Ve572 Lecture 2

Manuel and Jing

UM-SJTU Joint Institute

May 17, 2018

Definition

Correlation is any statistical relationship between two variables. It often, but not always, refers to how close two variables are to having a linear relationship.

- For example,
  - People's height and their shoe size
  - Your letter grade and the number of honour code violations

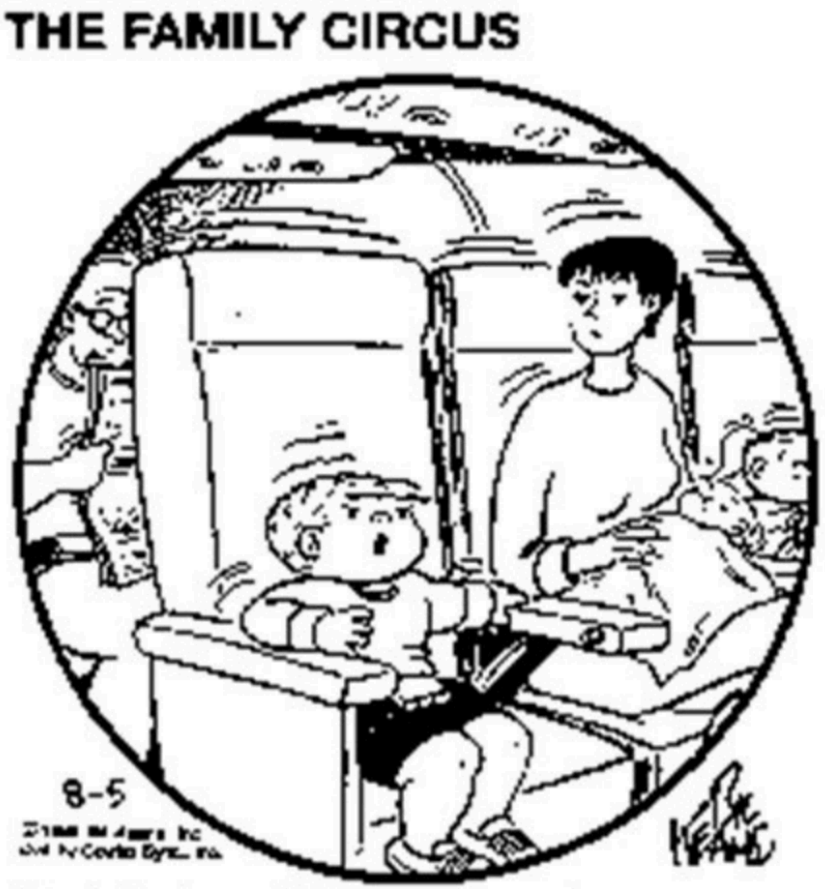
Definition

Causation is the relationship between cause and effect.

- For example,
  - If I stab you with a knife, you will bleed.
  - If you complain to the police about this stab, I will be arrested.
- Notice there is a difference between correlation and causation.

Q: Why can we no conclude A causes B from knowing A and B are correlated?

Reverse Causation

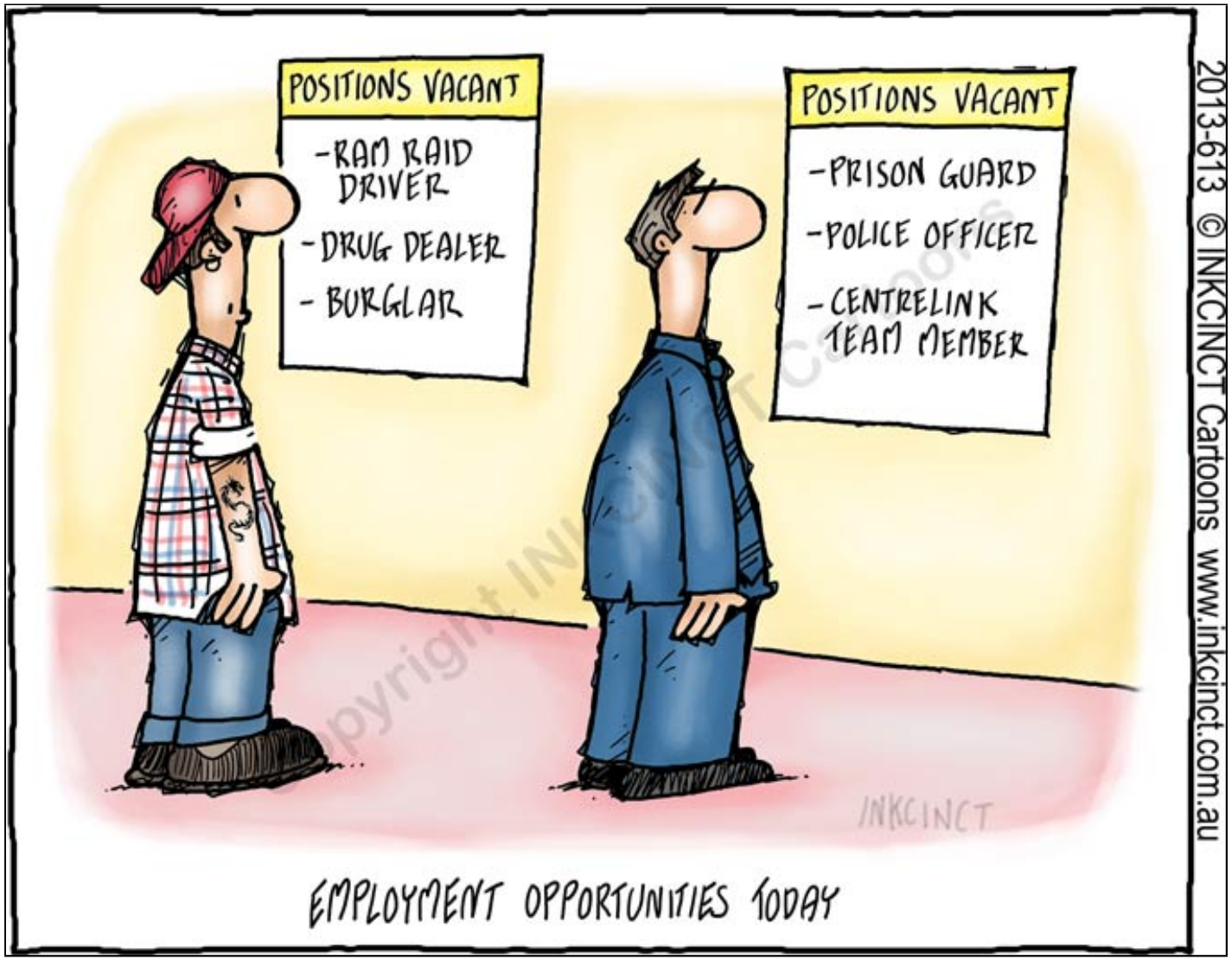


"I wish they didn't turn on that seatbelt sign so much! Every time they do, it gets bumpy."

