EC2401: INTRODUCTION TO DATA SCIENCE

LECTURE 0

Evan Piermont

Front Matter

This course:

- is an introduction to data science
- is an introduction to the Python programming language
- is a lab class / requires lots of practice
    - 1 hour of lecture per week
    - 2 hours of seminar / lab
- Office hours by appointment or see me after class

Assessment:

- 4 problem sets (ungraded)
- Midterm exam (**40%**)
  - Take home exam; programing exercises
- Final project (**60%**)
  - Data based project; can work in pairs

There will be ”coding interviews” throughout the year.

What is data science?

Data $\longrightarrow$ Information $\longrightarrow$ Insights

What is data science?

Data $\longrightarrow$ Information $\longrightarrow$ Insights

- Data Acquisition, Cleaning, Statistical Analysis

What is data science?

Data ⟶ Information ⟶ Insights

- Data Acquisition, Cleaning, Statistical Analysis
- Interpretation, Visualization, Providing context

What is data science?

How is this different from statistics / econometrics?

What is data science?

How is this different from statistics / econometrics?
Data science is statistics **+ implementation**:

What is data science?

How is this different from statistics / econometrics?
Data science is statistics + implementation:

- How to write code to gather data, clean it, analyze it
- How data is structured (in real life)
- Efficiency and elegance of our programs

Okay, but what is data?

Data is a set of states that represent basic units of meaning:

- Boolean — True/False; On/Off
- Discrete — Quantity; Type
- Continuous — Time; Intensity; Length
- Words (called Strings) — Description

We can create more complex data-types from simpler ones:

- A **list** of numbers
- A **dictionary** of key/value pairs
- A **graph** of connections between nodes
- A **table** (or matrix) of rows and columns

How does a computer store data?

But, since we are using computers—our data must be encodable by a computer Computers store data in bits, switches that are either on or off:

- `0` $\rightarrow$ Off ; `1` $\rightarrow$ On
- How to store Boolean data is obvious: store `true` as `1` and `false` as `0`.
- What about integers, strings, lists?

Binary integer encoding

$0 :$ 0000

$1 :$

$2 :$

$3 :$

$4 :$

$5 :$

$6 :$

$7 :$

$8 :$

Binary integer encoding

$0:$ `0000`

$1:$ `0001`

$2:$

$3:$

$4:$

$5:$

$6:$

$7:$

$8:$

Binary integer encoding

$0:$ `0000`

$1:$ `0001`

$2:$ `0010`

$3:$

$4:$

$5:$

$6:$

$7:$

$8:$

Binary integer encoding

$0 :$ 0000

$1 :$ 0001

$2 :$ 0010

$3 :$ 0011

$4 :$

$5 :$

$6 :$

$7 :$

$8 :$

Binary integer encoding

$0 :$ 0000
$1 :$ 0001
$2 :$ 0010
$3 :$ 0011
$4 :$ 0100
$5 :$
$6 :$
$7 :$
$8 :$

Binary integer encoding

$0 :$ 0000

$1 :$ 0001

$2 :$ 0010

$3 :$ 0011

$4 :$ 0100

$5 :$ 0101

$6 :$

$7 :$

$8 :$

Binary integer encoding

$0:$ `0000`
$1:$ `0001`
$2:$ `0010`
$3:$ `0011`
$4:$ `0100`
$5:$ `0101`
$6:$ `0110`
$7:$ `0111`
$8:$ `1000`

Binary integer encoding

$$
\begin{array}{ll}
0 : 0000 & (0 \times 8) + (0 \times 4) + (0 \times 2) + (0 \times 1) \\
1 : 0001 & (0 \times 8) + (0 \times 4) + (0 \times 2) + (1 \times 1) \\
2 : 0010 & (0 \times 8) + (0 \times 4) + (1 \times 2) + (0 \times 1) \\
3 : 0011 & (0 \times 8) + (0 \times 4) + (1 \times 2) + (1 \times 1) \\
4 : 0100 & (0 \times 8) + (1 \times 4) + (0 \times 2) + (0 \times 1) \\
5 : 0101 & (0 \times 8) + (1 \times 4) + (0 \times 2) + (1 \times 1) \\
6 : 0110 & (0 \times 8) + (1 \times 4) + (1 \times 2) + (0 \times 1) \\
7 : 0111 & (0 \times 8) + (1 \times 4) + (1 \times 2) + (1 \times 1) \\
8 : 1000 & (1 \times 8) + (0 \times 4) + (0 \times 2) + (0 \times 1)
\end{array}
$$

