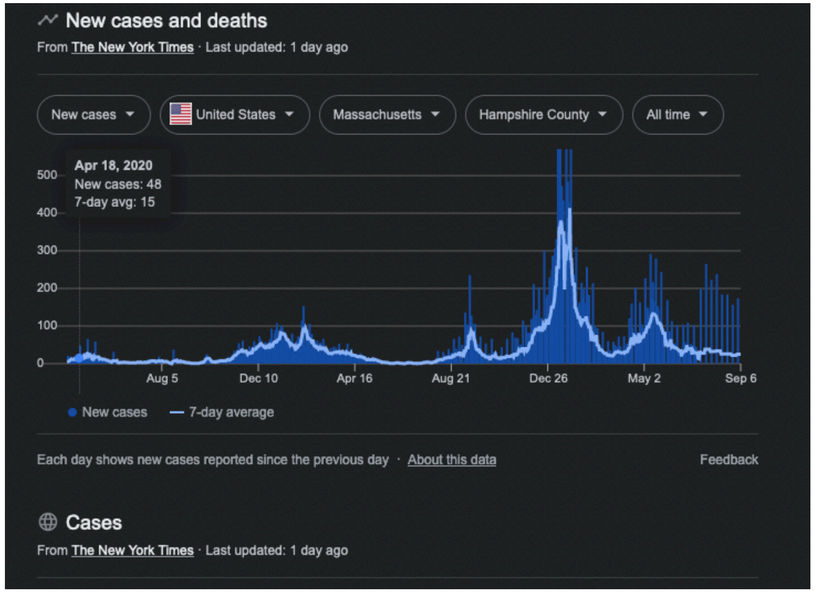
Types of Data Science Questions

1. Descriptive
2. Exploratory
3. Inferential
4. Predictive
5. Causal
6. Mechanistic

Types of Data Science Questions

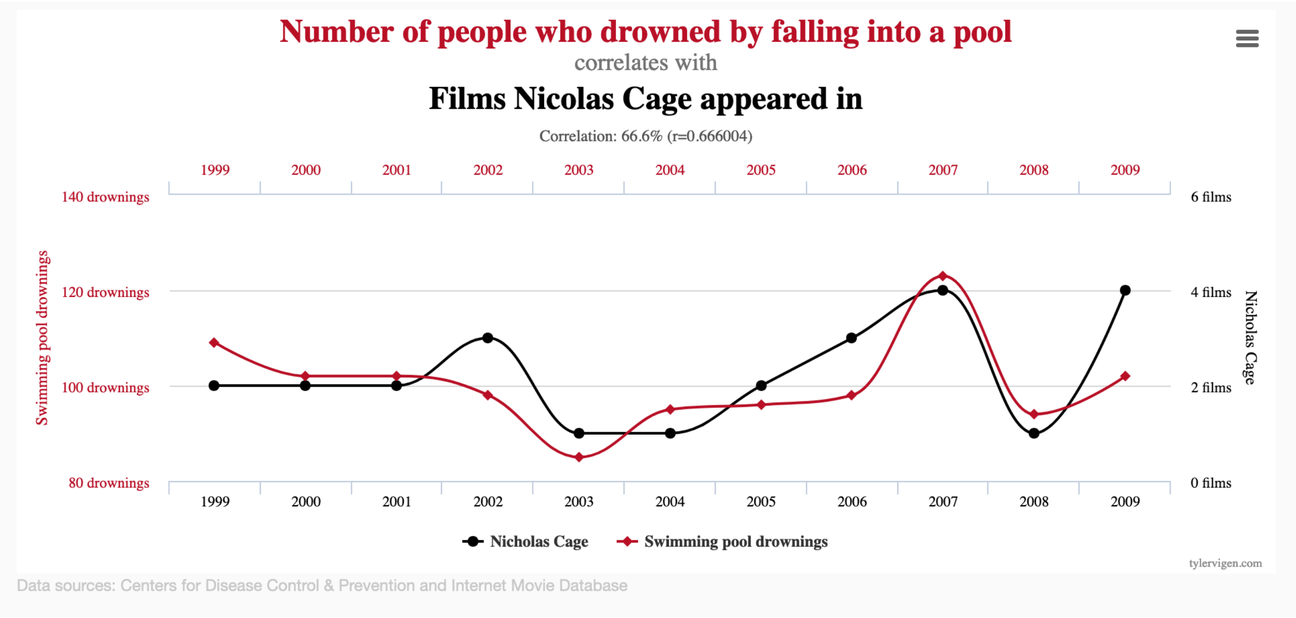
Descriptive Analysis

- Describe or summarize a set of data
  - Descriptive statistics (mean, median, mode, range, variance, etc.)
- NOT to generalize results to a larger population or draw conclusions.



