Lab of Signal Processing for Big Data M

Alex Marchioni

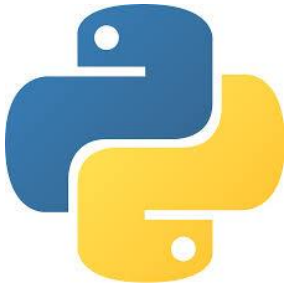
Learning Outcomes

- Implementation of some of the algorithms treated in the module of STATISTICS AND ARCHITECTURES FOR BIG DATA PROCESSING M in a suitable programming environment
- Performance assessment and discussion of the implemented algorithm.
- Writing of a report.

Outline

- Pyhton
- Jupyter Notebook
- Markdown
- Writing a Report
- Tasks

Python



Python

Python is an Interpreted and Object-Oriented Programming Language.

WHY Python?

- Simple syntax
- Very flexible
- Highly extensible
- Cross-platform
- Open-source with a huge community

Google says: *Python where we can, C++ where we must*

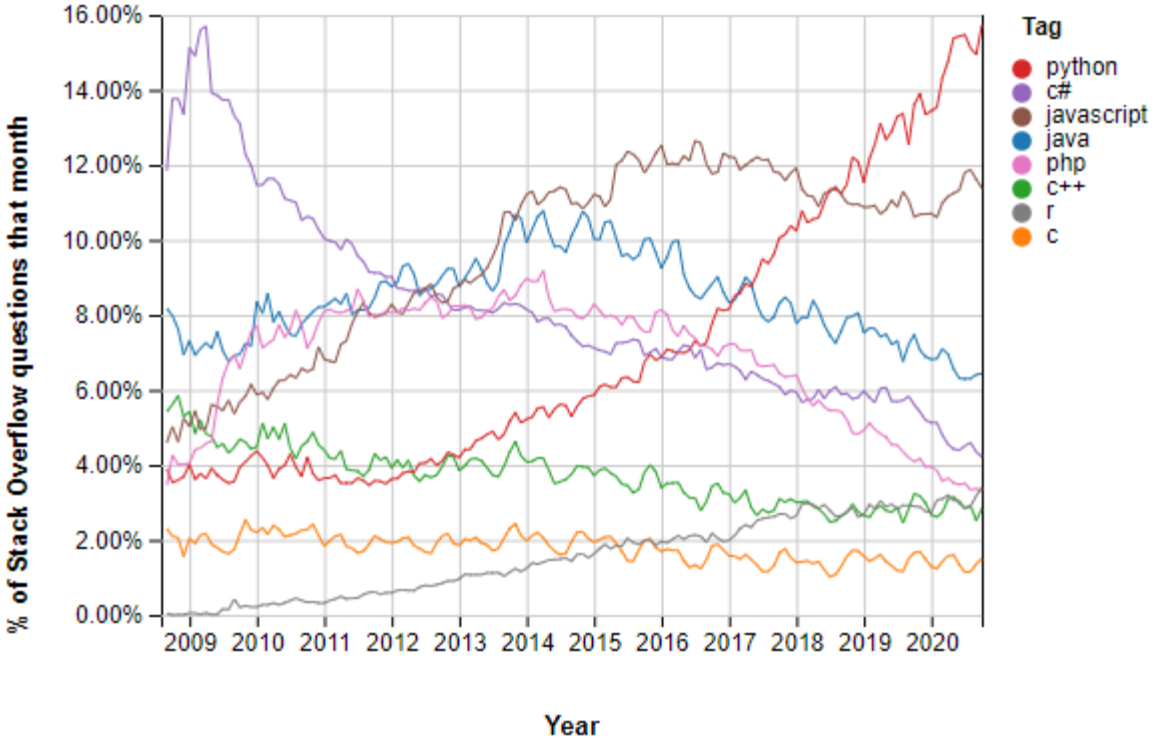
A bit of History

- It is not so recent, since it was conceived in the late 1980s and implemented in 90s by Guido van Rossum.
- Name is a tribute to Monty Python (a British surreal comedy group)
  in fact, as metasyntactic variables spam and eggs preferred to the traditional foo and bar
- **Python 2.0** released on Oct 2000, not supported since 01/2020
- **Python 3.0** released on Dec 2008
- **Last release** is **3.9.0** on Oct 2020 (https://www.python.org/downloads/)

Here a great Hackaday's post: Stop using Python2: What you need to know about Python3

