CMSC 320
Introduction to Data Science

Akilesh Praveen
Prof. John Dickerson • Fall 2020 • University of Maryland
https://cmsc320.github.io

Last Revision: April 29, 2020

Table of Contents

---

1 Notes & Preface

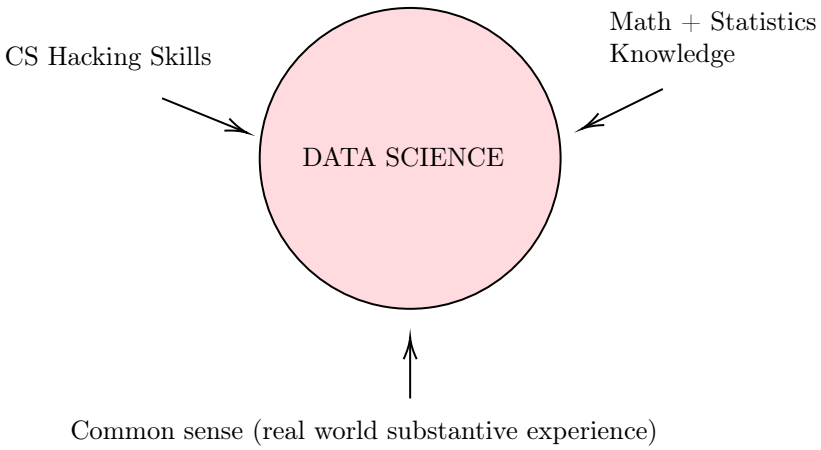
Course notes for CMSC320, under Prof. John Dickerson. Notes collected from previous and current lectures.

2 Lecture 1

What is Data Science?

Data Science is the application of computation and statistical techniques to address or gain insight. It's the intersection of statistics and Computer Science. Based on what I've learned thus far, learning to do data science is like learning how to use a TI-84 in statistics class. You're simply learning how to leverage programming tools in order to perform advanced, complex, and meaningful data-related operations.

It's the use of statistics and computer science in order to find real-world insights.



Topics

Here are the general topics that this class will cover.

- Processing data
- Visualizing data
- Understanding data
- Communicating data
- Extracting value from data

Tools

Here are some tools commonly employed by data scientists. We'll try to cover how to use most of them here.

- Python
- Scikit-Learn
- Docker
- PANDAS
- Spark
- TensorFlow

Conda

Conda is a package and environment manager for python that we can use with the command line. We can create multiple environments for us and install separate packages in each of them. This will be highly useful to us, as we sometimes want to consolidate the tools we use into separate environments.

3 Lecture 2

Definition:

Data Collection → The process of measuring and gathering information on targeted variables.

Literate Programming

The idea of **literate programming** is that you have the source code, an explanation of the source code, and the end result of running the code all in one file. Usually, this file is identified as a *notebook*. In other words, the syntax is no different from regular code, you just get a more organized way to show off tables, plots, and other outputs generated from your code.

Jupyter Notebook + Alternatives

Jupyter Notebook is a service that started off as `iPython`, but it's basically a web-based platform that we use for literate programming. Specifically, it supports Python-based literate programming. Most data scientists prefer it, and it can also apparently leverage big data tools, such as Apache Spark.

It saves files in `.ipynb` format, which most platforms (i.e. GitHub) have built in viewers for. Options to export in other readable formats are available. Basically, it's just Python with a bunch of bells and whistles on top to make the output of your code look pretty.

Apache Zeppelin is an alternative data analysis tool, but we will stick to Jupyter for our purposes. This is because Jupyter seems to be preferred in industry.

RStudio is the equivalent, for people who prefer to use the `R` programming language for data science.

This course will be centered around Jupyter Notebook.

List Comprehensions in Python

To make lists in Python, you can use loops or the `map()` function, but a *pythonic* way of doing this would be to use a list comprehension. Below is a simple example.

Example: Make a list of all the squares of `{0,1,2,3,4,5,6,7,8,9}`

List Comprehension:

```
squares = [i * i for i in range(10)]
```

A good way of thinking about this is that it allows you to build sets like a mathematician. This is a common theme in data science, where we can find the intersection between a lot of math stuff and computer science stuff. It's good to know how lists are generated in a mathematical sense in Python for that reason. Here's an example where we translate mathematical notation into a Python list comprehension.

Example: Make a list of all odd natural numbers from 0 to 999

Math Notation:
$E = \{x \mid x \text{ in } \mathbb{N} \land x \text{ is odd } \land x < 1000 \}$

List Comprehension:
```
E = [x for x in range(1000) if x % 2 != 0]
```

Using Python3

We will use Python3. Since I used Python2 during my internship, I'm going to note some big changes to keep track of.

