Week 2 – **Causality, Expressions, Arrays**

Slides by Suraj Rampure

Causality

What is a **treatment**?

A **treatment** is the factor of interest. For example, if wanted to study the effects of coffee on lung cancer, coffee would be the treatment.

What is an **outcome**?

An **outcome** is what the effect you believe your treatment causes. Following the same example, lung cancer would be the outcome.

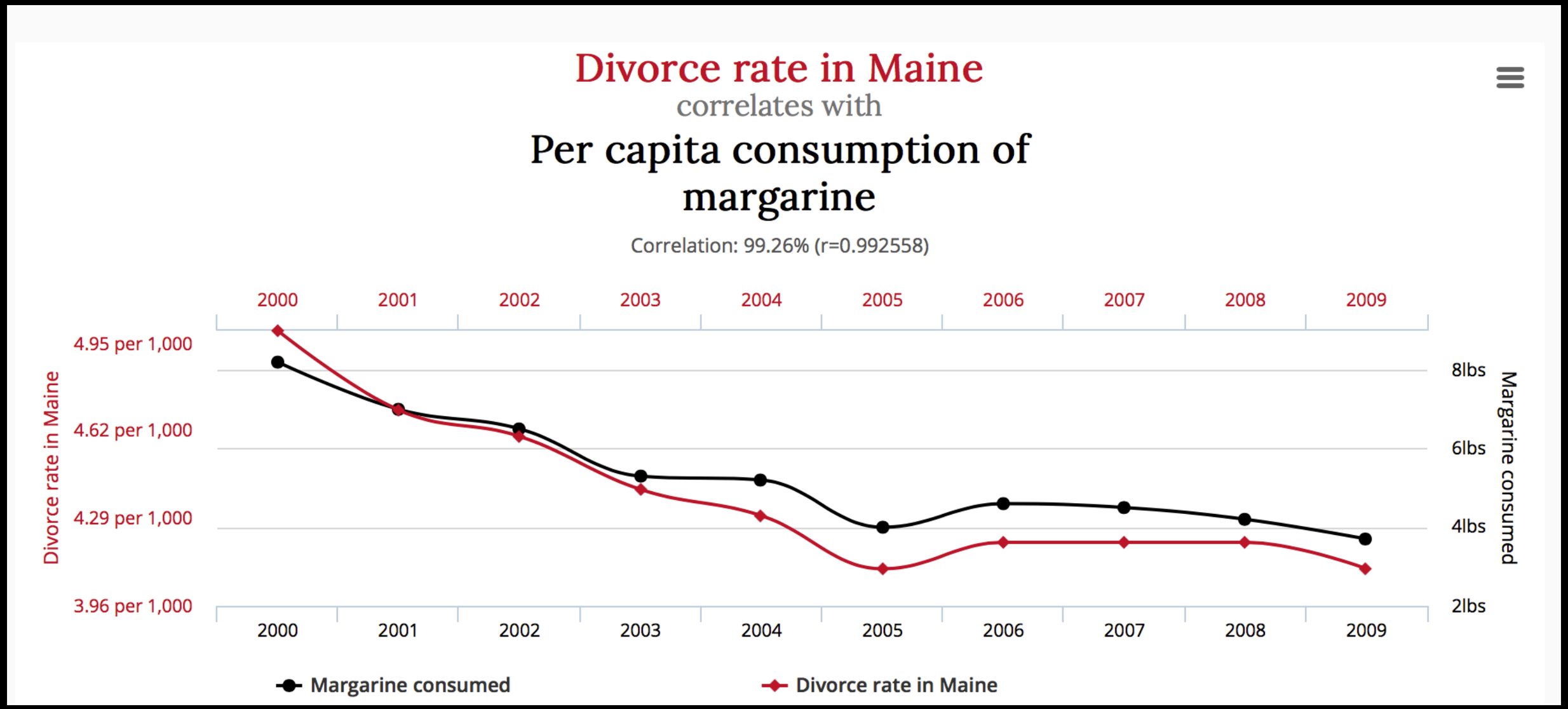
Any relationship between a **treatment** and **outcome** is called an **association**.

