python™ in Microbial Ecology

by
Nicola Gambardella

ciimar

BIOPORTUGAL S.A.
Químico, Farmacêutica

SPECO

INTRODUCTION

Python is a **general-purpose**, **open-source** programming language.

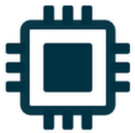
At the moment is one of the **most popular** programming languages in the world, meaning that:

- it is **easy** to get **help**
- it is **maintained** and **improved**
- **high** number of **libraries**

Python is an **easy** language **to read** and **write** due to its **high similarity** with human language.

It has **endless** possibilities provided by the **hundreds of specialized libraries**

Quantum Computing



QuTiP
PyQuil
Qiskit
PennyLane

Statistical Computing

Pandas
statsmodels
Xarray
Seaborn

Signal Processing

SciPy
PyWavelets
python-control
HyperSpy

Image Processing

Scikit-image
OpenCV
Mahotas

Graphs and Networks

NetworkX
graph-tool
igraph
PyGSP

Astronomy

AstroPy
SunPy
SpacePy

Cognitive Psychology

PsychoPy

Bioinformatics

BioPython
Scikit-Bio
PyEnsembl
ETE

Bayesian Inference

PyStan
PyMC3
ArviZ
emcee

Mathematical Analysis

SciPy
SymPy
cvxpy
FEniCS

Chemistry

Cantera
MDAnalysis
RDKit
PyBaMM

Geoscience

Pangeo
Simpeg
ObsPy
Fatiando a Terra

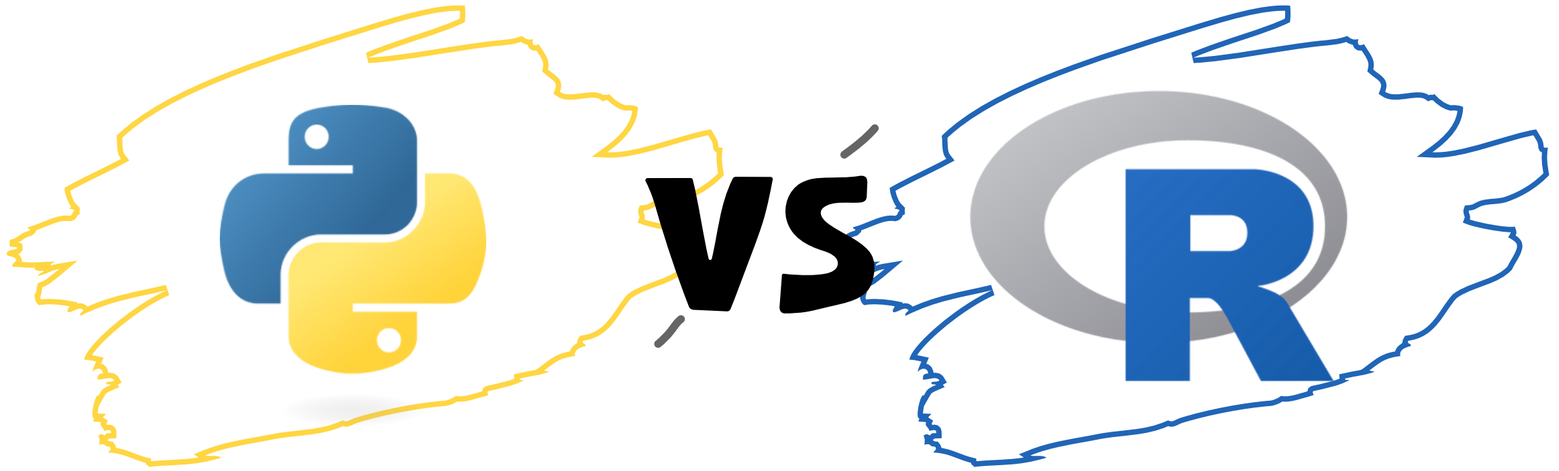
Geographic Processing

Shapely
GeoPandas
Folium

Architecture & Engineering

COMPAS
City Energy Analyst
Sverchok

https://numpy.org/



There is **no** single programming language that is **best** for **every** problem that may pop up during your career.

Programming languages are **not** mutually exclusive.

Programming Languages are **tools** and for this reason it is better to not focus on **only one** but rather use them together depending on your specific use case.

Ask yourself

- What programming language do your **colleagues** use?
- What **kind of problems** are you trying to solve?
- What are your **areas of interest**?

Jupyter Notebook

</>

Language of choice

Jupyter supports over 40 programming languages, including Python, R, Julia, and Scala.

