# Data Science with Python

**iCB2 – Introduction to Computational Biology and Bioinformatics**
November 10, 2015

Emidio Capriotti
http://biofold.org/

Biomolecules Folding and Disease

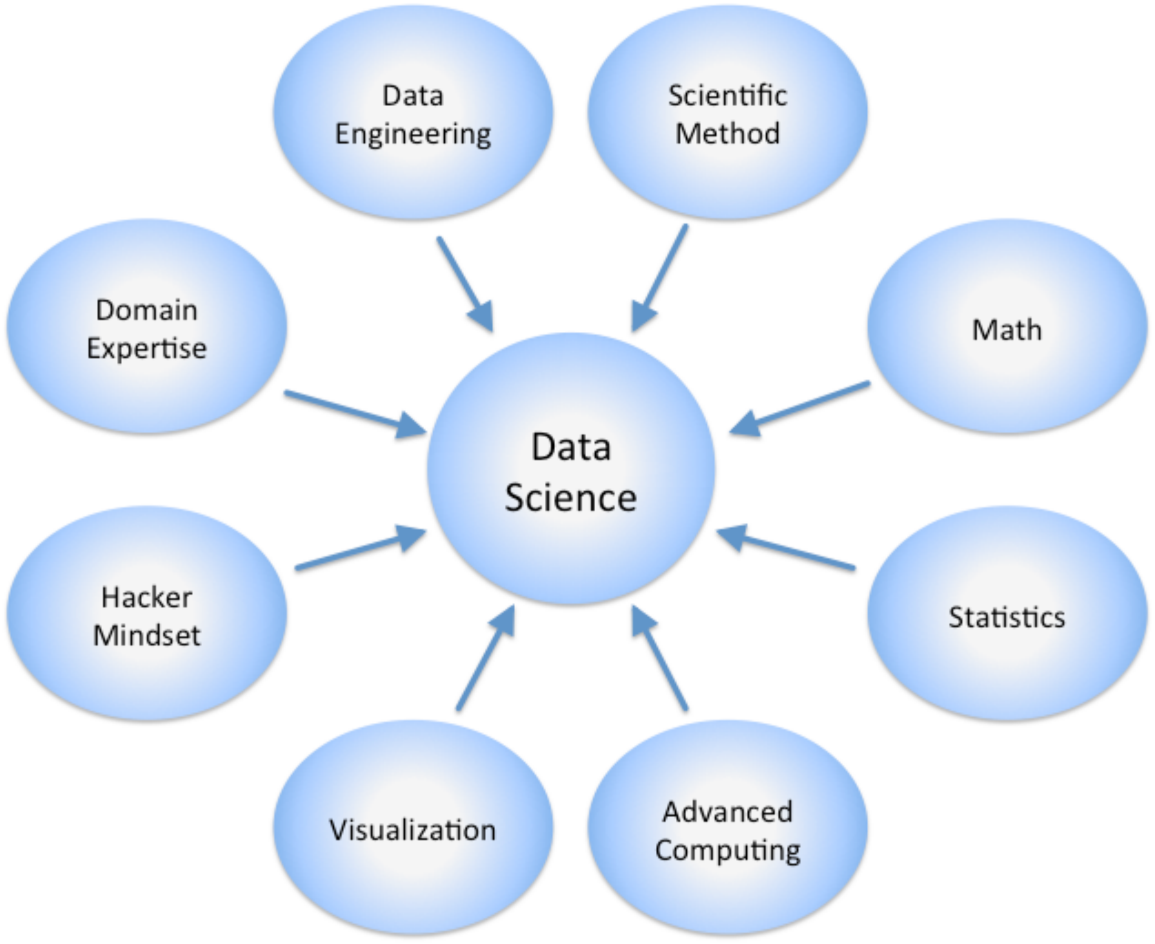
Institute for Mathematical Modeling
of Biological Systems
Department of Biology

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

# What is Data Science?

Data Science is an interdisciplinary field about processes and systems to extract knowledge or insights from large volumes of data in various forms, either structured or unstructured.