Exploratory Analysis

- Explore data and find relationships between variables.
- Do **not** confirm the relationships as causative.

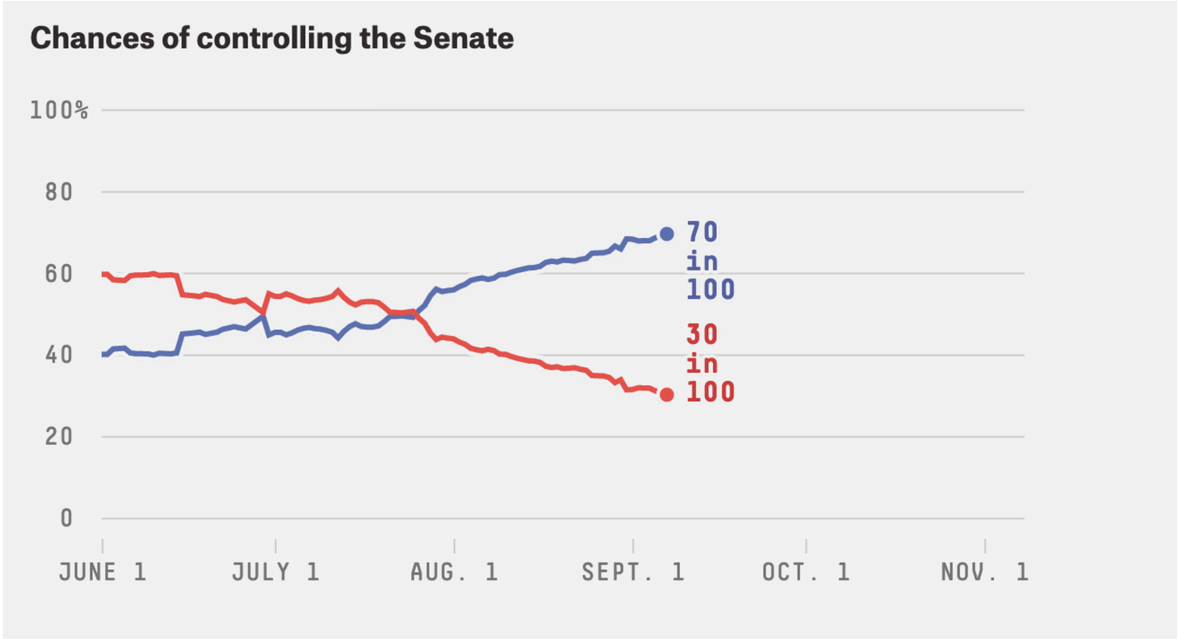


Inferential Analysis

- Use a relatively **small sample** to draw **inference** about a **population**.
- Example: political polling

Predictive Analysis

- Use **current** data to make **predictions** about the **future.**



https://projects.fivethirtyeight.com/2022-election-forecast/senate/?cid=rrpromo

Causal Analysis

- The goal of causal analysis is to determine the **effect** of one variable on another variable.
- Complicated by the presence of confounding variables.
- Can be very difficult to do without a randomized controlled trial.