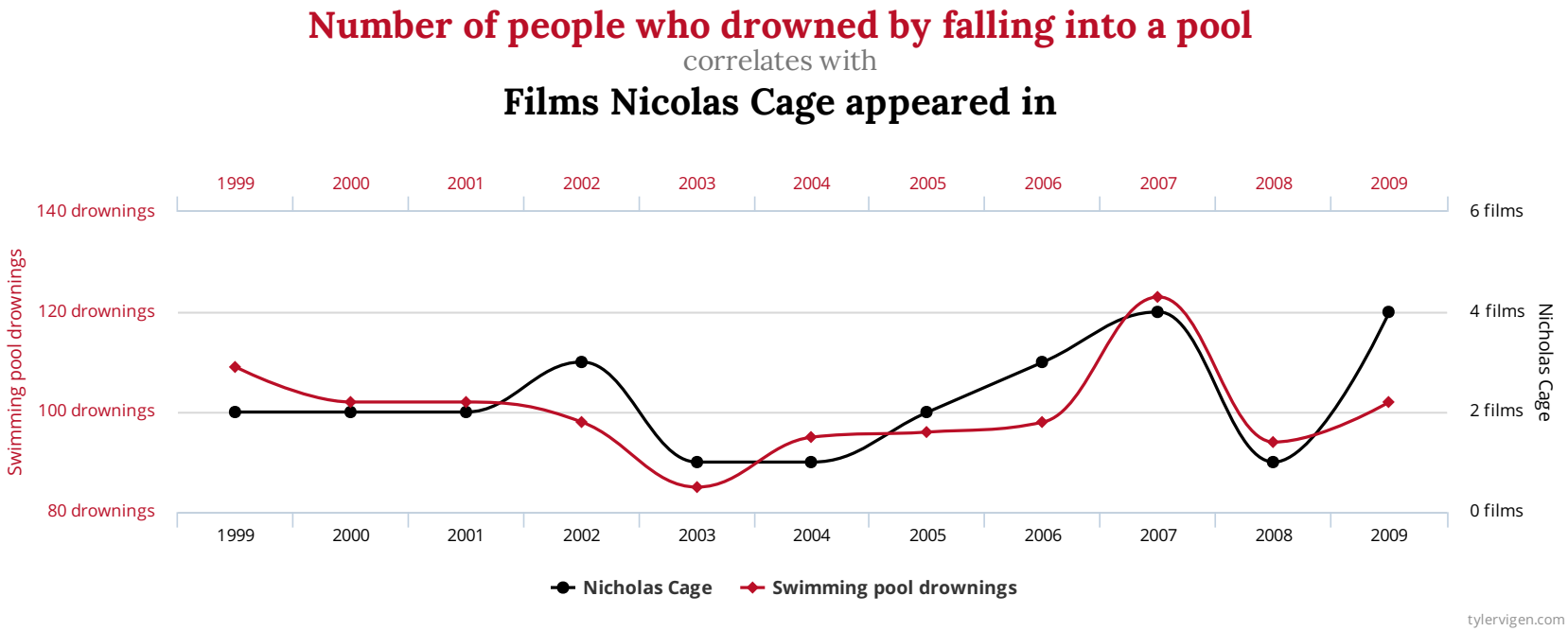
- Usually the homeless population and the crime rate in an area are correlated.

It is not because homeless people are committing crimes, or having more crimes is causing people to be homeless.

Unemployment is the Confounding Variable here!



Spurious while Entertaining correlations

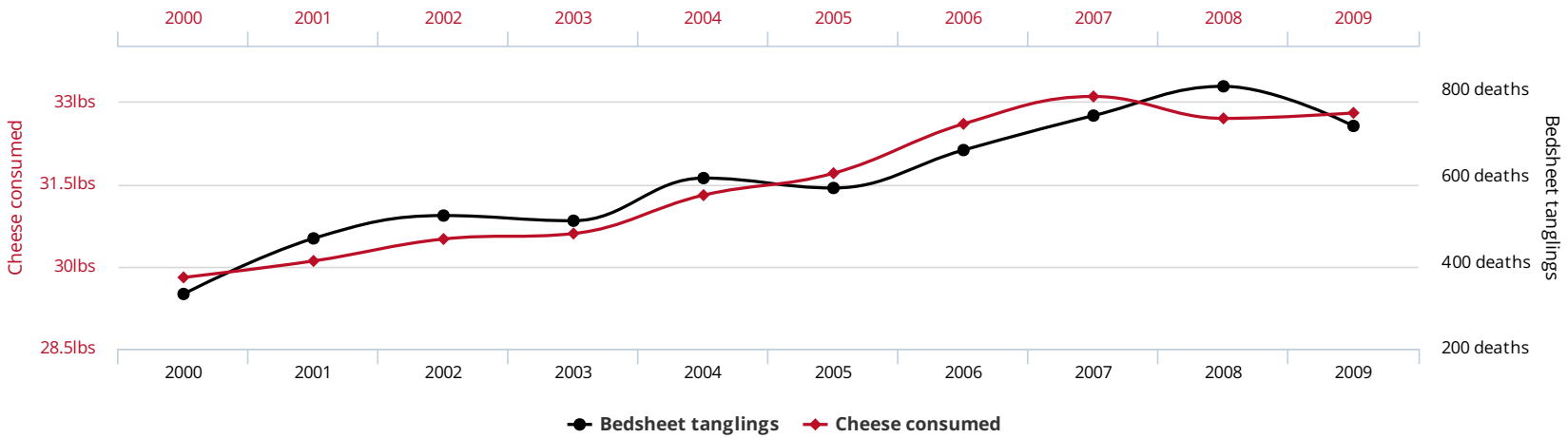


I do not think there is any causal relationship between the two variables, not even a confounding variable.