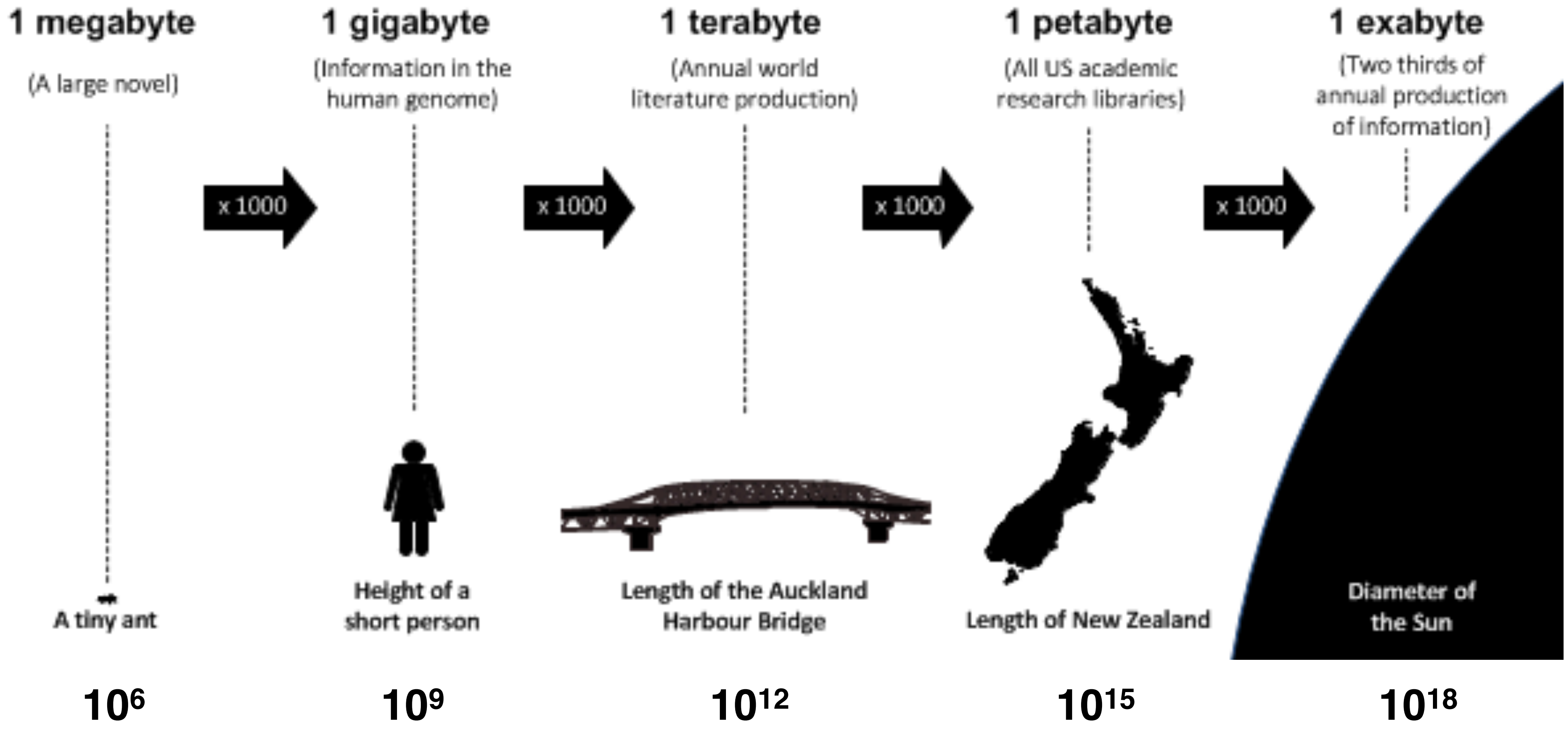
The need for data scientists emerged in response to the “data deluge” – the increasingly large amounts of data generated each year – and the realization that some of this data is uniquely valuable.



http://dreamhost.com

# Data Deluge

Comparing data size with physical objects to visualize the magnitude of data growth.