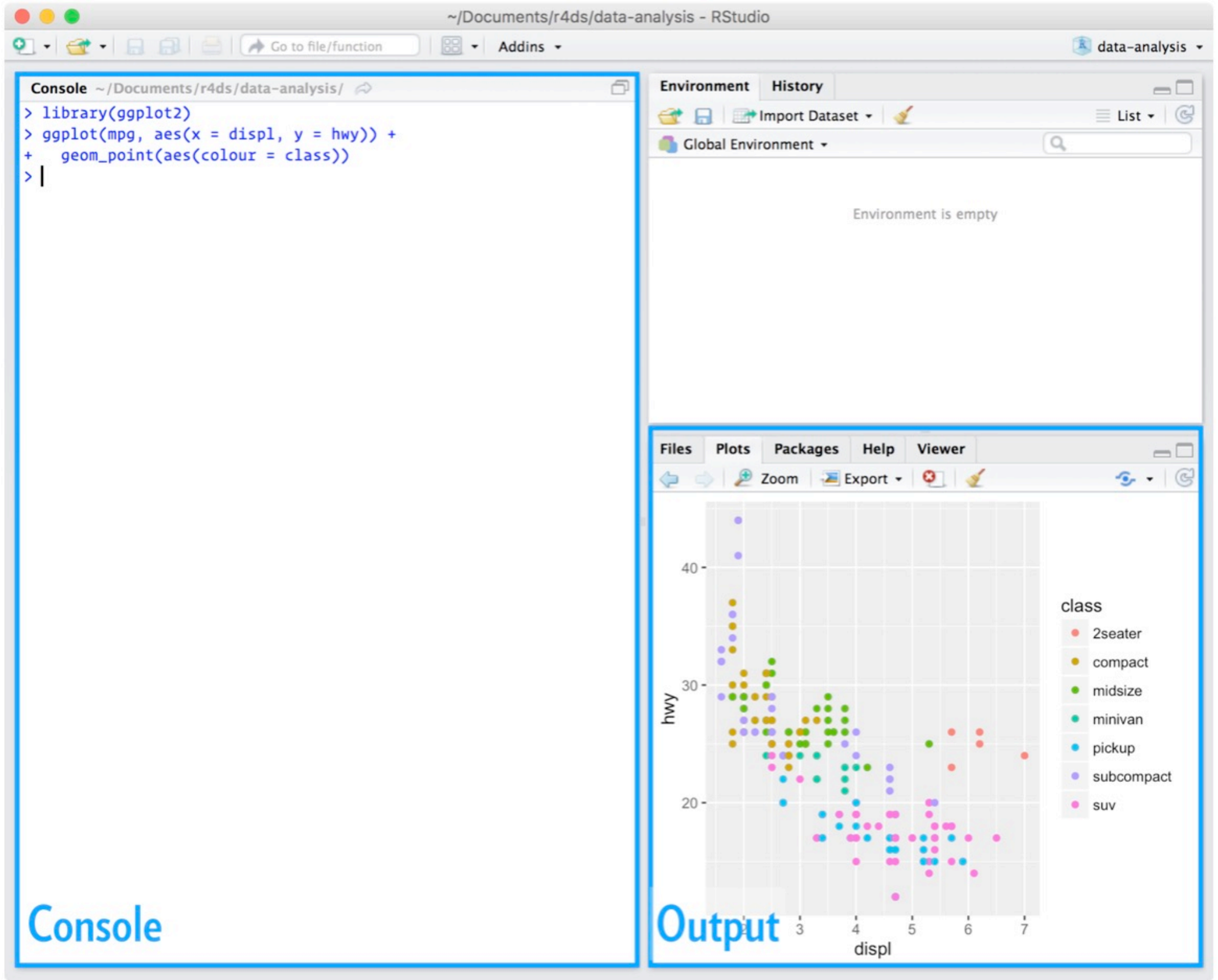
Mechanistic Analysis

- Quantify how **exact changes** in one variable lead to **exact changes** in other variables.
- Less common, often used in other disciplines such as physics/engineering
- Generally carefully controlled with the only noise being measurement error.

Introduction to R

R: Do not open this

RStudio: Open this


~/Documents/r4ds/data-analysis - RStudio
Go to file/function
Addins
data-analysis
Console ~/Documents/r4ds/data-analysis/
> library(ggplot2)
> ggplot(mpg, aes(x = displ, y = hwy)) +
+ geom_point(aes(colour = class))
>
Console
Environment
History
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Zoom
Export
hwy
displ
class
2seater
compact
midsize
minivan
pickup
subcompact
suv
Output

R Basics

- Object oriented
- Open source
- Free!