Causality

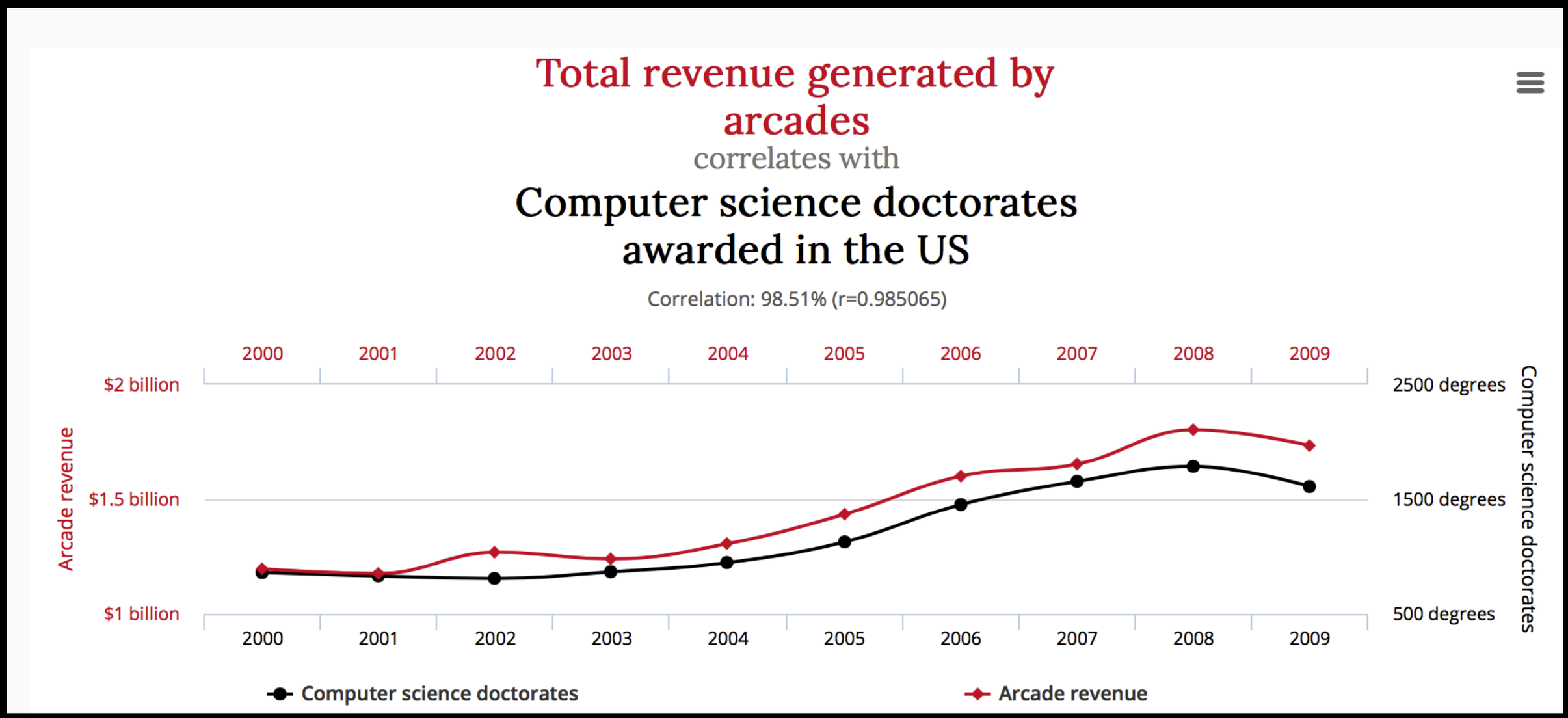
What does **causation** mean?

If and only if the treatment causes the outcome to occur, we say the association is **causal** (not casual). While there used to be an association (or correlation) between drinking coffee and lung cancer, this association wasn't causal.

Why does this matter so much? Let's look at some examples.



