Math for Data Science

Number Theory and Data Science

Joanna Bieri DATA100

Important Information

- Email: joanna_bieri@redlands.edu
- Office Hours take place in Duke 209 unless otherwise noted – Office Hours Schedule

Today's Goals:

- Define different types of numbers,variables and data
- How does basic math and number types apply to Data Science?

Important Types of Numbers

Natural Numbers 1,2,3....

Whole Numbers 0,1,2,3...

Integers ...-3,-2,-1,0,1,2,3...

Important Types of Numbers

Rational Numbers Numbers that can be expressed fractions. Remember that $3 = \frac{3}{1}$ so this contains all the integers.

Irrational Numbers Decimal Numbers that cannot be expressed as fractions. They have an infinite number of decimal places. For example the number π.

Real Numbers All rational and irrational numbers

Complex and Imaginary Numbers Numbers that contain the square root of negative 1. We express this as $i = \sqrt{-1}$. These happen mathematically a lot!

Important types of Python Variables

int these are integers - no decimal

float these are rational or irrational numbers (within rounding). A computer can't keep an infinite number of decimal places so there is rounding going on. Typically python will save 15-17 decimal digits of precision.

string these are words - anything with quotes around it.

Important types of Data Science Variables

Numerical - this is data represented by numbers

Categorical - this is data represented by strings

Python Code - Explore Data Types

```
import numpy as np
import sympy as sp

a = 1
b = 0.5
c = 'hello'
d = np.pi

print(type(a))
print(type(b))
print(type(c))
print(type(d))
```

Why does the number type (Number Theory) even matter?

Data is Messy!

When you are working with data a big part of your job will be cleaning data and deciding what types of numbers/variables should be allowed in the data. Here is an example of a messy data set about movies. It has 9999 observations (movies) listed and lots of information about those movies.

Data is Messy!

```
import pandas as pd

file_location = 'https://joannabieri.com/mathdatascience/
                 data/movies.csv'
DF = pd.read_csv(file_location)

DF.head(10)
```

Data is Messy!

	MOVIES	YEAR	GENRE
0	Blood Red Sky	(2021)	\nAction, Horror, Th
1	Masters of the Universe: Revelation	(2021–)	\nAnimation, Action
2	The Walking Dead	(2010–2022)	\nDrama, Horror, T
3	Rick and Morty	(2013–)	\nAnimation, Adven
4	Army of Thieves	(2021)	\nAction, Crime, Ho
5	Outer Banks	(2020–)	\nAction, Crime, Dr
6	The Last Letter from Your Lover	(2021)	\nDrama, Romance
7	Dexter	(2006–2013)	\nCrime, Drama, My
8	Never Have I Ever	(2020–)	\nComedy
9	Virgin River	(2019–)	\nDrama, Romance

Data is Messy!

Look at lust one row:

```
MOVIES                                                  Blood Red Sky
YEAR                                                           (2021)
GENRE                    \nAction, Horror, Thriller
RATING                                                            6.1
ONE-LINE     \nA woman with a mysterious illness is forced ...
STARS        \n    Director:\nPeter Thorwarth\n| \n    Star...
VOTES                                                        21,062
RunTime                                                       121.0
Gross                                                           NaN
Name: 0, dtype: object
```

Data is Messy!

What kind of numbers do we expect? What kind of variables are we getting? What does NaN mean?

- MOVIES – string or words
- YEAR – should be a number or integer
- GENRE – string or words
- RATING – should be a number or rational/float
- ONE-LINE – string or words
- STARS – string or words
- VOTES – should be a number or integer
- RunTime – should be a number - maybe integer maybe float?
- Gross – NaN means not a number - either infinity or no data was given

Data is Messy!

Some of the data is not the right format!!!!

```
display(DF['YEAR'].iloc[0])
```

```
'(2021)'
```

```
display(type(DF['YEAR'].iloc[0]))
```

```
str
```

The data in the year column is a string not an integer. Can we just turn them all into integers? Not really! Some of the data represents a range of years!

Always keep track of what type of data you might be interacting with!

Why does the number type (Number Theory) even matter?

Feature Engineering

In data science we often do something called *feature engineering* this means taking given data and creating new data with it. Sometimes it is as simple as multiplying or adding to pieces of data. Sometimes it is much more complicated.