Per capita cheese consumption

correlates with

Number of people who died by becoming tangled in their bedsheets

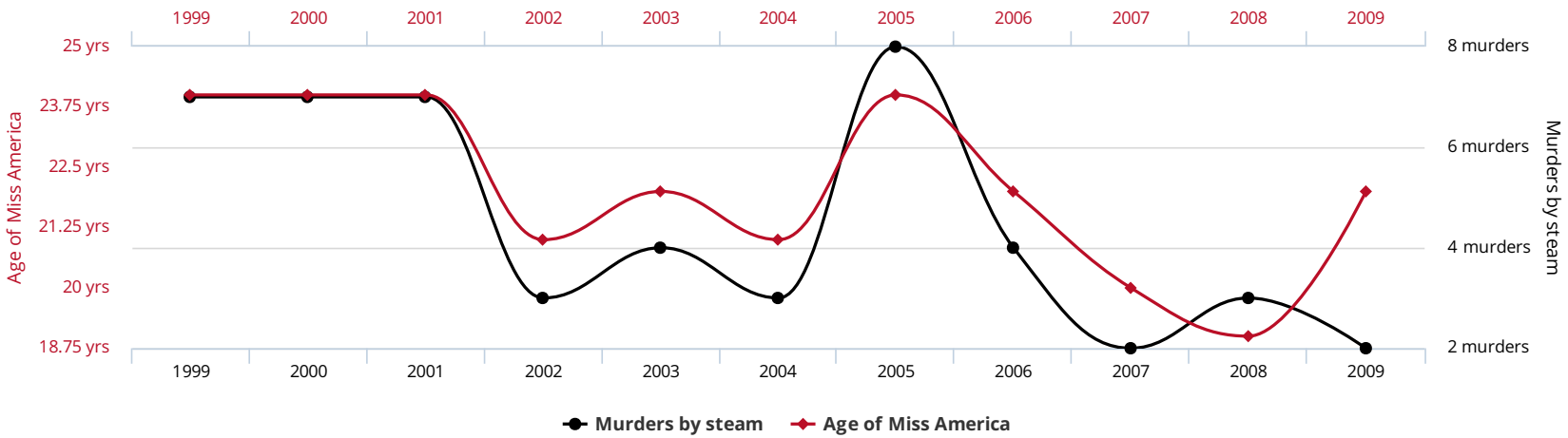


tylervigen.com

Age of Miss America

correlates with

Murders by steam, hot vapours and hot objects



tylervigen.com

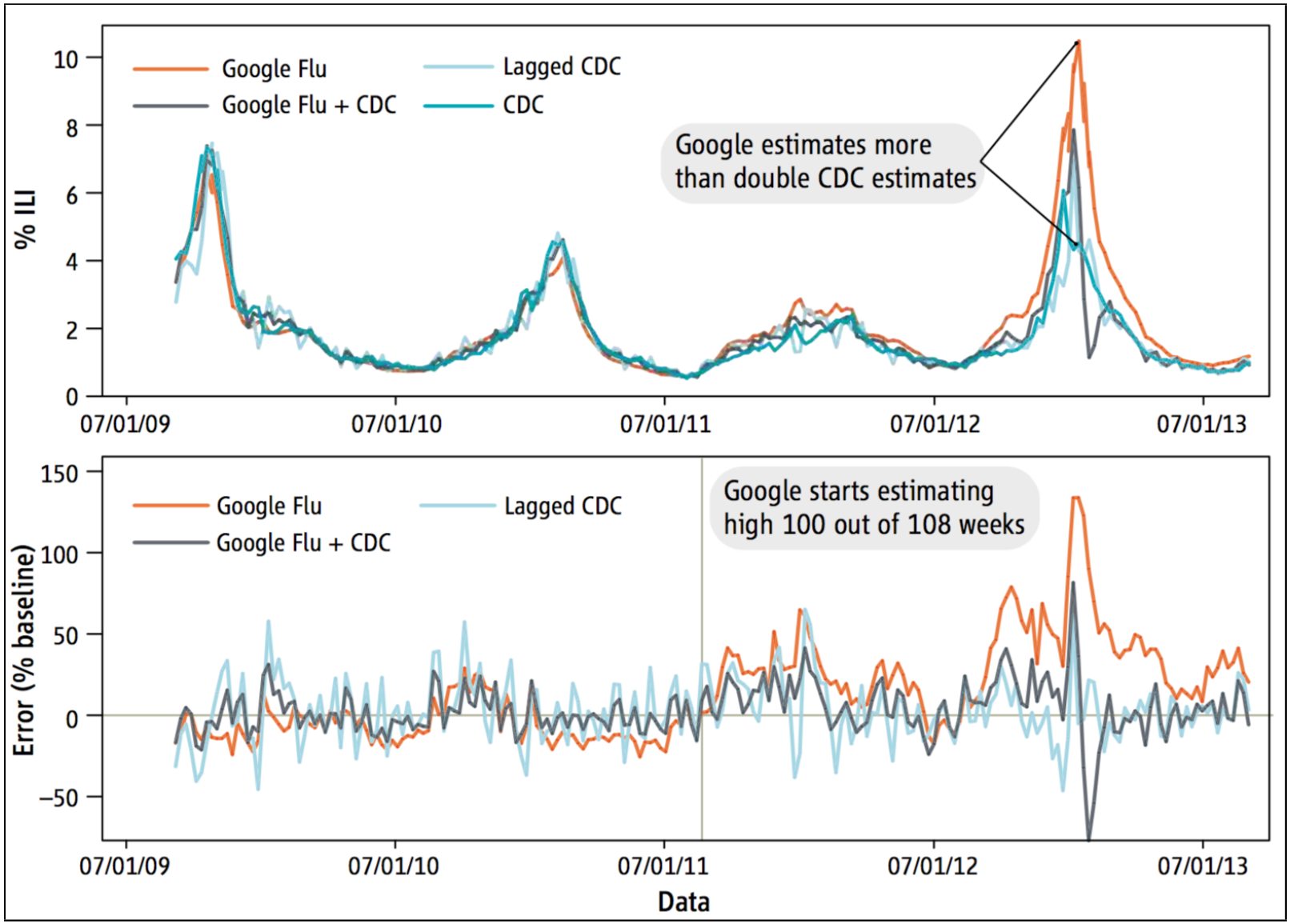
- Merely knowing the existence of a strong association/correlation is not a proof of causation. Investigating association between two variables is an attempt to infer the existence of causation between the two variables.
- For any two correlated events, A and B, the relationship, if any, include:
  - A causes B
  - B causes A
  - A causes B, and B causes A
  - A and B are caused by a common C, but do not cause each other
  - There is no relationship between A and B, the correlation is spurious

Q: How can establish causation?