Share notebooks

Notebooks can be shared with others using email, Dropbox, GitHub and the Jupyter Notebook Viewer.

Interactive output

Your code can produce rich, interactive output: HTML, images, videos, LaTeX, and custom MIME types.

https://jupyter.org

A computational notebook is a shareable document
that combines **computer code**, **plain language** descriptions, **data**, rich **visualizations**, and more.

A notebook provides a **fast interactive** environment
for prototyping and explaining code, exploring and visualizing data, and sharing.

It use **Markdown**, a **lightweight markup language** used to
add formatting elements to plain text documents

Jupyter Notebook

Jupyter notebook is an **IDE** (Interactive Development Environment) divided in **cells**.

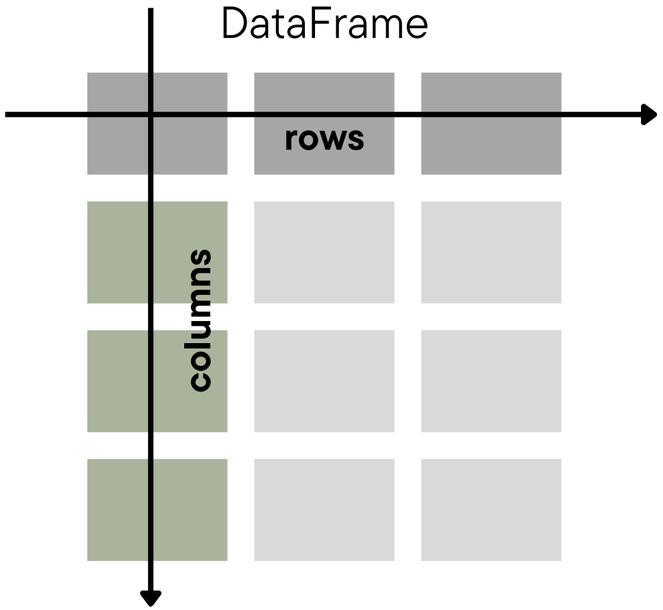
There are 2 types of cells:

- **code** cells, to write and run the code
- **markdown** cells, plain and enriched text, url, images

The output of a cell is printed rigth below the cell.

The notebook can be exported as **pdf**, **html**, **py** or other formats to be shared.

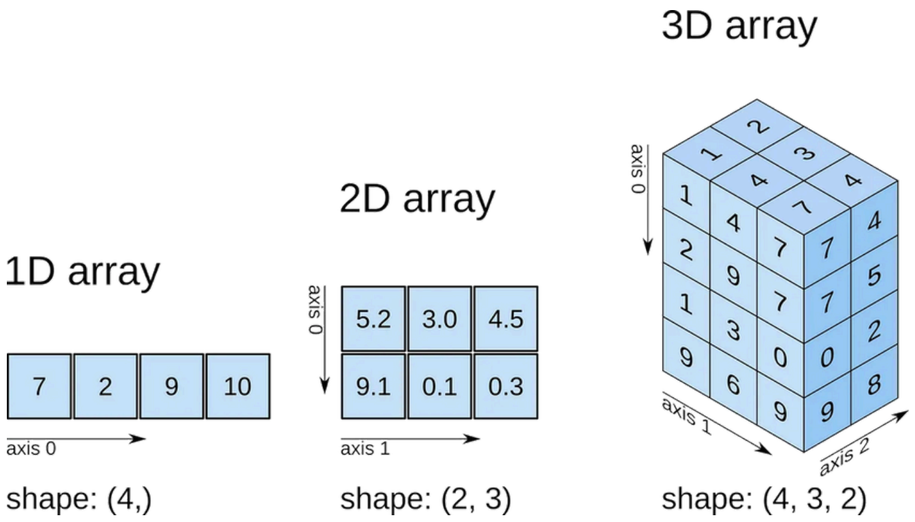
pandas



- Essential tool to manage **tabular data** (spreadsheets, databases, ...).
- It let you to **explore**, **clean**, and **process** your data.
- It supports the **integration** with many file **formats** and offers methods for **slicing**, **selecting**, and **extracting** the data of interest.
- pandas provides also methods to **plot** the data and allow to easily calculate **basic statistics** (mean, median, min, max, counts...).
- It also allow to perform **database-like operations** and several table **manipulations**.

numpy

- **Nearly all** operation in Python rely on **NumPy** functionalities
- NumPy offers comprehensive **mathematical functions**, random number generators, linear algebra routines, Fourier transforms, and more!



CODING TIME

Biopython is a collection of freely available Python **modules** for computational molecular biology.