Source: http://www.tylervigen.com/spurious-correlations



Clearly, no one of these causes the other – these relationships are not causal.

Source: http://www.tylervigen.com/spurious-correlations

Causality

Why aren't all associations causations?
What makes causation so special?

Confounding factors are things we neither observed nor accounted for in the relationship between our treatment and control.

It turns out that in earlier times, people who drank a lot of coffee tended to smoke a lot, which we know causes lung cancer. This confounding factor lead to a lot of people assuming that coffee causes the lung cancer, but clearly, that wasn't true.

Causality

How can we **establish causation**?

Without randomization, you can’t prove causality, no matter how much you believe a causal relationship exists.

Randomized control trials (RCTs) exist exactly for this purpose.

- Randomly split population into a treatment and control group (not necessarily of same size), but don’t tell who is who

- Give the treatment to the treatment group, and give a placebo to the control group (this is important!)

If you can’t run an RCT, and are simply working with data that you observed, you are performing an **observational study**.

Causality

Sample Problem

You want to study whether reading news affects whether Data 8 students vote. Students are randomly assigned to respond to either survey (a) or survey (b) below.

a) Please read the New York Times for the next 20 minutes and then answer the following questions.

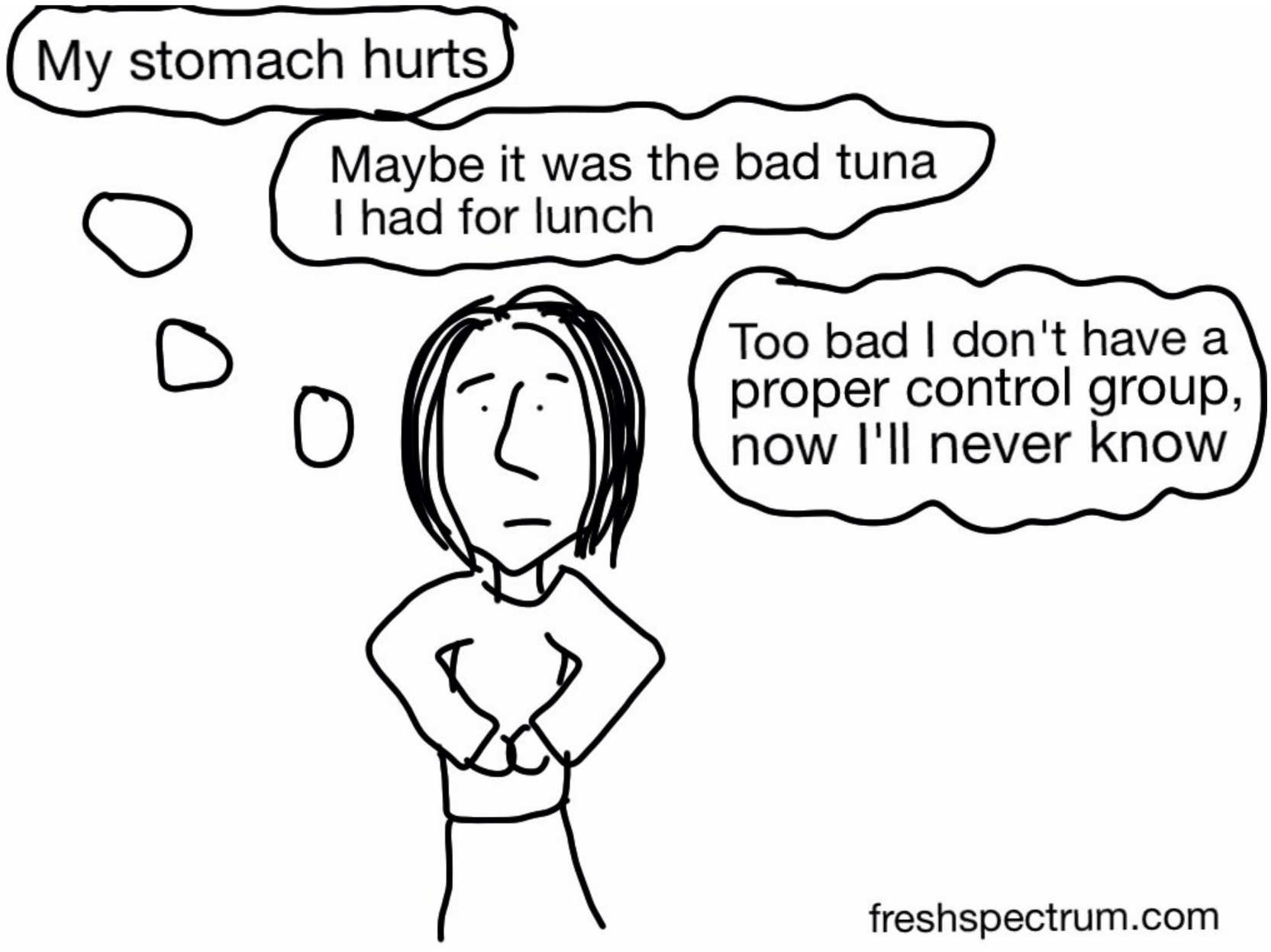
- Do you read a newspaper at least 5 times a week?
- Did you vote in the last presidential election?