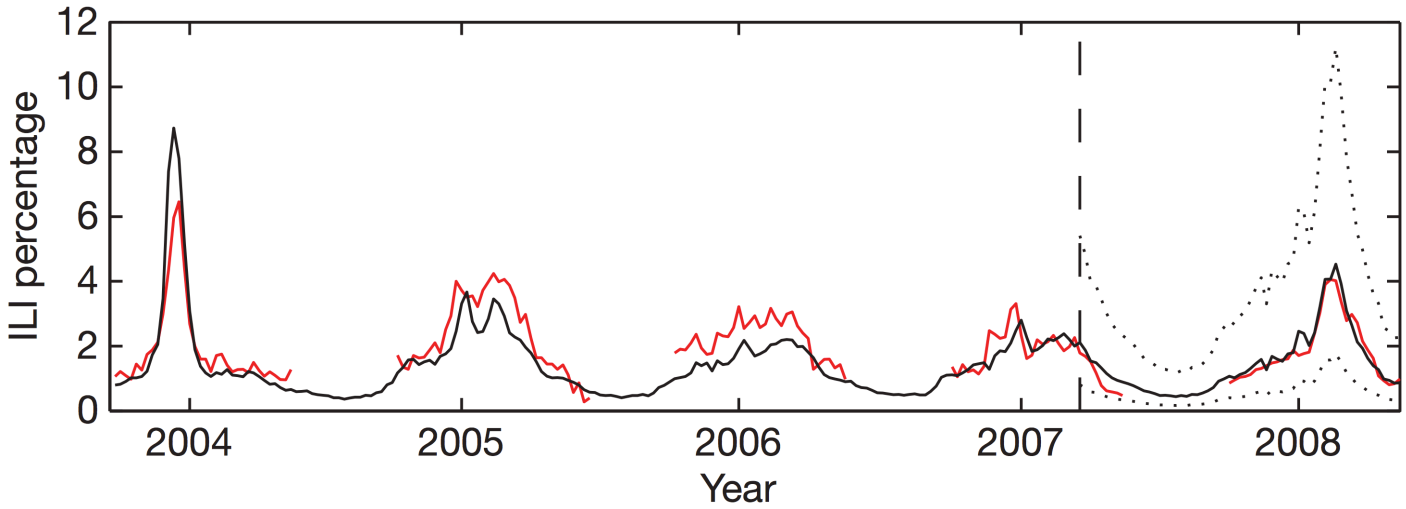
- Causation requires not just a correlation, but a counterfactual dependence.
- A major goal of scientific studies and statistical methods is to approximate the counterfactual state of the world, the quality of the conclusion is largely decided by the quality of this approximation.

Q: Now are you surprised that GFT cannot sustain its performance?

- How bad did Google Flu Trends become?

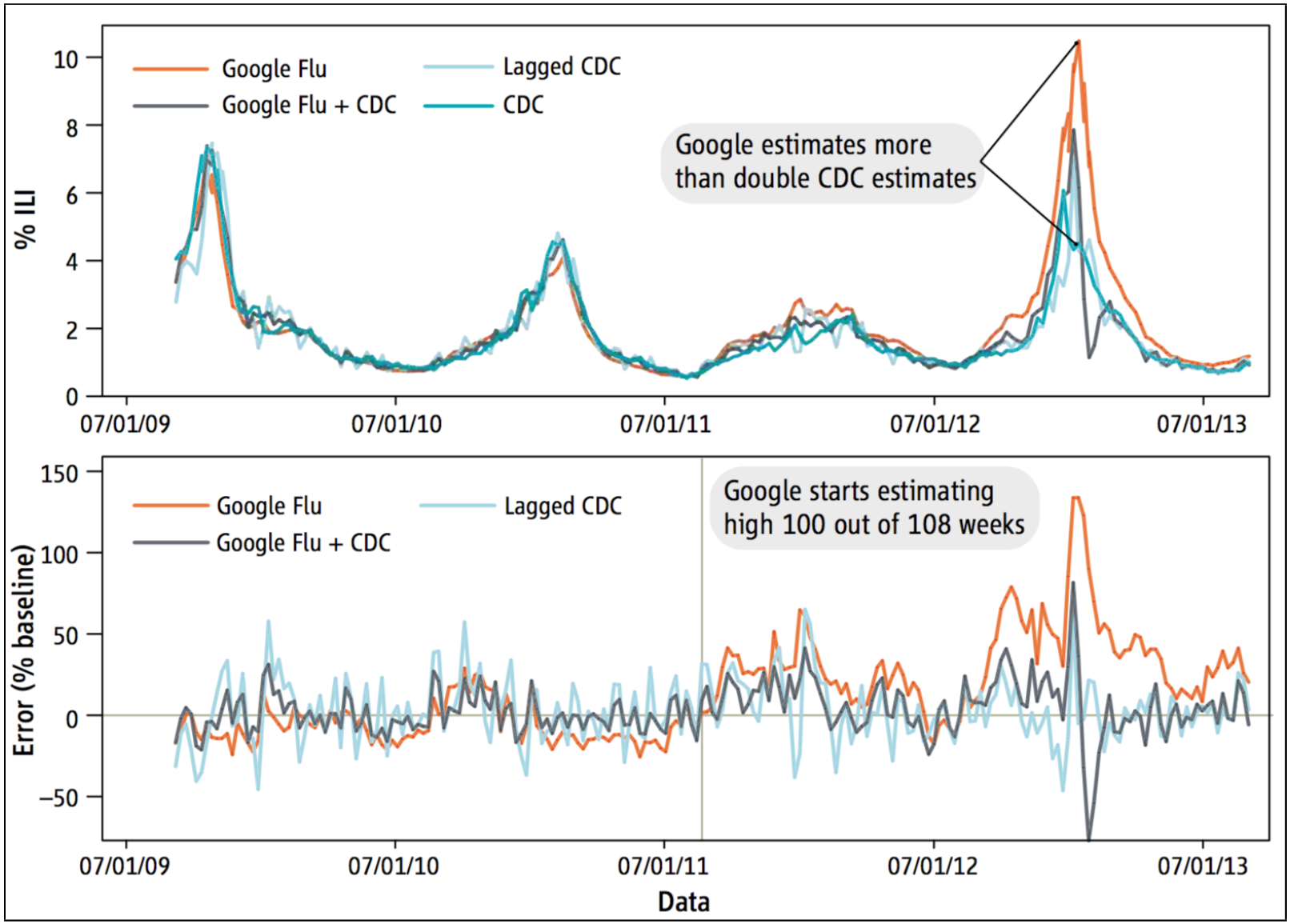


- In defence of Google Flu Trends
  - It was hubris, but the media, with their layman's understanding of data analysis, is partially responsible for the excessive enthusiasm initially.



  - Models are always vulnerable to structure changes, and extreme events.
  - It was so blindly implemented.
  - It costs so little by comparison.
  - It could still be valuable.

- When combining with CDC's model, the performance improves.



- Things that google was criticised by the media
  - Inaccurate by comparison
  - Privacy
- Things that google was criticised by the scientific community
  - Remarkably opaque in terms of method and data
- The lesson from the epic failure of Google Flu Trends
  - Correlation does not imply causation
  - Be aware of changes when building and maintaining a model
  - The importance of knowing "why" as well as "what" and "how"

Q: What language was Google, BackRub, written in at the beginning?

Data Science Wars: Python Vs R

Q: So Python needs no introduction to this audience, however, what is R?

Vs

- There are two unusual things regarding its origin.
- Firstly, it also comes from a quiet and small place,

  University of Auckland, New Zealand

- Secondly, it was created by two statisticians instead of a computer scientist

  Ross Ihaka and Robert Gentleman

- Release Year

  1991

- Inspiration

  C

- Purpose

  Emphasises productivity and code readability.

- Release Year

  1995

- Inspiration

  S

- Purpose

  Focuses on better, user friendly data analysis, statistics and graphical models.

- Usability

  Coding and debugging is easier to do. Any piece of functionality is always written the same way.

- Ease of Learning

  Python's focus on readability and simplicity makes that its learning curve is relatively low and gradual.