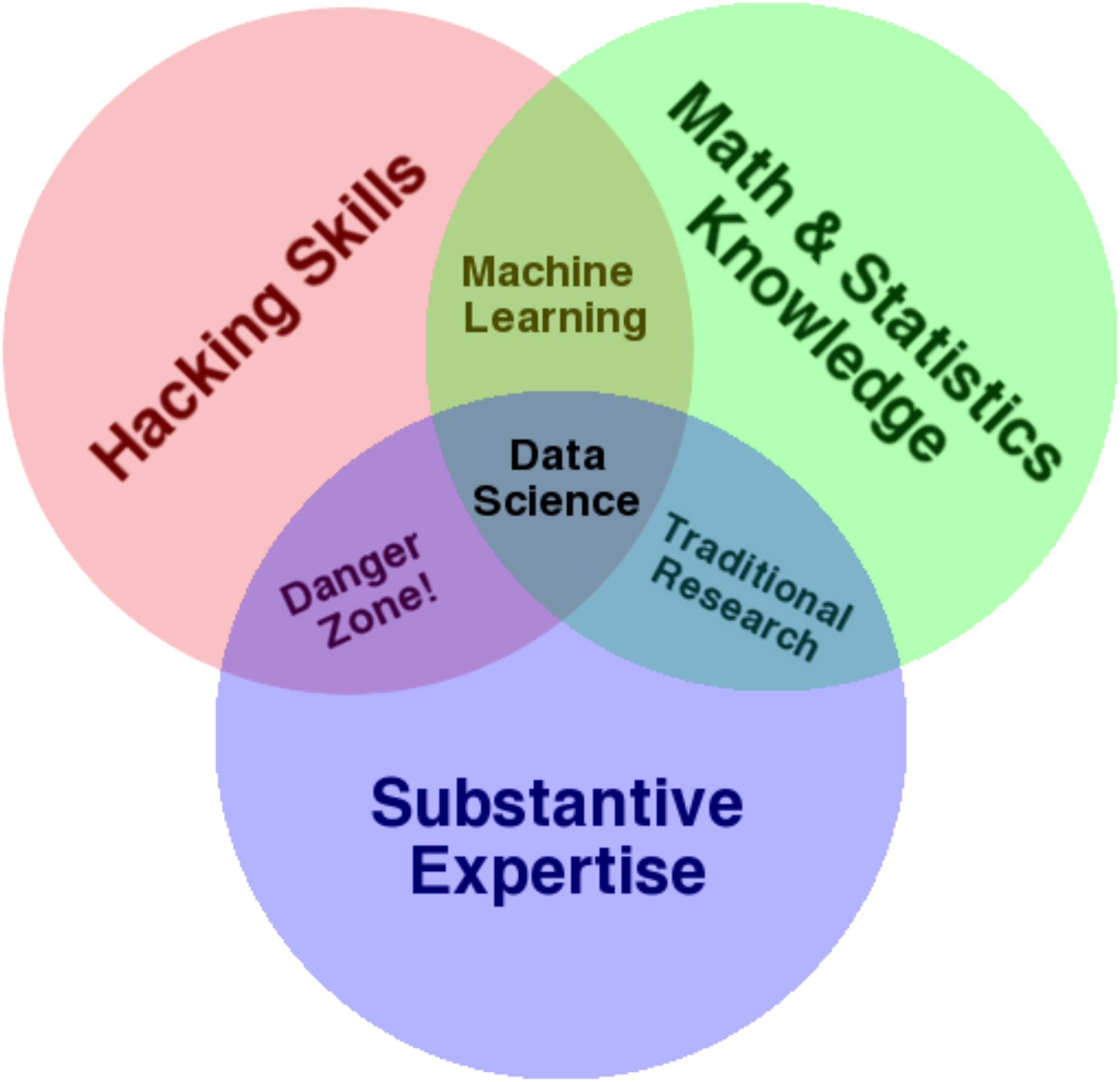


| | |
|---|---|
| Length of a tiny ant | 1.4 mm |
| Height of a short person | 1.4 m |
| Length of the Auckland Harbor Bridge | 1,020 m |
| Length of New Zealand | 1,600 km |
| Diameter of the Sun | 1,390,000 km |

http://seradigm.co.nz

# Data Science Venn Diagram

The primary skills in data science are hacking, math and stats knowledge, and substantive expertise.



http://http://drewconway.com/

# Required Skills

Data scientists use their data and analytical ability to:

- find and interpret rich data sources;
- manage large amounts of data;
- merge data sources;
- ensure consistency of datasets;
- build mathematical models using the data;
- visualize and communicate the data insights/findings.

Data Science can be used in different domains. Some example are:

- Public Health
- Consumer target models

# Basic variable types

The simplest type of variable in programming is the boolean

```
>>> bit=True
>>> type(bit)
<type 'bool'>
```

The simplest numeric variable is integer

```
>>> inum=1
>>> type(inum)
<type 'int'>
```

More complex numeric variable is float

```
>>> fnum=1.5
>>> type(fnum)
<type 'float'>
```

In python character variable does not exist.

```
>>> text=‘Hello World’
>>> type(text)
<type 'str'>
```

# String variables

In low level program languages the string is not a basic variable. It is actually a group of concatenated characters.

In programming languages such as fortran the length of a string is fix. In python a string can assume any length and it do not need to be declared.