b) Please answer the following questions.

- Do you read a newspaper at least 5 times a week?
- Did you vote in the last presidential election?

Can we answer:

1. Is reading the newspaper regularly associated with voting?
2. Is reading the Times for 20 minutes associated with voting?
3. Is reading the Times for 20 minutes associated with reading news regularly?
4. Does reading the newspaper regularly cause people to vote more or less often?
5. What causal relationship can we establish?



Source: https://i2.wp.com/freshspectrum.com/wp-content/uploads/2015/06/No-control-group.jpg?resize=1024%2C768

Read more:

http://www.surajrampure.com/resources/data8/ch2-causation.pdf

Expressions – Lab 2 Tips

Tip 1: Some **function** calls don't change the **objects** they're called on.

input

```
greeting = "hello"
greeting.replace("ello", "owdy")
greeting
```

output

```
'hello'
```

input

```
greeting = "hello"
cowboy_greeting = greeting.replace("ello", "owdy")
cowboy_greeting
```

output

```
'howdy'
```

Even though we called `.replace` on `greeting`, `.replace` doesn't change the value of `greeting`, it just creates a new string. We need to assign some variable, `cowboy_greeting` for example, to this new string if we want to keep its value. Instead of `cowboy_greeting`, we very well could've used the name `greeting` again.

Expressions – Lab 2 Tips

Tip 2: Not all **functions** require an **argument**. Arguments, or parameters, are what we pass into functions – we do this by placing arguments in parentheses ().

input

```
"MAkE THis aLL uppER caSE".upper()
```

output

```
"MAKE THIS ALL UPPER CASE"
```

input

```
import numpy as np
np.random.random()
```

output

```
0.752428603838495
```

`np.random.random()` returns a random real number between 0 and 1

Expressions – Lab 2 Tips

Tip 3: Strings and **numbers** don’t commute!

input

```
23 + “23”
```

output

```
TypeError: unsupported operand type(s) for +: ‘int’
and ‘str’
```

input

```
23 + int(“23”)
```

output

```
46
```

input

```
“45” + “15”
```

output

```
‘4515’
```

Usage of `int(x)` and some other useful tools:

input

```
int(5.491241)
```

output

```
5
```

input

```
int("341")
```

output

```
341
```

input

```
int("365.25")
```

output

```
ValueError
```

`int(x)` converts **x** into an integer, only if it makes sense to

input

```
str(np.pi)
```

output

```
'3.141592653589793'
```

`str(x)` converts **x** into a string

input

```
len("wagner")
```

output

```
6
```

`len(x)` finds the length of **x** (could be a string, or even an array)