- Usability

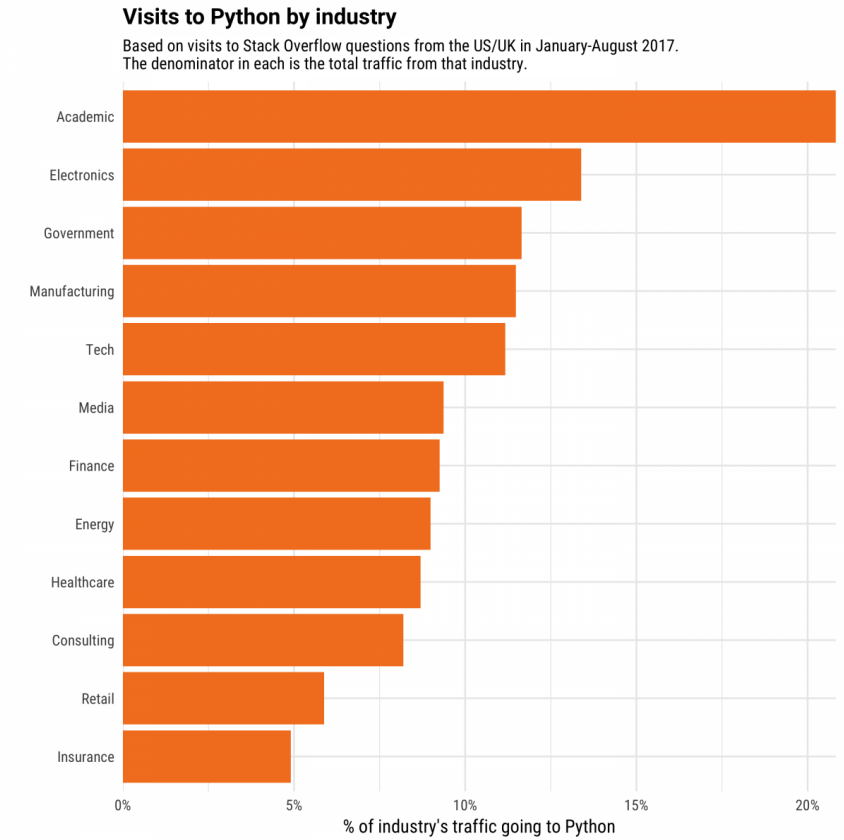
  Statistical models can be written with only a few lines. The same piece of functionality can be written in several ways.

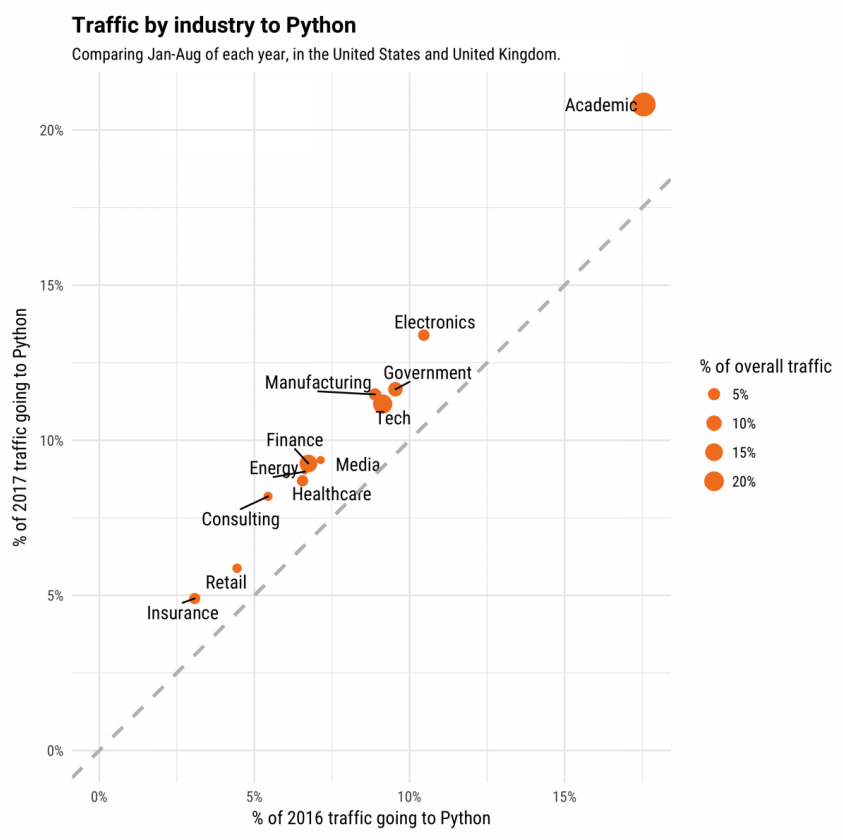
- Ease of Learning

  Require only a minimum knowledge of computing at start, then a steep learning curve later on. However, R is easy to learn for experience programmers in C.

- Community

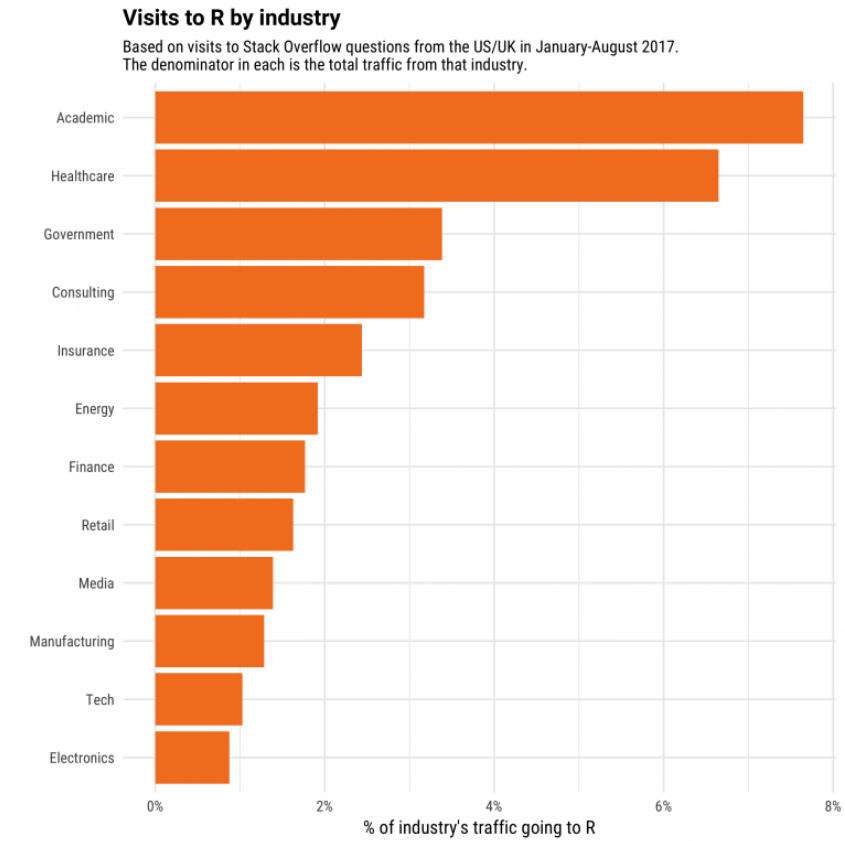
  Python is used by programmers that want to delve into data analysis or apply statistical techniques, and by developers that turn to data scientists.