Python provides built-in functions for dealing with string

```
>>> text=“Hello world!”
>>> print len(text)
12
>>> print text[0]
H
>>> print text[-1]
!
>>> print text[2:5]
llo
```

# *if* and operators

Basic structure of if in python. Also elif can be used.

```
if    (condition 1):
          do something 1
elif (condition 2):
          do something 2
else:
          do something 3
```

Standard operators are ==, !=, >, >=, <, <= that can be combined with and, or, not

Write a function that check names for length and first characters

```
>>> def check_name(name,name_len,letter):
...             if (len(name>=name_len and name[0]==letter):
...                         return True
...             else:
...                         return False
...
>>> print chech_name(‘Goofy’,5,’G’)
True
```

# *for* and *while* loops

Basic structure of the for and while loop in python.

```
for  i  in  list:
        do something                # Indentation is needed

while (condition):
        do something                # Indentation is needed
```

Build a function that takes a text variable and print all the letters

```
>>> def print_for_letters(text):
…               for i in text:
…                           print i

>>> def print_while_letters(text):
…               i=0
…               while i<len(text):
…                     print text[i]
…                     i+=1
```

# List

One way to represent a group of data in python is the list.

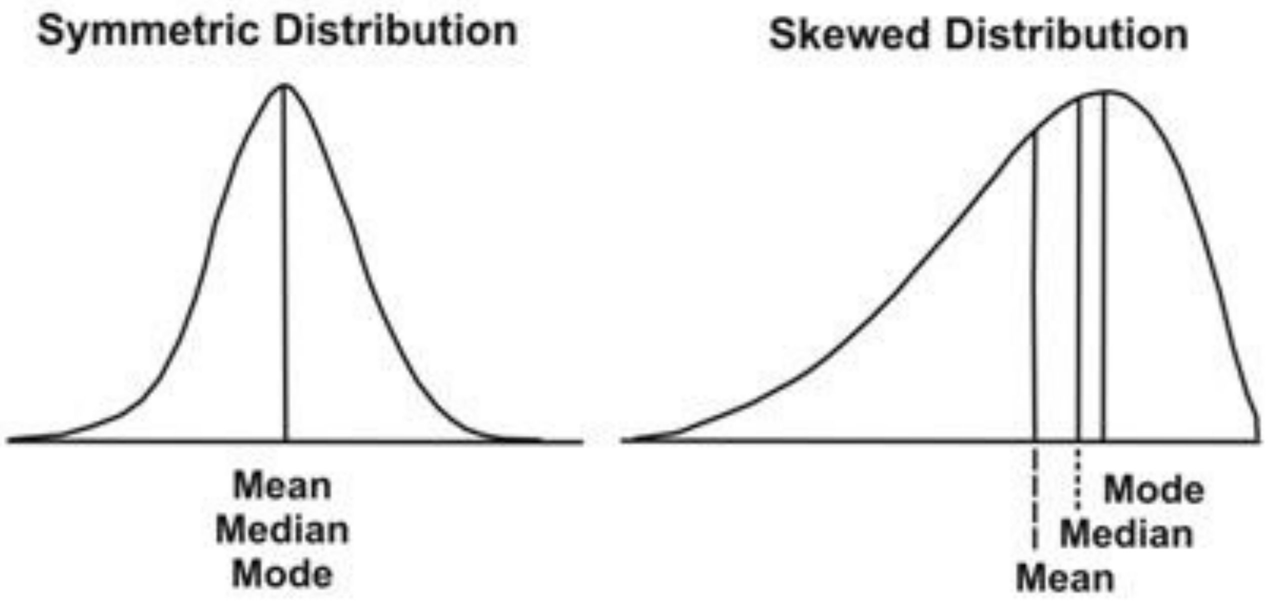
A list or sequence is a data type that implements a finite **ordered** collection of values.

List in python

```
>>> mylist=[3, 4, 5, 11, 9]
>>> print mylist[2]
5
>>> print mylist[1:3]
[4, 5]
>>> mylist[-1]
9
>>> mylist[1]=‘a’
>>> print mylist
[3, ’a’, 5, 11, 9]
>>> print mylist+[True]
[3, ’a’, 5, 11, 9, True]
>>> mylist.append(True)
[3, ’a’, 5, 11, 9, True]
```

# Basics in Statistics

Given a set of values *A = [*$x_1$*,* $x_2$*,* $x_3$*, …*$x_N$*]* that defines a probability distribution *p* we can calculate the following measures



Mean: $$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Standard Deviation: $$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Median: $$m \mid \ \mathrm{P}(X \leq m) \geq \frac{1}{2} \text{ and } \mathrm{P}(X \geq m) \geq \frac{1}{2}$$

where *P* is the probability.