input

```
max(5.12, int("13"), -44)
```

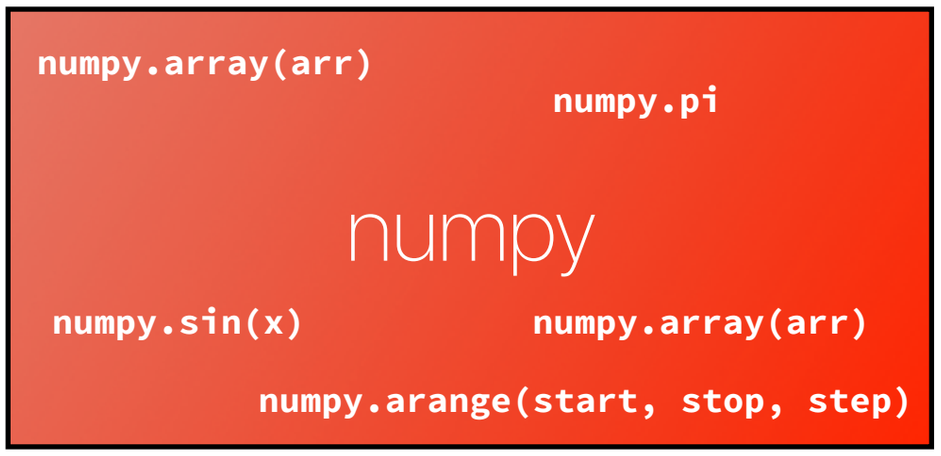
output

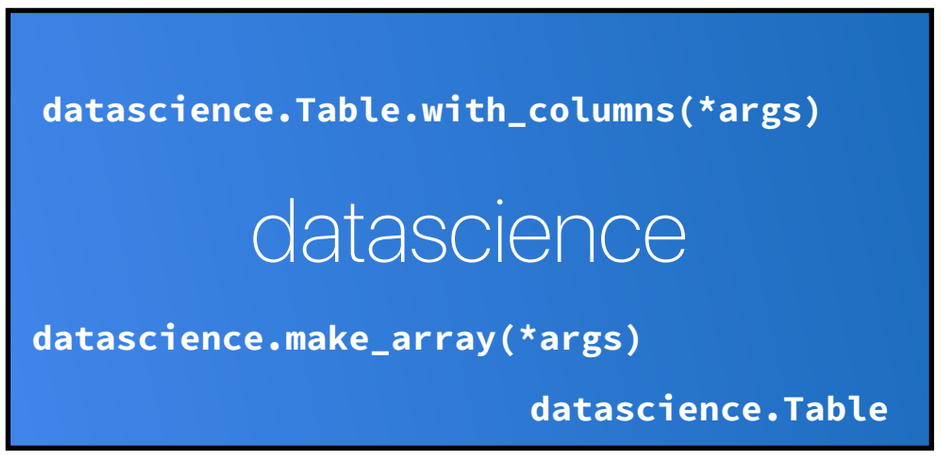
```
13
```

`max(x1, x2, …)` returns the largest of the parameters it is passed

Expressions – Lab 2 Tips

Tip 4: Packages are like toolboxes with pre-defined classes, functions and values.






```
import numpy
new_array = numpy.arange(0, 10, 2)
```

```
import numpy as np
new_array = np.arange(0, 10, 2)
```

```
import datascience
states = datascience.Table.read_csv("usa.csv")
```

```
from datascience import *
states = Table.read_csv("usa.csv")
```

Expressions – Lab 2 Tips

Tip 4: Packages are like toolboxes with pre-defined classes, functions and values.

If you import a package with no other details, you need to mention that package's name every time you use its features.

```
import numpy
new_array = numpy.arange(0, 10, 2)
```

```
import datascience
states = datascience.Table.read_csv("usa.csv")
```

You can also specify which specific tools you want from a package. If you import *, this gives you all tools in a package, and you don't need to mention the package's name.

```
import numpy as np
new_array = np.arange(0, 10, 2)
```

You can also rename a package before importing it. There are good reasons for doing this, instead of importing *.

```
from datascience import *
states = Table.read_csv("usa.csv")
```

Expressions – Lab 2 Tips

Tip 5: Arrays are collections of values of all the same type (all strings, all floats, all tables, etc.)

```
vals1 = make_array(3, 4, -4, -3, 12, 19, 0, -4)
np.sum(vals1) # 27
vals1.item(3) # -3
vals1 + 5 # [8, 9, 1, 2, 17, 24, 5, 1]
np.sin
```

vals1: array(int)

3	4	-4	-3	12	19	0	-4

Remember, in Python (and most programming languages) we start counting at 0, not 1. The first element in an array is at spot 0, the second is at spot 1, etc.