Popularity

- According to StackOverflow's survey and trends Python among all programming languages is the:
  - **1st** most questioned
  - **4th** most used
    behind JavaScript, HTML/CSS, and SQL
  - **2nd** most loved
    behind Rust
  - **1st** most wanted
    developers who do not yet use it say they want to learn it



Applications for Python

- Web and Internet Development
- Scientific and Numeric
- Education
- Desktop GUIs
- Software Development
- Business Applications

Basically anything, like English for spoken languages

Scipy

Scipy.org = Python for math/science/engineering

- **Numpy**: Numerical Python package (inspired by Matlab)

  N-dimensional array capabilities and some linear algebra, Fourier analysis, random number capabilities, etc.

- **Scipy**: Scientific Python

  For Matlab users, it's very much like many of the core toolboxes.

- **Matplotlib:** most popular data visualization package for Python

  Inspired by Matlab plots, but then it has evolved into something more.

- **Pandas:** Data Science Python

  high-performance, easy-to-use data structures and data analysis tools

Jupyter Notebook

Jupyter Notebook

open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text.

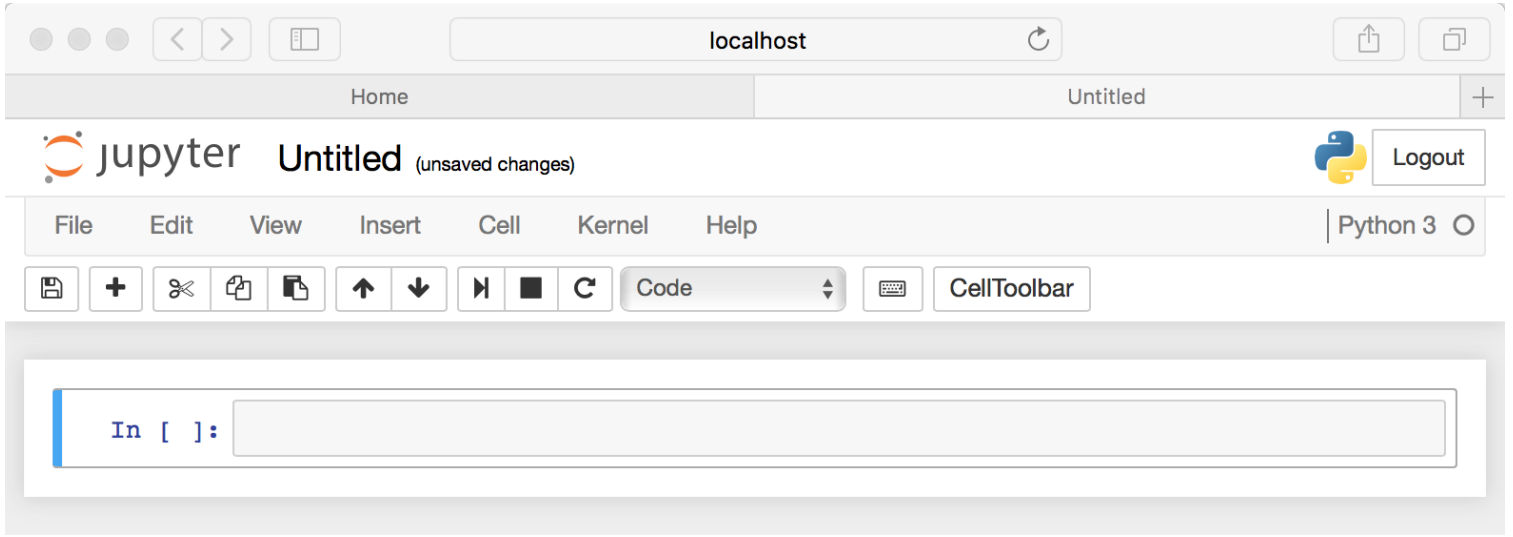
The name comes from the core supported programming languages that it supports: **Ju**lia, **Pyt**hon, and **R**

The application can be executed on a PC without Internet access, or it can be installed on a remote server, where you can access it through the Internet.