Data types

- Integers: values like -1, 2, 1024
- Doubles: a larger set of values containing both the integers, but also fractions and decimal values like -0.8, 20.15
- Logical: either TRUE or FALSE
- Characters/strings: text such as “Cabbage”, “UMass”, “I am coding in R”
- Both integers and doubles are called numerics. You can use the function `typeof()`
- **Your turn:**
    - What are the types of “1” “one” one 1

Your turn

Types of data

- What are the types of:
    - “1”
    - “One”
    - One
    - 1

Objects

- Objects are data saved in R
- You can create new objects with `<-`
  - Shortcut: **option** + **-** on Mac, **alt** + **-** on PC
- For example, `x <- 3`
- All objects have the same form: `object_name <- value`
- I recommend using `<-` for assigning values instead of `=`
- `=` has two meanings in R:
  - Assignment operator, the same as `<-`
  - Syntax token for passing arguments to functions, example: `ggplot(data = mpg)`

Your Turn

Create an object in R

- Creating object names is an art
  - You want your object names to be descriptive, but not too long
  - Recommend having a convention for longer names: some.people.use.dots and someUseCamelCase
- x <- 3
- this_is_a_long_name <- 2.5
- To inspect this object, try out RStudio's completion facility
  - Type "this", keep adding characters until you have a unique prefix, press return

Using R as a calculator

Try running:

```
1/100*20
```

```
1+2+4
```

```
pi^2
```

Other Useful Functions

- Try

```
100 %/% 30
```

```
100 %% 30
```

This can be useful for extracting hours and minutes from time data for example.

```
1350 %/% 100
```

```
1350 %% 100
```

Your turn

- Consider the format hhmm for the time. How many minutes elapsed from 240 to 1530? (Hint: first calculate the total minutes of 240 and 1530 counted from midnight)
- `(1530%/%100*60 + 1530%%100) - (240%/%100*60 + 240%%100)`

Relational Operators

`x < y`	Less than
`x > y`	Greater than
`x == y`	Equal to
`x <= y`	Less than or equal to
`x >= y`	Greater than or equal
`x != y`	Not equal to
`x %in% y`	Group membership

Run the following

```
1 < 1

1 <= 1

1 == 1

1 != 1
```

Logical Operators

- Logical operators are used to carry out Boolean operations like AND, OR, etc. Zero is considered FALSE and non-zero numbers are taken as TRUE.

```
! 3>2

2>1 & 3<2

2>1 | 3<2
```

Your turn

- Try the following code. Are you surprised by the result? Think about why.

```
sqrt(4)^2 == 4

sqrt(2)^2 == 2
```

Vectors

- A series of values, for example (3, 2, 5, 9)
- Created using `c()` where c stands for “combine” or “concatenate”
- R is very good at dealing with vectors. The arithmetic operators introduced before can be used for vectors.

```
x <- c(1,2,3,4,5)
x/2
x^2
x %% 2
2/x
y <- c(6,7,8,9,10)
x/y
x^y
c(1,2,3) < c(3,2,1)
```

Vectors

Your turn

Run the following code and explain the result. Do some experimentation to verify your thought.

```
x <- c(2,4,6,8)

y <- c(1,2)

x/y
```

Vectors

Your turn

Run the following code and explain the result. Do some experimentation to verify your thought.

```
x <- c(2,4,6,8)

y <- c(1,2)

x/y

[1] 2 2 6 4
```

Subsetting a vector

Positive integers return elements at the specified positions

```
> letters
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s" "t" "u"
[22] "v" "w" "x" "y" "z"
> letters[2]
[1] "b"
> letters[c(3,1,5)]
[1] "c" "a" "e"
```

Negative integers omit elements at the specified positions

```
> letters[-1]
 [1] "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
> letters[-c(1,3)]
 [1] "b" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
```

Subsetting a vector (Continued)

Logical vectors select elements where the corresponding logical value is TRUE

```
> a <- 1:4
> a[c(TRUE,FALSE,TRUE,FALSE)]
[1] 1 3
> a[a > 2]
[1] 3 4
```

Subsetting a vector

Your Turn

- Use subsetting to create a vector with the 5th, 10th, 15th, 20th and 25th letters.

Subsetting a vector

Your Turn

- Use subsetting to create a vector with the 5th, 10th, 15th, 20th and 25th letters.

```
> letters[c(5, 10, 15, 20, 25)]
[1] "e" "j" "o" "t" "y"
```

```
> letters[5*1:5]
[1] "e" "j" "o" "t" "y"
```

Have a nice weekend!