Biopython includes parsers for various bioinformatics **file formats** (BLAST, Clustalw, FASTA, Genbank, ...), access to **online services** (NCBI, Expasy, ...), a standard sequence class, **sequence alignment** and **motif analysis** tools, **clustering** algorithms, a module for structural biology, and a module for **phylogenetics** analysis.

It provides the ability to parse bioinformatics files into **Python** usable **data structures**

```
from Bio import SeqIO

for record in SeqIO.parse("file.fasta", "fasta"):
    print(record.id)
    print(record.seq)

myseq = [record.seq for record in
SeqIO.parse("file.fasta", "fasta")]

from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord

rec1=SeqRecord(Seq("MMYQQGCFASAC",),
id="ID_1",description="myprotein1")

rec2=SeqRecord(Seq("MMYKKSSAC",),
id="ID_2",description="myprotein2")

myfile = [rec1,rec2]

SeqIO.write(myfile, "myfile.faa", "fasta")
```

Biological Observation Matrix

The BIOM file format is designed to be a **general-use** format for representing **biological** sample by observation **contingency** tables.

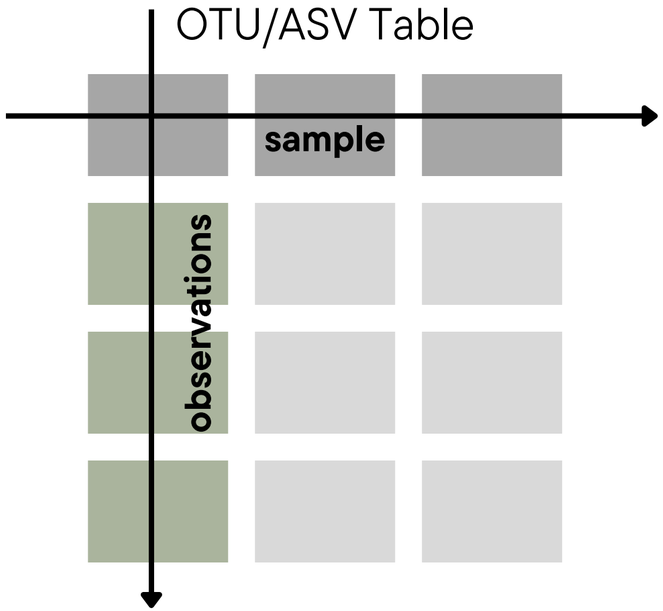
BIOM is a recognized standard in projects like the Earth Microbiome Project and is a Genomics Standards Consortium supported project.

in **marker-gene** analyis, this format is to represent **OTUs**/ASVs tables:

the **observations** are OTUs/ASVs and the **matrix** contains counts corresponding to the number of times each OTU/ASV is observed in each sample.

```
import pandas as pd
from biom import load_table

table = load_table('mytable.biom')

table.to_dataframe()
```

Biological Observation Matrix




```
print(load_table('feature-table.biom').ids(axis='sample')[:10])
```

```
['49' '56B' '60' '62B' '65B' '67C' '68' '68C' '68F' '69']
```

```
print(load_table('feature-table.biom').ids(axis='observation')[:10])
```

```
['81023771cad8637c6b48e60c873a1b47' '8cbcfed6c0090265c4d7e204585bcc34'
 'f4dcd86ee0af72bfc8e39524eb37a1a3' 'd07a78258b9434e3161b335e8ac98550'
 'b92c836c4467ba1258c3712857bfd431' '2bdf07c8fb0ba735018bc7d07496fdef'
 '496dc87a7244cb11dce3b8a647781eec' '7314dc937e1c9bd566101ce35a6792a2'
 '3c077535b76db277b7507f522a7c7cad' '85b453f32b99b6a4e4d9dddb568a18e3']
```