Expressions – Lab 2 Tips

Tip 5: Arrays are collections of values of all the same type (all strings, all floats, all tables, etc.)

```
vals1 = make_array(3, 4, -4, -3, 12, 19, 0, -4])
vals2 = make_array([12, 3, 0, 0, 3, 9, -6, 11])
vals3 = vals1 + vals2
```

vals1: array(int)

3	4	-4	-3	12	19	0	-4

vals2: array(int)

12	3	0	0	3	9	-6	11

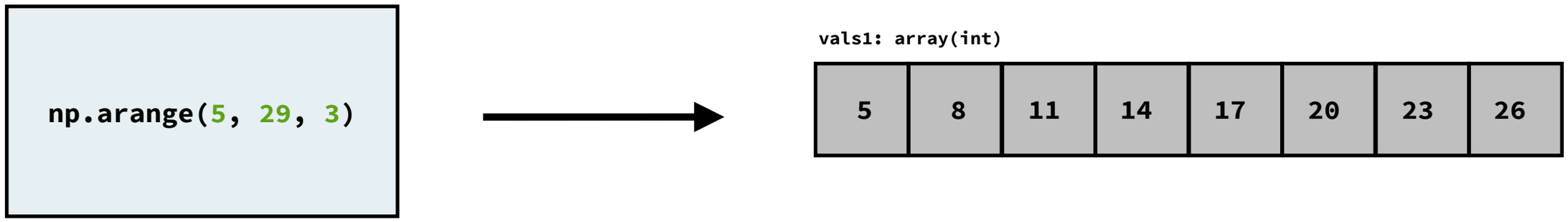
vals3: array(int)

15	7	-4	-3	15	28	-6	7

You can even perform operations between two arrays. In these cases, Python performs the operations element-wise.

Expressions – Lab 2 Tips

Tip 5: Arrays are collections of values of all the same type (all strings, all floats, all tables, etc.)



`np.arange(start, stop, step)` takes three arguments. It creates a new array with every `step`-th value starting from **`start`** and ending before **`end`**. In the above example, passing in the arguments (5, 29, 3) created an array with every 3rd integer starting from 5 and ending before 29. If you don't specify the third element, Python uses the default step value of 1.

Expressions – Lab 2 Tips

Tip 5: Arrays are collections of values of all the same type (all strings, all floats, all tables, etc.)

`movies: Table`

Title	Studio	Gross	Gross (Adjusted)	Year
Star Wars: The Force Awakens	Buena Vista (Disney)	906723418	906723400	2015
Avatar	Fox	760507625	846120800	2009
Titanic	Paramount	658672302	1178627900	1997
Jurassic World	Universal	652270625	687728000	2015
Marvel's The Avengers	Buena Vista (Disney)	623357910	668866600	2012
The Dark Knight	Warner Bros.	534858444	647761600	2008
Star Wars: Episode I - The Phantom Menace	Fox	474544677	785715000	1999
Star Wars	Fox	460998007	1549640500	1977
Avengers: Age of Ultron	Buena Vista (Disney)	459005868	465684200	2015
The Dark Knight Rises	Warner Bros.	448139099	500961700	2012

Each of the columns of a Table is an array as well!

Source: https://www.inferentialthinking.com/chapters/06/1/visualizing-categorical-distributions.html

```
movies.column(0) # [“Star Wars: The Force Awakens”, “Avatar”, ..]
movies.column(“Year”) # [2015, 2009, 1997, ..]
```