Jupyter Notebook

Two main components:

- **Kernel**: program that runs and introspects the user's code.
- **Dashboard**: shows you the notebook documents and manage the kernels (see which are running or shut them down)



Jupyter Notebook

In [1]:
```
print 'Hello World'
```

```
Hello World
```

Getting started with Python

We have done the following

- installed Python
- started iPython Notebook

Create variables in Python

In [3]:
```
i = 4 # int
```

In [4]:
```
type(i)
```

Out[4]:
```
int
```

In [5]:
```
f = 3.14
```

In [6]:
```
type(f)
```

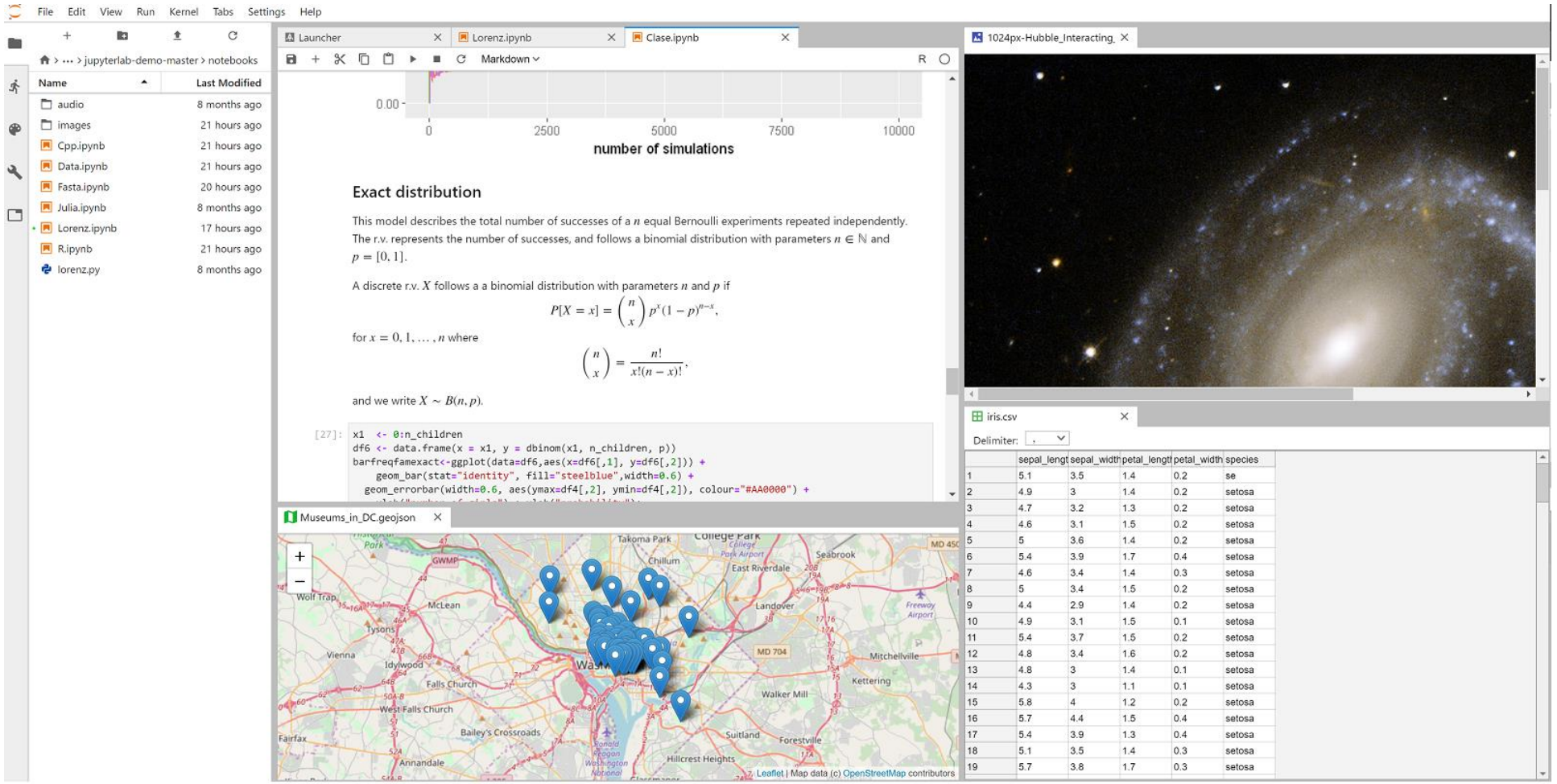
Out[6]:
```
float
```

Jupyter Lab

JupyterLab is the next-generation user interface for Project Jupyter offering all the familiar building blocks of the classic Jupyter Notebook (notebook, terminal, text editor, file browser, rich outputs, etc.) in a flexible and a more powerful user interface.