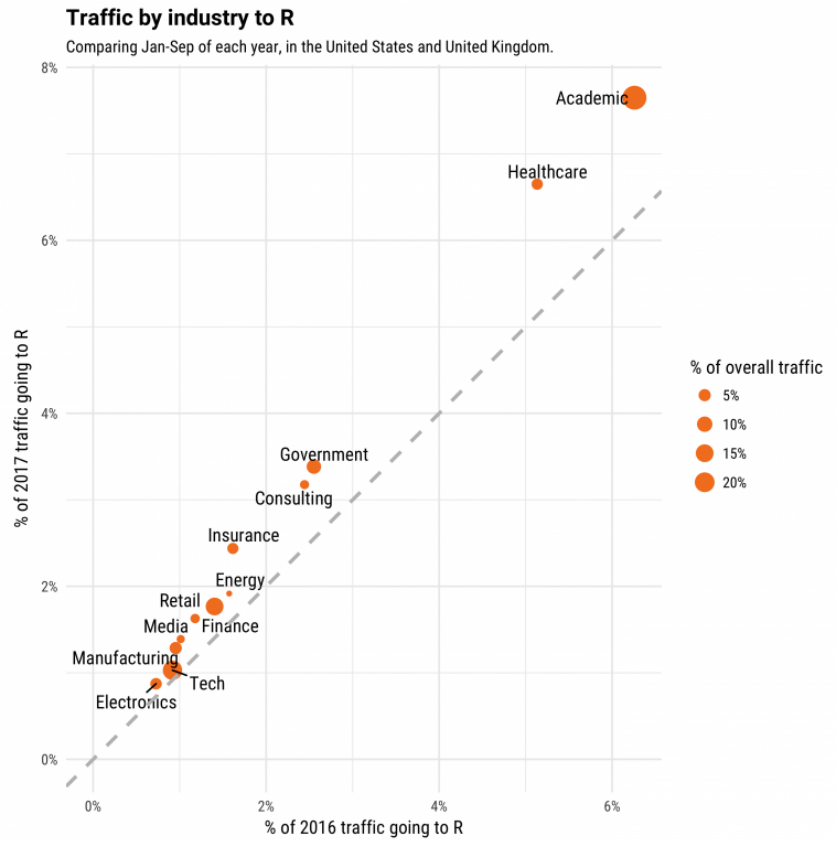




- Community

  Traditionally, R had been used primarily in academic and research institutes. However, R has expanded into the enterprise market.





- Usage

  Python is generally used when the data analysis tasks need to be integrated with web apps or incorporated into a production database.

- Good for

  Implementing algorithms for production use.
  Easy to share your work with other developers.

- Usage

  R is mainly used when the data analysis tasks require standalone computing

- Good for

  Beginners in statistics.
  Easy to interact with.

- So if you are good at programming and you want to learn statistics,

  R

  is the way to go, especially, if you begin with something standard.
- However, if you are a member of a research and development team,