Binary word encoding

We simply map each character to an integer. The most common encoding is ASCII. For example:

- ‘ ’ → `0100000`
- ‘A’ → `1000001`
- ‘B’ → `1000010`
- ‘C’ → `1000011`

We need to be able to manipulate data:

- Combine Booleans (`and`, `or`, `not`, etc.)
- Arithmetic on numbers (addition, multiplication, etc.)
- Manipulation of strings (concatenation, replacement, etc.)

This is very tedious at the level of bits.

Programming Languages

A programming language is a set of rules that turn (human readable) instructions into machine level manipulation of data.

- A program is a set of instructions that turns an input (data) into an output (data)
- Programs are reusable, scalable, and, usually, fast (compared to human calculations)
- We care about:
  - Abstraction
  - Modularity

Python

We will use Python for our implementation.

- Python is a general purpose programing language
- It is free and open source
- We will explore three different ways of interacting with Python:

local

terminal /
command prompt

`.py`

anaconda

`.ipynb`

cloud

google colab

Boolean Logic

We will consider three connectives:

$\texttt{AND}$: p $\texttt{AND}$ q is true if p and q are both true, and false otherwise.

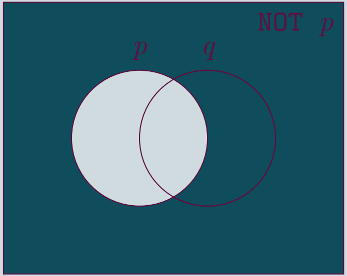
$\texttt{OR}$: p $\texttt{OR}$ q is false if p and q are both false, and true otherwise.

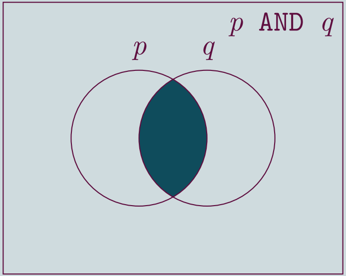
$\texttt{NOT}$: $\texttt{AND}$ p is true if p is false, and false p is true.

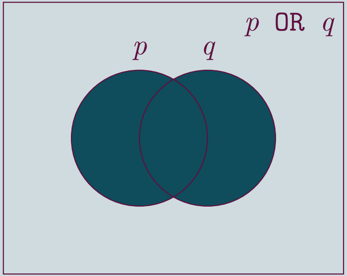
We can capture these relations via truth tables:

p	q	$\texttt{NOT}\ p$	$\texttt{NOT}\ q$	$p\ \texttt{AND}\ q$	$p\ \texttt{OR}\ q$
T	T	F	F	T	T
T	F	F	T	F	T
F	T	T	F	F	T
F	F	T	T	F	F

We could also view this graphically:







Use one of these methods to show that:

$$p \ \texttt{AND} \ q \quad = \quad \texttt{NOT}\big((\texttt{NOT} \ p) \ \texttt{OR} \ (\texttt{NOT} \ q)\big)$$

Booleans as Numbers

Recall that Python will convert `True` to `1` and `False` to `0`.

Then what are our connectives as operations on $\{0, 1\}$?

- $\texttt{NOT}\ \ p = 1 - p$
- $p\ \ \texttt{AND}\ \ q = \min\{p, q\}$
- $p\ \ \texttt{OR}\ \ q = \max\{p, q\}$

The same truth tables verify this:

p	q	$\texttt{NOT}\ p$	$\texttt{NOT}\ q$	$p\ \texttt{AND}\ q$	$p\ \texttt{OR}\ q$
T	T	F	F	T	T
T	F	F	T	F	T
F	T	T	F	F	T
F	F	T	T	F	F

The same truth tables verify this:

p	q	$1 - p$	$1 - q$	$\min\{p, q\}$	$\max\{p, q\}$
1	1	0	0	1	1
1	0	0	1	0	1
0	1	1	0	0	1
0	0	1	1	0	0