https://towardsdatascience.com/jupyter-lab-evolution-of-the-jupyter-notebook-5297cacde6b

Jupyter Lab


File Edit View Run Kernel Tabs Settings Help
jupyterlab-demo-master > notebooks
Name
Last Modified
audio 8 months ago
images 21 hours ago
Cpp.ipynb 21 hours ago
Data.ipynb 21 hours ago
Fasta.ipynb 20 hours ago
Julia.ipynb 8 months ago
Lorenz.ipynb 17 hours ago
R.ipynb 21 hours ago
lorenz.py 8 months ago
Launcher
Lorenz.ipynb
Clase.ipynb
Markdown
number of simulations
Exact distribution
This model describes the total number of successes of a n equal Bernoulli experiments repeated independently. The r.v. represents the number of successes, and follows a binomial distribution with parameters $n \in \mathbb{N}$ and $p = [0, 1]$.
A discrete r.v. X follows a a binomial distribution with parameters n and p if
$$P[X = x] = \binom{n}{x} p^x (1-p)^{n-x},$$
for $x = 0, 1, \ldots, n$ where
$$\binom{n}{x} = \frac{n!}{x!(n-x)!},$$
and we write $X \sim B(n, p)$.
[27]:
x1 <- 0:n_children
df6 <- data.frame(x = x1, y = dbinom(x1, n_children, p))
barfreqfamexact<-ggplot(data=df6,aes(x=df6[,1], y=df6[,2])) +
geom_bar(stat="identity", fill="steelblue",width=0.6) +
geom_errorbar(width=0.6, aes(ymax=df4[,2], ymin=df4[,2]), colour="#AA0000") +
Museums_in_DC.geojson
Leaflet | Map data (c) OpenStreetMap contributors
1024px-Hubble_Interacting.
iris.csv
Delimiter:
sepal_lengt sepal_width petal_lengt petal_width species
1 5.1 3.5 1.4 0.2 se
2 4.9 3 1.4 0.2 setosa
3 4.7 3.2 1.3 0.2 setosa
4 4.6 3.1 1.5 0.2 setosa
5 5 3.6 1.4 0.2 setosa
6 5.4 3.9 1.7 0.4 setosa
7 4.6 3.4 1.4 0.3 setosa
8 5 3.4 1.5 0.2 setosa
9 4.4 2.9 1.4 0.2 setosa
10 4.9 3.1 1.5 0.1 setosa
11 5.4 3.7 1.5 0.2 setosa
12 4.8 3.4 1.6 0.2 setosa
13 4.8 3 1.4 0.1 setosa
14 4.3 3 1.1 0.1 setosa
15 5.8 4 1.2 0.2 setosa
16 5.7 4.4 1.5 0.4 setosa
17 5.4 3.9 1.3 0.4 setosa
18 5.1 3.5 1.4 0.3 setosa
19 5.7 3.8 1.7 0.3 setosa

Markdown

Markdown

- One of the world's most popular **markup language** created by John Gruber in 2004.
- Differently from WYSIWYG (what you see is what you get) editor where format changes are visible immediately, in Markdown you add Markdown syntax to the text to indicate which words and phrases should look different.
- Markdown syntax is designed to be readable and unobtrusive, so the text in Markdown files can be read even if it isn't rendered

https://www.markdownguide.org/getting-started

Markdown

What you see

```
A First Level Header
====================

A Second Level Header
---------------------

Now is the time for all good men to come to
the aid of their country. This is just a
regular paragraph.

The quick brown fox jumped over the lazy
dog's back.

### Header 3

> This is a blockquote.
>
> This is the second paragraph in the blockquote.
>
> ## This is an H2 in a blockquote
```

What you get

A First Level Header

A Second Level Header

Now is the time for all good men to come to the aid of their country. This is just a regular paragraph.

The quick brown fox jumped over the lazy dog's back.

Header 3

> This is a blockquote.
>
> This is the second paragraph in the blockquote.
>
> ## This is an H2 in a blockquote

[Dillinger] is one of the best online Markdown editors.

Why Use Markdown?

- can be used for **everything** and is **everywhere**. It is used for websites, documents, notes, books, presentations, email messages, and technical documentation and is supported by websites like Reddit or GitHub, by lots of desktop and by web-based applications .
- is **portable** and **platform independent**. Files containing Markdown-formatted text can opened using virtually any application on any device running any operating system.
- is **future proof**. In any case you are be able to read your Markdown-formatted text using a text editing application.