Feature Engineering

Here is an example data set where all of the variables have been encoded in some way. It describes the number of bikes that were rented in Washington DC. It gives information about the weather, and day of the week. For example

- season : season (1:spring, 2:summer, 3:fall, 4:winter)
- yr : year (0: 2011, 1:2012)
- mnth : month (1 to 12)
- hr : hour (0 to 23)

Feature Engineering

More complicated is how they "normalized" the temperatures.

- temp : Normalized temperature in Celsius. The values are divided to 41 (max)
- atemp: Normalized feeling temperature in Celsius. The values are divided to 50 (max)
- hum: Normalized humidity. The values are divided to 100 (max)
- windspeed: Normalized wind speed. The values are divided to 67 (max)

Feature Engineering

```
file_location = 'https://joannabieri.com/mathdatascience/
                data/bikeshare-day.csv'
DF = pd.read_csv(file_location)

DF.head(10)
```

Feature Engineering

	instant	dteday	season	yr	mnth	holiday	weekday	workingd
0	1	2011-01-01	1	0	1	0	6	0
1	2	2011-01-02	1	0	1	0	0	0
2	3	2011-01-03	1	0	1	0	1	1
3	4	2011-01-04	1	0	1	0	2	1
4	5	2011-01-05	1	0	1	0	3	1
5	6	2011-01-06	1	0	1	0	4	1
6	7	2011-01-07	1	0	1	0	5	1
7	8	2011-01-08	1	0	1	0	6	0
8	9	2011-01-09	1	0	1	0	0	0
9	10	2011-01-10	1	0	1	0	1	1

Feature Engineering

As a data scientist you might want to undo these calculations. For example all the temperatures have been divide by 41 which was the maximum temperature. So we would need to do

$$Temp = 41 * atemp$$

to get temperatures in Celsius.

Feature Engineering

What if we wanted to convert to Farenheit?

$$F = \frac{9C}{5} + 32$$

so

$$Temp = \frac{9}{5}(41 * atemp) + 32$$

You have to be confident in putting together calculations like this in Python. Some important questions to keep in mind:

- What kind of numbers do I expect to get out of the calculation?
- What would a big or small value be for our calculation?

Feature Engineering

Here I will do this calculation and add a new column (feature or variable) to my data:

```
DF['newtemp']=DF['atemp'].apply(lambda x: 9/5*(41*x)+32)
```

	dteday	atemp	newtemp
0	2011-01-01	0.363625	58.835525
1	2011-01-02	0.353739	58.105938
2	2011-01-03	0.189405	45.978089
3	2011-01-04	0.212122	47.654604
4	2011-01-05	0.229270	48.920126

Does my answer make sense?

Why does foundational math matter?

As we see, computers can do so much math for us these days! But when you are dealing with data sets and trying to come to important or interesting conclusions you need math at your fingertips!

Making Predictions

Often one of the goals of Data Science is to make a prediction about what we can expect in the world around us.

Here is some data about the temperature and then number of cricket chirps per minute. Maybe we are wondering can we predict the temperature just based on counting the chirps?

NOTE: I am also going to introduce you to a new way of plotting!

Making Predictions

```
file_location = 'https://joannabieri.com/mathdatascience/
                 data/Cricket_chirps.csv'
DF = pd.read_csv(file_location)

DF.rename(columns = { 'X':'Temperature',
                      'Y':'Chirps per Minute'},
                      inplace=True)
DF.head(10)
```

Making Predictions

	Temperature	Chirps per Minute
0	88.599998	19
1	71.599998	16
2	93.300003	22
3	84.300003	17
4	80.599998	19
5	75.199997	19
6	69.699997	17
7	82.000000	18
8	69.400002	15
9	83.300003	18