# Statistics with Python

Scipy is the most popular library of scientific tools in python that implements many statistical functions. Basic statistical functions are also available in numpy.

```
>>> import scipy
>>> A=[2,2,3,4,5,10,7]
>>> scipy.mean(A)
4.7142857142857144
>>> scipy.std(A)
2.9277002188455996
>>> scipy.median(A)
4.0
```

Using the module stats, scipy can be used to generate random normal distributions with given mean and standard deviation:

```
>>> import scipy.stats as stats
>>> dist=stats.norm(10,1)
>>> stats.rvs(size=10)
array([  9.63517267,   8.88827343,   9.01236771,   9.40981448,
        10.25279873,  11.09357001,  10.20825657,   8.52022001,
         9.99897057,   9.07548534])
```

# Exercise I

Write a simple python script that generates *n* random number with given mean m and standard deviation s. Calculate the mean and median of the generated numbers. Use the sys module that is used to capture the input variables of the script: mean (n), standard deviation (s) and number of values (n).

## The module sys

access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter.

```
>>> import sys
>>> sys.argv[0]
```

**Suggestion**:
sys.argv returns list of all the values provide in input to the script. Try yourself creating and executing a file *testsys.py* the contains lines:

```
#!/urs/bin/python
import sys
print sys.argv
```

# Files (I)

In python files are represented as objects:

Creating a file object in python

```
f=open(filename,mode)        # object file f is associated to filename
```

modes are: read ('r'), write ('w'), and append ('a').

Most important methods on the object file in write mode

```
>>> f=open(‘file.txt’,’w’)
>>> f.write(“Hello world!”)                 # The argument is a string
>>> f.writelines([‘ Emidio’, ‘ Malay’])     # The argument is a list
>>> f.close()
```

Most important method in on the object file in read mode

```
>>> f=open(‘file.txt’,’r’)
>>> cont = f.read()
>>> print cont
Hello world! Emidio Malay
```

# Files (II)

Use *readlines* for reading file

```
>>> f=open(‘file.txt’,’r’)
>>> cont = f.readlines()
>>> print cont
[‘Hello world! Emidio Malay’]
```

File object are similar to a stack

```
>>> f=open(‘file.txt’,’r’)
>>> print f.read(5)
‘Hello’
>>> print f.read(50)
‘ world! Emidio Malay’
>>> print f.read(50)
‘’
```

In text file you can have special characters “\t” tab and “\n” newline

# A good practice

It is a good programing practice when you open a file to read one line at the time. To avoid that huge file can make your machine crashing.

Read one line at the time

```
with open(filename,’r’) as fobj:
        for line in fobj:
                do_something(line)
```

Read one line at the time the friend_file.txt file

```
>>> c=0
>>> with open(‘friend_file.txt’,’r’) as fobj:
…               for line in fobj:
…                       c=c+1
…                       print c,line.rstrip()
1 Pietro        42
2 Zef           42
3 Tommy         43
```

An alternative is to import the module fileinput

```
import fileinput
for line in fileinput.input(['myfile']):
        do_something(line)
```

# Exercise II

Write a python script that calculates the mean, standard deviation and median of the numbers in a give column of a file. Use the sys module to take from the input the file name and the column number.

**Suggestion**:

You should read the file line by line and generate a list that contains all the numbers extracted from the file.

**Warning**:

Although your file contains numbers when read by the program they are seen as string. Therefore you need to used the command *float* to change transfer the string into a float. Similar procedure has to be done for the column number in input using *int*

```
>>> a=“1”
>>> float(a)
1.0
```

# Dictionary

Dictionaries are not ordered lists indexed by keys, which can be any immutable type; strings and numbers can always be keys.

Tuples can be used as keys if they contain only strings, numbers, or tuples.

The list used in the previous exercise can be stored as a dictionary

```
>>> fdic= {‘Pietro’:42, ‘Zef’: 42, ‘Tommy’: 43}
>>> print fdic[‘Pietro’]
42
>>> print fdic.get(‘Pietro’,0)
42
>>> print fdic.get(‘Goofy’,0)
0
>>> print fdic.keys()
[‘Pietro’,’Zef’,’Tommy’]
>>> print fdic.values()
[42, 42, 43]
>>> for key,value in fdic.iteritems():
          print key, value
>>> dic= {(0,True):1, (0:False):2, (1,True):3, (1:False):4}
>>> print  dic[(0,True)]
```

# More exercises

1. Write a python script that takes in input a fasta file of only one protein sequence and calculate the frequency of each amino acid. Check the output of for the files:
   http://www.uniprot.org/uniprot/P53_HUMAN.fasta
   http://www.uniprot.org/uniprot/BRCA1_HUMAN.fasta

2. Write a code that analyze a fasta file with multiple sequences and calculate the length of each sequence. Save the results in a file.