- Python3 is backwards incompatible. (Don't write in Python2!)
- Print has changed from a command to a function, so make sure to use proper function notation when invoking it.
- Division has changed. `1/2` no longer equals 0. `1/2 == 0.5` and floored division is now taken care of this way: `1//2 == 0`

Python vs. R for Data Scientists

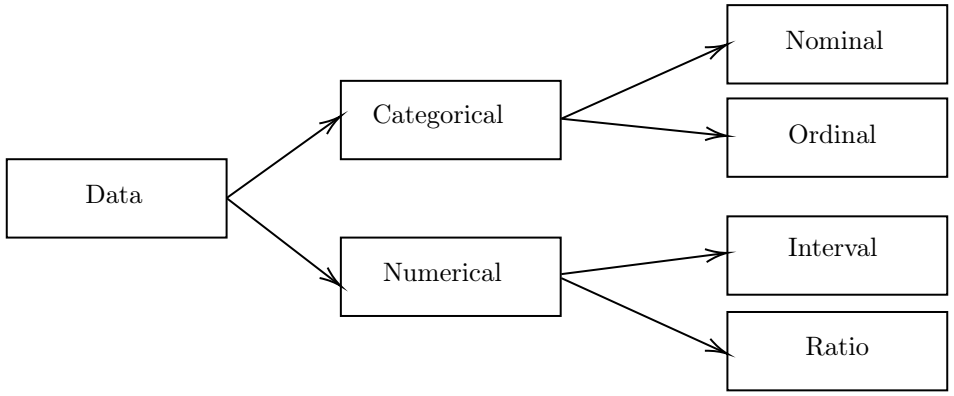
Some arguments for both sides in terms of what to use.

- Python is a 'full' programming langauge. Also, if you've got prior experience with Python paradigms or just programming in general, that's a big plus in terms of learning curve.
- R has more mature 'pure statistics' libraries, but Python is apparently catching up.
- In terms of **processing speed**, R is certainly faster. It was designed and optimized for statistics processing.
- Python is preferable for machine learning operations, which is pretty big right now.

My personal choice will be to use Python as much as I can when I'm studying this course. Since it's more prominent in the tech industry, I should be using it more anyway.

The Classic Statistical View of Data

There are **four** main types of data: Nominal, Ordinal, Interval, and Ratio data. They can each be classified under two main subgroups, Categorical and Numerical data. Here's a visualization.



Nominal Data

A type of categorical data, nominal data value have names and describe the state of things. For example, your marriage status is nominal data because you can either be *single*, *married*, or *separated*. Another example is the type of drink you're going to have. Will it be *Milk*, *Beer*, or *Juice*?

The key here is that there can be no quantitative values assigned to each of these categories, as that would allow us to do math with them and would defeat the purpose of these labels. These values **cannot be easily compared**, so they have no material value. *E.g. being single is not quantitatively better than being married (objectively), and vice versa.*

Example: What is your marital status?

- Married
- Divorced (separated)
- Single

Ordinal Data

Ordinal data represents values that have names that describe the state of things, but in this case, there **is** an ordering of those values. This is what sets it apart from nominal data.

Example: What did you think of the movie?

- Strongly liked
- Liked
- Indifferent
- Disliked
- Strongly Disliked

Given how subjective some of these things can be, the distinction between nominal and ordinal can be **blurry** at times. For example, going back to our nominal example, some people may think that being single is quantitatively better than being married.

Interval and Ratio Data

Interval and Ratio data are pretty similar, and both can be used to measure things that can be represented by either integers or real numbers.

Interval data scales with fixed but arbitrary values. That might sound silly, but a good example is **dates**. Below is an example of two data comparisons of interval data that seem arbitrary, but indeed hold integer value.

Example: The following two operations are equal.

```
10/1/2019 - 9/1/2019
10/1/2018 - 9/1/2018
```

The measures don't look like integer values at first, but we can quantify them by marking them with days.

Here's what sets **Interval** data apart, however. You have **no method** of computing ratios or scales with it. For example, never mind that you can try computing `(9/1/2019 × 8/25/2015)`, the unit of the answer would be totally

useless to us, and neither would the actual number, even if you went ahead with the operation.

Ratio data is, in essence, the same as interval data in that it is numerical, but the scale itself **has a true zero**. While dates don't necessarily have a true zero, we can say that money counts as ratio data. For example, having zero money means that you're at the absolute zero of that scale, whereas the absolute zero for dates is disputable. Are we saying we're starting at O A.D.? The Big Bang? Even earlier?

Differentiating between the two is usually a case-by-case basis thing, which is what I'm thinking is the best way to handle any conflicts I end up running into between ratio and interval data.

Example: Interval data

Temperature on the scale of Celsius or Fahrenheit is interval-type data because $0°$ is set to an arbitrarily fixed point. Also, we can't scale it properly- $30°F$ isn't twice as hot as $15°F$.

Example: Ratio data

Temperature on the Kelvin scale is ratio data. $0K$ is set at legitimate absolute zero, and $50K$ is truly twice as cold as $100K$.

Data Science at a Glance

4 Test Section

Lower Level

Here's a cool code example.

```c
#include <stdio.h>
#include <math.h>

int main() {
   double value;

   printf("Enter a number: ");
   scanf("%lf", &value);     /* Notice the use of %lf */

   printf("sqrt %f: \n", sqrt(value));
   printf("power of 2: %f\n", pow(value, 2));
   printf("sin: %f\n", sin(value));

   return 0;
}
```

Here's some more information about our code example.

5 Footnotes

Taken by Akilesh Praveen.