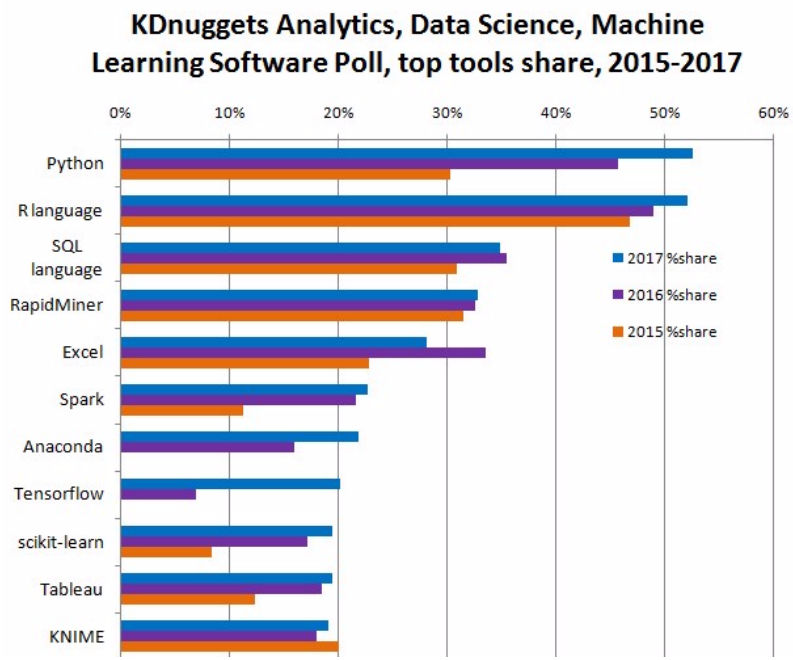
  Python

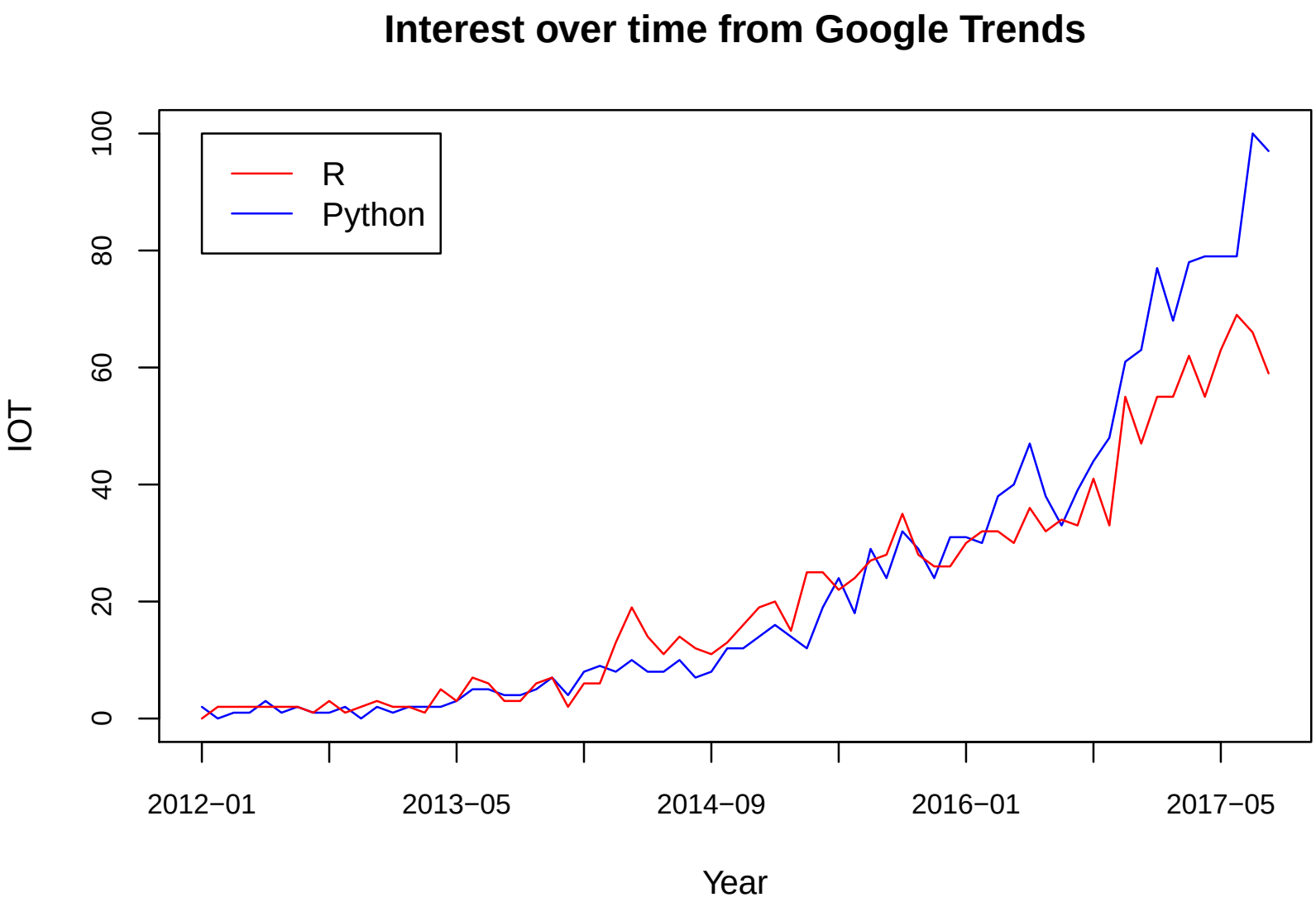
  is the way to go when integrating with web apps is important.

Q: Who is winning at the moment?

2017 IEEE top programming languages

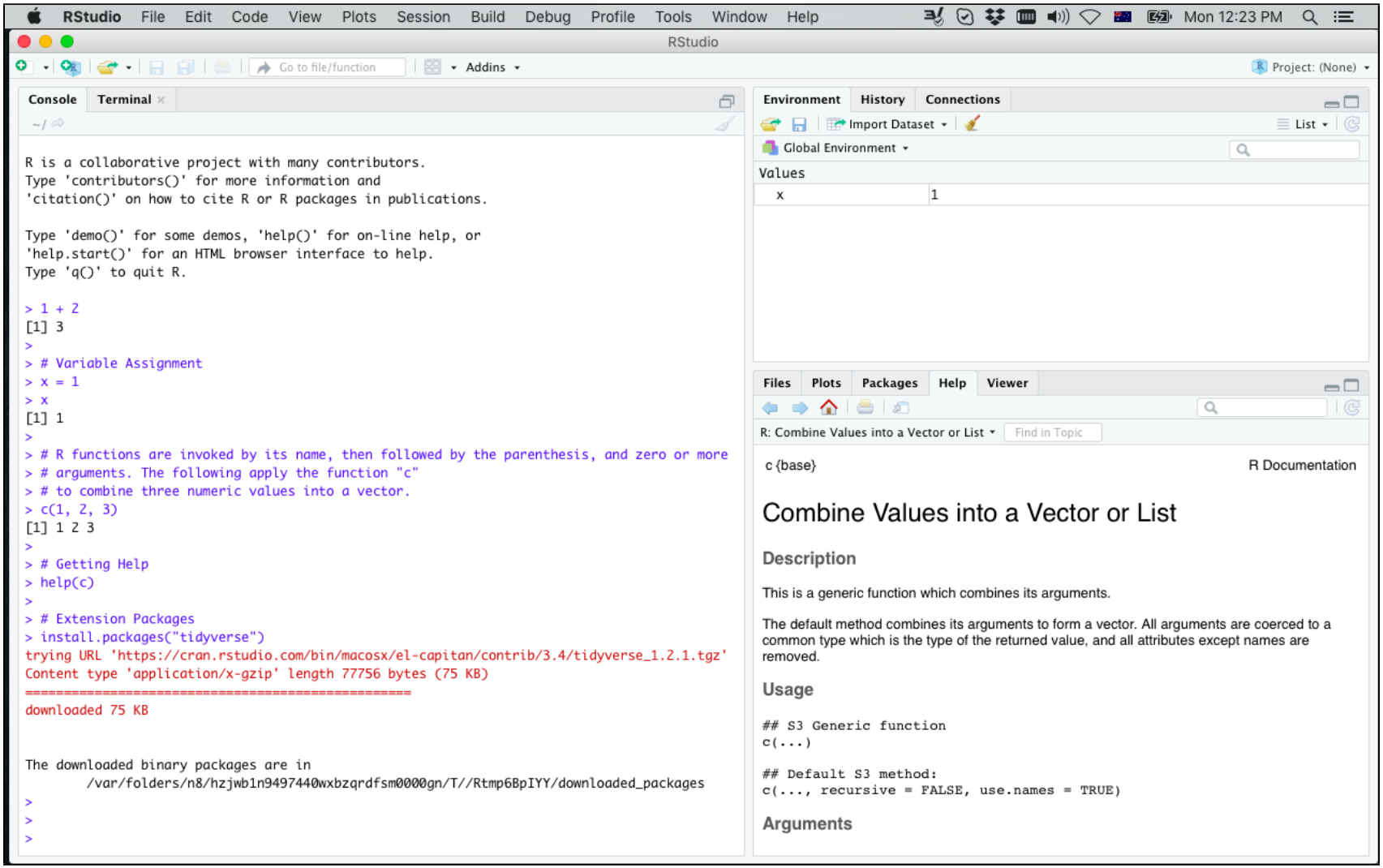
1.	Python	100.0
2.	C	99.7
3.	Java	99.4
4.	C++	97.2
5.	C#	88.6
6.	R	88.1
7.	JavaScript	85.5
8.	PHP	81.4
9.	Go	76.1
10.	Swift	75.3





- More Information on Google Trends

- R can be downloaded from HERE
- RStudio is an IDE for R, and can be downloaded from HERE



Data Type

- In contrast to C, R sets the type based on the given value or expression.

```
> 3.14      # numeric, double-precision real
```

```
[1] 3.14
```

```
> TRUE      # logical, TRUE/FALSE
```

```
[1] TRUE
```

```
> "Hello" # Character
```

```
[1] "Hello"
```

```
> is.logical(TRUE); is.logical("TRUE")
```

```
[1] TRUE
[1] FALSE
```