Writing a Report

Writing a Report

Though underestimated, communicating what an engineer has done is an essential part of her/his work that makes a difference as much as her/his engineering skills.

“engineers spend between 20% and 40% of their workday writing, a figure that increases as they move up the career ladder. Another study indicates that in their first few years on the job, engineers spend roughly 30% of their workday writing, while engineers in middle management write for 50% to 70% of their day; those in senior management reportedly spend over 70% and as much as 95% of their day writing.”

J. A. Leydens, "Novice and Insider Perspectives on Academic and Workplace Writing: Toward a Continuum of Rhetorical Awareness," in *IEEE Transactions on Professional Communication*, vol. 51, no. 3, pp. 242-263, Sept. 2008.

Report Structure

- Title
  - Do not forget it.
- Introduction
  - Contextualize your work.
  - Show the aim of the work.
  - Clear your contribution (and for everything else, you must cite the sources).
- Work Description
  - Describe the model of the problem and the parameters it depends on.

Report Structure

- Numerical Evidences
  - Describe the metrics used for performance assessment.
  - Describe how experiments are performed.
  - Show and explain the experimental results (always prefer quality to quantity).
- Conclusion
  - Summarize the whole work remarking the message you want to convey.
- References
  - You never reinvent the wheel so cite what you start from.

Tips

- Keep in mind that the target reader is an engineer as you but with no knowledge about what you are going to present. Therefore, **be clear** and **take nothing for granted**.
- **Be sythetic**. There needs to be all the necessary information but they must be expressed as synthetic as possible.
- **Be precise**, a report cannot be vague. Every sentence must be supported by a proof or by measures otherwise there needs to be a reference where a proof/measure is provided.

Tasks

Streaming setting

To build a streaming setting we need:

- A source for our **stream**, that is an object that can generate a potentially infinite number of symbols.
- A kind of **function** that can be applied to each stream element separately maintaining an internal state which is updated every time the function is called.

In Python these two functionalities are provided by **generators/iterators** and **classes**.

Python Generators

A generator is an object that you can iterate over and that generates and output at each iteration.

It is defined with the `def` keyword as a function, but the output is returned with the `yield` keyword instead of return.

Here an example of a generator that can generate up to N random numbers with a uniform distribution.

```
def generator(N):
    import random
    for i in range(N):
        yield random.uniform(0,1)
```

Some good reference to get started with Python Generators/Iterators:

- **Hackaday**: Learn to loop the Python way: Iterators and Generators Explained
- **Tutorialspoint**: Python Basics: Iteration, Iterables, Iterators, and Looping
- **Programiz**: Python Iterators, Python Generators
- **wiki.python.org**: Iterator, Generators

Python Generators

Here it is the same generator that potentially never stops, where the for loop is replaced by a `while True`.

To practically deal with an infinite stream we need the `islice` function that allows you to select only the first N elements of an iterable.

Inside the `for` loop that iterate over the stream, you get one sample at a time and you can try you streaming algorithms.

```
# generator definition
def generator():
    import random
    while True:
        yield random.uniform(0,1)

from itertools import islice

N = ... # number of iterations
stream = generator() # generator instance

# iterate over the first N samples
for sample in islice(stream, N):
  # do stuff
  out = streaming_algorithm(sample)
```

Python Classes

Classes are the foundation of the **Object Oriented Programming** (**OOP**).

An "object" may contain data in the form of **attributes**, and code in the form of **methods**.

A class defines how the objects should be: their status and the actions that they can perform to create their status. To create a particular specimen of a certain class is said creating an **instance** of that class.

Some good reference to get started with Python Classes:

- **docs.python.org**: https://docs.python.org/3/tutorial/classes.html
- **Programiz**: https://www.programiz.com/python-programming/class
- **Tutorialspoint**: https://www.tutorialspoint.com/python/python_classes_objects.htm

Python Class

`class` keyword defines a class.

`self` keyword refers to the class instance yet-to-be-created (the same as *this* in Java).

`__init__` is the constructor method, that is the function called when an instance is created.

Once a class instance is created the dot symbol `.` is used to access to attributes and methods.

```
# class definition
class Classname():

  def __init__(self, init_args):
    ...
    self.attribute = ...
    ...

  def method(self, args):
    ...

# creating a class instance
instance_name = ClassName(init_args)

# access to attribute
att = instance_name.attribute

# use class method
instance_name.method(args)
```

Streaming setting in Python

A **generator** to define the source of the stream.