Matplotlib.pyplot

```
import matplotlib.pyplot as plt
```

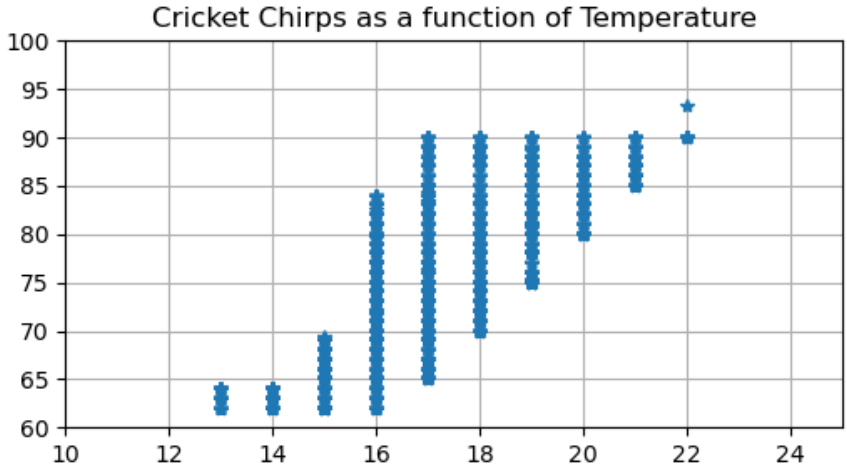
This is a python package for plotting. It can plot more things that sympy can, even though I would still use sympy for plotting basic functions. What sympy cannot do is easily plot data. In this class I will use **matplotlib.pyplot**.

NOTE: There are LOTS of other ways to make plots in Python. In DATA101 we used **plotly.express** and another great package is **seaborn**. You are welcome to use any of these, but my notes and code will use either sympy or matplotlib this semester.

Matplotlib.pyplot - Chirps as a function of Temperature

```
y=np.array(DF['Temperature'])
x=np.array(DF['Chirps per Minute'])

plt.plot(x,y,'*')
plt.grid()
plt.xlim([10,25])
plt.ylim([60,100])
plt.title('Cricket Chirps as a functoin of Temperature')

plt.show()
```

Matplotlib.pyplot - Chirps as a function of Temperature



Matplotlib.pyplot - Chirps as a function of Temperature

There seems to be an increase in the number of chirps as the temperature increases. We can put a straight line through this data using a polynomial fit

```
coefficients = np.polyfit(x, y, 1)
coefficients
```

```
array([ 3.36808636, 17.96720331])
```

Matplotlib.pyplot - Chirps as a function of Temperature

The numbers we see here are the coefficients for the equation of the line. The line that fits our data is