- Other data type exists, but often not explicitly specified or used

```
> 7L                                                    # integer
```

```
[1] 7
```

```
> is.integer(7L); is.integer(7)
```

```
[1] TRUE
[1] FALSE
```

```
> 2+3i                                                  # complex
```

```
[1] 2+3i
```

```
> sqrt(-1); sqrt(as.complex(-1))                        # R in R
```

```
[1] NaN
# Warning message:
# In sqrt(-1) : NaNs produced
[1] 0+1i
```

Data Structure

- Assignment

```
> x = 1

> x <- 1

> 1 -> x
```

note all of above statements store the value 1 under the name `x`.

```
> y = TRUE
> z = "TRUE"

> class(x); class(y); class(z)     # Object Classes
```

```
[1] "numeric"
[1] "logical"
[1] "character"
```

- Vector

```
> c(1, 2, 3)
```

```
[1] 1 2 3
```

```
> x.vec = c(.Last.value, x)      # Special Variable
> x.vec
```

```
[1] 1 2 3 1
```

```
> 1:12                                  # Sequence
```

```
[1]  1  2  3  4  5  6  7  8  9 10 11 12
```

Q: What do you think the result of the following statement is?

```
> v.vec = 5:-5
```

```
> v.vec
```

```
 [1]  5  4  3  2  1  0 -1 -2 -3 -4 -5
```

- A more general sequence can be generated

```
> y.vec = seq(
+   from = 1,                    # starting
+   by = 1,                      # increment
+   length.out = 16)             # number of elements
```

Q: What do you think the result of the following statement is?

```
> tmp = seq(as.Date('2018-05-15'),
+           by = 7, length.out = 12)

> tmp
```

```
 [1] "2018-05-15" "2018-05-22" "2018-05-29"
 [4] "2018-06-05" "2018-06-12" "2018-06-19"
 [7] "2018-06-26" "2018-07-03" "2018-07-10"
[10] "2018-07-17" "2018-07-24" "2018-07-31"
```

```
> rm(tmp)                      # Remove
```

Watch out for the recycling rule in R

Q: What do you think the result of the following statement is?

```
> c(1, 2, 3, 4) + c(1, 2)
```

Recycling Rule

Vectors occurring in the same expression need not all be of the same length. If they are not, the value of the expression is a vector with the same length as the longest vector which occurs in the expression. Shorter vectors in the expression are recycled as often as need be (perhaps fractionally) until they match the length of the longest vector. In particular a constant is simply repeated.

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}1\\2\end{bmatrix} \longrightarrow \begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}1\\2\\1\\2\end{bmatrix} = \begin{bmatrix}2\\4\\4\\6\end{bmatrix}$$

- Categorical Variable

```
> z.vec = c("hard",
+           "really-hard",
+           "extremely-hard",
+           "bite-your-head-off-hard",
+           "extremely-hard", "really-hard", "hard")

> z.fac = factor(z.vec, order = TRUE,
+                levels = c(
+                  "hard",
+                  "really-hard",
+                  "extremely-hard",
+                  "bite-your-head-off-hard"))

> class(z.fac); mode(z.fac)      # The storage mode
```

```
[1] "ordered" "factor"
[1] "numeric"
```

- By running the factor statement, R has assigned integers to each level

```
> str(z.fac)                    # Structure
```

```
 Ord.factor w/ 4 levels "hard"<"really-hard"<..:
 1 2 3 4 3 2 1
```

```
> z.fac[1]>=z.fac[2]      # Comparison of 1st & 2nd
```

```
[1] FALSE
```

```
> table(z.vec)
```

```
z.vec
bite-your-head-off-hard
                      1
        extremely-hard
                      2
                   hard
                      2
             really-hard
                      2
```

```
> table(z.fac)
```

```
z.fac
                   hard
                      2
             really-hard
                      2
        extremely-hard
                      2
bite-your-head-off-hard
                      1
```

Q: What happens when you mix types inside a vector?

```
> x = 1; y = TRUE; z = "TRUE";
> u.vec = c(x,y); class(u.vec)
> u.vec = c(x,z); class(u.vec)
> u.vec = c(x,z.fac); class(u.vec)
> u.vec = c(y,z.fac); class(u.vec)
> u.vec = c(y,z.fac); class(u.vec)
```

- R does so-called implicit coercion to mixed types, the coercion rule goes

$$\text{logical} -> \text{integer} -> \text{numeric} -> \text{complex} -> \text{character}$$

```
> u.vec = c(x,y); class(u.vec)
```

```
[1] "numeric"
```

```
> u.vec = c(x,z); class(u.vec)
```

```
[1] "character"
```

$$\text{logical} -> \underbrace{\text{factor}}_{\text{integer}} -> \text{numeric} -> \text{complex} -> \text{character}$$

- R effectively assigns an integer to each level of a factor,

```
> u.vec = c(x,z.fac); class(u.vec)
```

```
[1] "numeric"
```

```
> u.vec = c(y,z.fac); class(u.vec)
```

```
[1] "integer"
```

```
> u.vec = c(z,z.fac); class(u.vec)
```

```
[1] "character"
```

- Matrix

```
> A = matrix(
+         y.vec,                    # the data elements
+         nrow = 4,                 # number of rows
+         ncol = 4,                 # number of columns
+         byrow = TRUE)             # fill matrix by rows
```

- Matrices are special vectors in R.

```
> class(A)
```

```
[1] "matrix"
```

```
> mode(A)
```

```
[1] "numeric"
```

```
> attributes(A)
```

```
$dim
[1] 4 4
```

A matrix is stored as a vector with dimensions added on to it.

- Indexing

```
> A
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
[4,]   13   14   15   16
```

```
> A[2,4]
```

```
[1] 8
```

```
> A[3,]
```

```
[1]  9 10 11 12
```

```
> A[c(1,3),c(2,4)]
```

```
     [,1] [,2]
[1,]    2    4
[2,]   10   12
```

- List

```
> xyzlist = list(x.vec=x.vec, y=y, z.fac=z.fac)
> xyzlist
```

```
$x.vec
[1] 1 2 3 1

$y
[1] TRUE

$z.fac
[1] hard
[2] really-hard
[3] extremely-hard
[4] bite-your-head-off-hard
[5] extremely-hard
[6] really-hard
[7] hard
4 Levels: hard < ... < bite-your-head-off-hard
```

- List is a special vector, each element of which can be a different class.

```
> class(xyzlist)
```

```
[1] "list"
```

```
> attributes(xyzlist)
```

```
$names
[1] "x.vec" "y"     "z.fac"
```

```
> class(xyzlist$x.vec); class(xyzlist$y)
```

```
[1] "numeric"
[1] "logical"
```

```
> xyzlist$x.vec[1]; xyzlist[[1]][1]    # 1st of 1st
```

```
[1] 1
[1] 1
```

Q: What do you think the result of the following statement is?

```
> xyzlist[["y"]]; xyzlist[[y]]
```