A **class** for the streaming algorithm so that some attributes can be used to store the state and a method can implement the procedure to be applied to each incoming new stream sample.

A **for loop** with a finite number of iterations to test your implementation.

```
# define a class for the algorithm
class StreamingAlgorithm():
  ...

# define a generator for the stream
def generator(gen_args):
  ...

# create a class and a generator instance
stream = generator(gen_args)
alg = StreamingAlgorithm(alg_args)

# apply algorithm to the stream
for sample in islice(lfsr, N):
  out = alg.method(sample)
```

Task 1

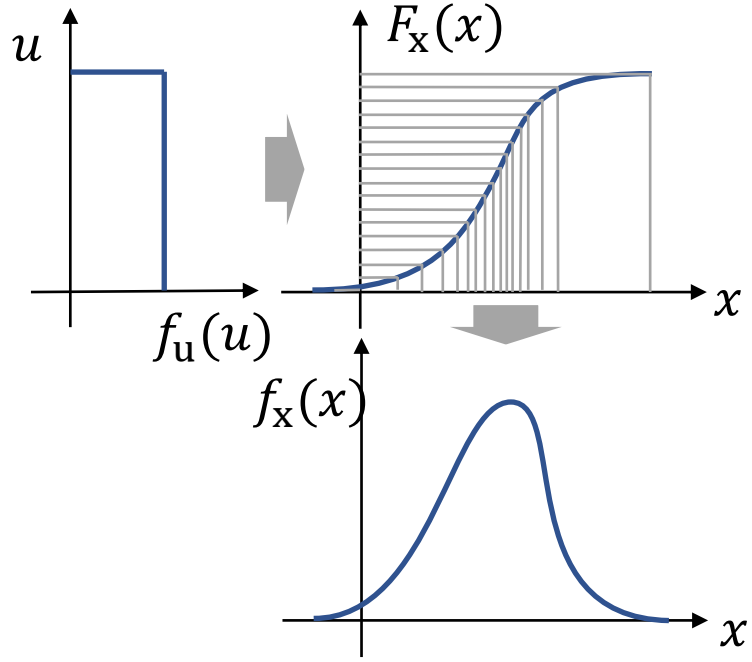
- Given a random variable (RV) and its probability distribution (in the form of cdf – cumulative density function), implement a Python generator that generates a instances of the RV in streaming fashion. Each RV instance must be generated by means of the inverse transform sampling (see next slides).
- Given a stream of instances of a RV, compute mean and variance in a streaming fashion by means of the Welford's algorithm (see next slides).

Inverse transform sampling

Method for generating random samples from any probability distribution given their cumulative distribution function (cdf), i.e., an instance of a random variable x with pdf f_{x}, and cdf F_{x}.

Let us consider $\mathrm{u} \sim U(0,1)$, We want to find a strictly monotone transformation $T: [0,1] \mapsto \mathbb{R}$ such that $T(\mathrm{u}) = \mathrm{x}$.

$$F_{\mathrm{x}}(x) = P(\mathrm{x} \leq x) = P(T(\mathrm{u}) \leq x)$$
$$= P\left(\mathrm{u} \leq T^{-1}(x)\right) = T^{-1}(x)$$
$$\Rightarrow x = T(u) = F_{\mathrm{x}}^{-1}(u)$$



Inverse transform sampling

Since $x = F_{\mathrm{X}}^{-1}(u)$, there is the need to compute the inverse of a cdf.

In this task you are asked to implement two methods:

- **cdf discretization**:
  discretize the cdf function so that, given a u instance u, you can approximate the correspondent x;
- **bisection**:
  given a u instance u find the x such that $F_{\mathrm{X}}(x) - u = 0$.

Online Mean and Variance Computation

Welford's Algorithm:

Method to compute mean M and variance V by considering one sample x_i at a time.