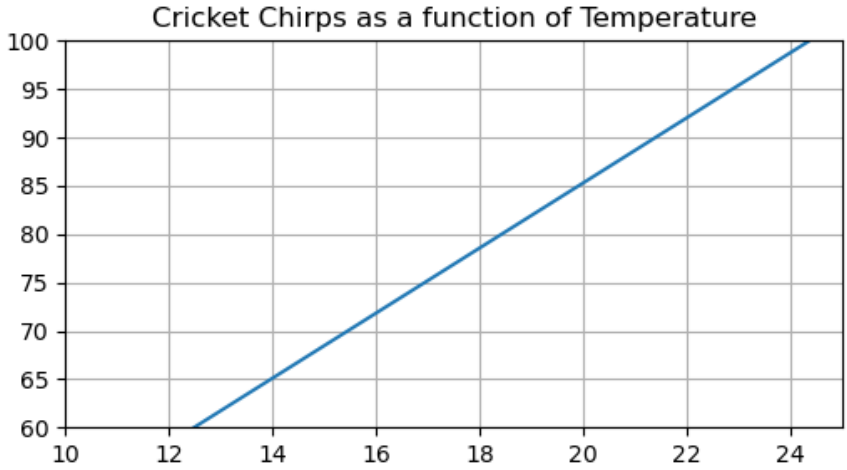
$$T = 3.36808636C + 17.96720331$$

Matplotlib.pyplot - Chirps as a function of Temperature

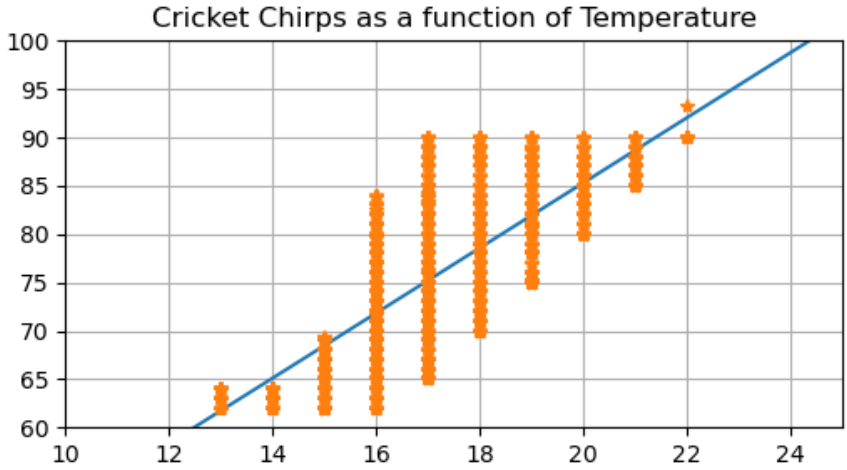
Let's see how we can plot this function in matplotlib.

```
# Get your range of x values
x = np.arange(10,25,.5)
# Create your y values using your function
y = 3.36808636*x + 17.96720331

# The plot stuff pretty much looks the same!
plt.plot(x,y,'-')
plt.grid()
plt.xlim([10,25])
plt.ylim([60,100])
plt.title('Cricket Chirps as a function of Temperature')

plt.show()
```

Matplotlib.pyplot - Chirps as a function of Temperature



Matplotlib.pyplot - Chirps as a function of Temperature



Chirps as a function of Temperature - Predictions

But now I can ask all sorts of questions!!!

1. If I hear 20 chirps per minute what is a good estimate for the temperature?
2. If the temperature is 70 degrees how many chirps should I expect to hear?
3. For what range of numbers does my equation make sense?

Chirps as a function of Temperature - Predictions

1. If I hear 20 chirps per minute what is a good estimate for the temperature?

```
x = sp.symbols('x')
y = 3.36808636*x + 17.96720331
y.subs(x,20)
```

$$85.32893051$$

If there are 20 chirps per minute then it is about 85 degrees outside.

Chirps as a function of Temperature - Predictions

2. If the temperature is 70 degrees how many chirps should I hear?

```
# We need to find the inverse
x,y = sp.symbols('x,y')
my_expr = y - 3.36808636*x - 17.96720331

sp.solve(my_expr,x)
```

```
[0.296904500987914*y - 5.33454353290395]
```

```
y = sp.symbols('y')
x = 0.296904500987914*y - 5.33454353290395
x.subs(y,70)
```

$$15.44877153625$$

If it is 70 degrees outside then I should expect about 15 chirps per minute.

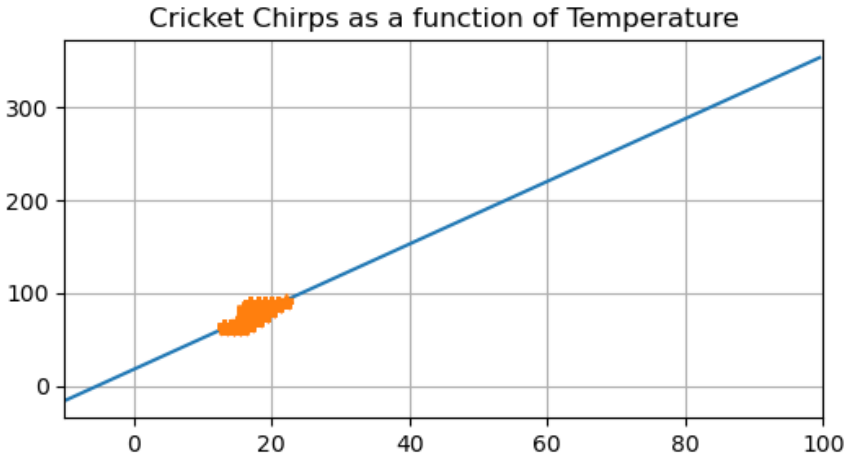
Chirps as a function of Temperature - Predictions

3. For what range of numbers does my equation make sense?

I know that my temperature can't go infinitely high. What would it even mean to have negative chirps?

Technically I could extend my linear fit way beyond where it is reasonable. This is why it is important to know what kind of numbers you might expect. Think about the following plot, what are the limitations on where we could use our prediction?

Chirps as a function of Temperature - Predictions



Moral of the Story

- Understanding number types helps in data cleaning, feature engineering, and predictions.
- The more confident you are in your basic math skills the more you can do with the data!
- We didn't talk specifically about order of operations (PEMDAS) but you have to make sure what you are typing into python makes sense!