B. P. Welford, "Note on a method for calculating corrected sums of squares and products", Technometrics, vol. 4, no. 3, pp. 419–420, Aug. 1962

input: $x = (x_1, \dots, x_n)^\top$

$m \leftarrow x_1, S \leftarrow 0$

for $k \leftarrow 2, \dots, n$ **do**

$M \leftarrow m + (x_k - m)/k$

$S \leftarrow S + (x_k - m)(x_k - M)$

$m \leftarrow M$

end for

$V \leftarrow S/(n-1)$

output: M, V

Task 2

- Given a covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$ and a mean vector $\boldsymbol{\mu} \in \mathbb{R}^{n}$, generate a dataset composed by N instances of gaussian random vectors (consider a dimension $n \geq 10$).
- Estimate covariance matrix of the generated dataset and perform eigendecomposition.
- Extract the highest eigenvalue and its correspondant eigenvector by means of the Power Method. Discuss accuracy and convergenge of the algorithm.

Random Normal Vector Generation

Method for generating a random gaussian vector $\boldsymbol{x}$ from any mean vector $\boldsymbol{\mu} \in \mathbb{R}^n$ and any covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$, i.e., $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

Let us consider a white normal vector $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{0}_n, \boldsymbol{I}_n)$, we want to find a linear operator $\boldsymbol{U}: \mathbb{R}^n \mapsto \mathbb{R}^n$ such that $\boldsymbol{x} = \boldsymbol{U}\boldsymbol{y} + \boldsymbol{\mu}$. Its covariance matrix is

$$\begin{aligned} \boldsymbol{\Sigma} &= \mathbb{E}[\boldsymbol{x}\,\boldsymbol{x}^\top] = \mathbb{E}[\boldsymbol{U}\boldsymbol{y}\,\boldsymbol{y}^\top\boldsymbol{U}^\top] \\ &= \boldsymbol{U}\mathbb{E}[\boldsymbol{y}\,\boldsymbol{y}^\top]\boldsymbol{U}^\top = \boldsymbol{U}\boldsymbol{I}_n\boldsymbol{U}^\top = \boldsymbol{U}\boldsymbol{U}^\top \end{aligned}$$

Considering that $\boldsymbol{\Sigma} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top$ where $\boldsymbol{\Lambda} \in \mathbb{R}^{n \times n}$ diagonal matrix whose elements are the eigenvalues ($\Lambda_{i,i} = \lambda_i$) and $\boldsymbol{V} \in \mathbb{R}^{n \times n}$ is an orthonormal matrix whose columns $\boldsymbol{v}_i$ are eigenvectors correspondent to λ_i.

By choosing $\boldsymbol{U} = \boldsymbol{V}\sqrt{\boldsymbol{\Lambda}}\boldsymbol{V}^\top$, we get $\boldsymbol{U}\boldsymbol{U}^\top = \boldsymbol{V}\sqrt{\boldsymbol{\Lambda}}\boldsymbol{V}^\top\left(\boldsymbol{V}\sqrt{\boldsymbol{\Lambda}}\boldsymbol{V}^\top\right)^\top = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top = \boldsymbol{\Sigma}$

Then an instance $\boldsymbol{x}$ can be generated as follows: $\boldsymbol{x} = \boldsymbol{U}\boldsymbol{y} + \boldsymbol{\mu} = \boldsymbol{V}\sqrt{\boldsymbol{\Lambda}}\boldsymbol{V}^\top\boldsymbol{y} + \boldsymbol{\mu}$

Power Method

Iterative method that, given a diagonalizable matrix $\boldsymbol{\Sigma}$, estimates the greatest eigenvalue λ_1 and the corresponding eigenvector $\boldsymbol{v}_1$.

$$\boldsymbol{b}_k = \frac{\boldsymbol{\Sigma}\ \boldsymbol{b}_{k-1}}{\|\boldsymbol{\Sigma}\ \boldsymbol{b}_{k-1}\|_2},\ \{\boldsymbol{b}_k\} \xrightarrow[k\to\infty]{} \boldsymbol{v}_1$$

$$a_k = \frac{\boldsymbol{b}_k^\top \boldsymbol{\Sigma}\ \boldsymbol{b}_k}{\boldsymbol{b}_k^\top \boldsymbol{b}_k},\ \{a_k\} \xrightarrow[k\to\infty]{} \lambda_1$$

This is true if λ_1 is strictly greater than the 2° greatest eigenvalue λ_2 ($\lambda_1 > \lambda_2$) and if $\boldsymbol{b}_0 \neq \boldsymbol{0}_n$.

Task 3

1. Implementation, assessment and discussion of the streaming algorithm that counts the number of distinct values in a stream of symbols (example of credit cards).
2. Implementation, assessment and discussion of the streaming algorithm that finds the k most frequent values in a stream.