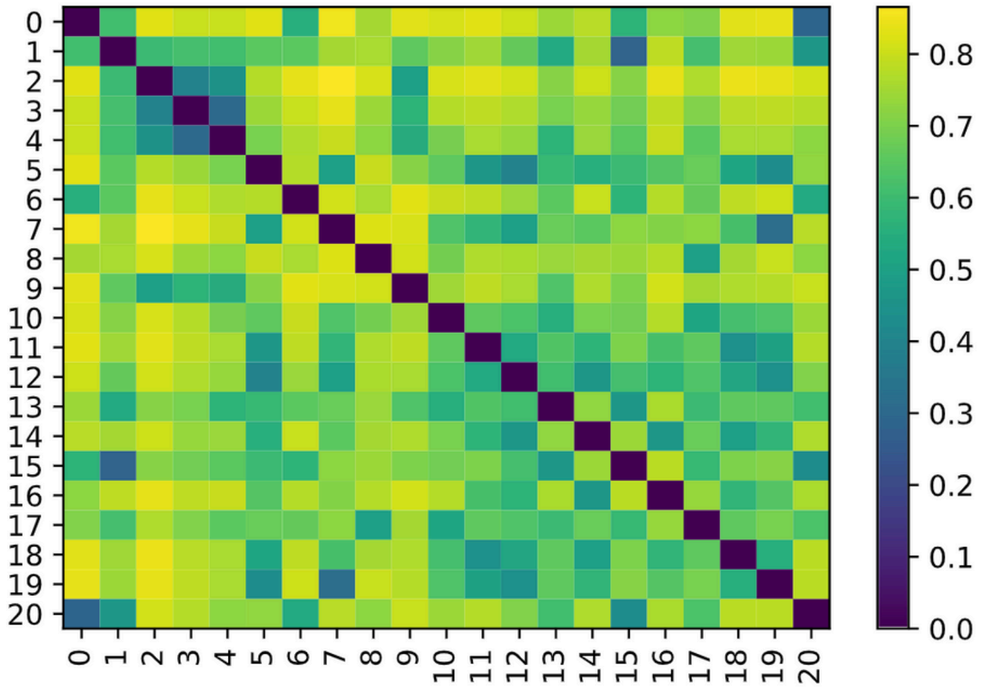
```
print(load_table('feature-table.biom').head())
```

```
# Constructed from biom file
#OTU ID 49      56B     60      62B     65B
81023771cad8637c6b48e60c873a1b47        38.0    901.0   8053.0  8373.0  10262.0
8cbcfed6c0090265c4d7e204585bcc34        3754.0  4367.0  617.0   958.0   633.0
f4dcd86ee0af72bfc8e39524eb37a1a3        230.0   1541.0  177.0   181.0   830.0
d07a78258b9434e3161b335e8ac98550        3994.0  27.0    10.0    0.0     9.0
b92c836c4467ba1258c3712857bfd431        87.0    2541.0  5863.0  1050.0  88.0
```

scikit-bio

```
from skbio import diversity #ecology
```

```
diversity.beta_diversity(counts=pacbio_feat_tab.transpose(),
                         metric='braycurtis')
```



skbio.sequence

This module provides functionality for storing and working with molecular sequences based on IUPAC-defined alphabets (DNA, RNA, Protein), custom alphabets (GrammaredSequence), and generic/non-biological sequences with no alphabet restrictions (Sequence).

skbio.diversity

This module provides functionality for analyzing biodiversity of communities. For example, to compute alpha and beta diversity.

skbio.metadata

This module provides functionality for storing and working with metadata – the data that describes other data.

skbio.workflow

It allow workflow construction in which the specific methods run are determined at runtime.

SciPy

SciPy provides **algorithms** for many classes of mathematical and statistical problems.

scipy.spatial.distance.pdist

Pairwise distances between observations in n-dimensional space.

scipy.cluster.hierarchy.linkage

Perform hierarchical/agglomerative clustering.

scipy.cluster.hierarchy.dendrogram

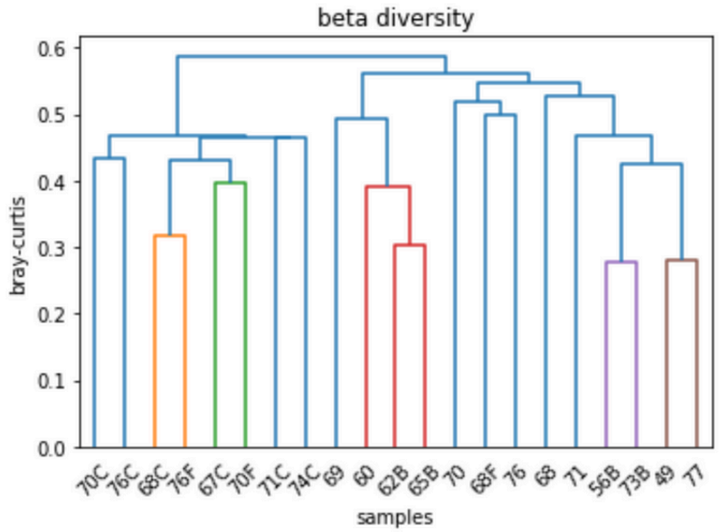
Plot hierarchical clustering as a dendrogram.

```
from scipy.cluster.hierarchy import linkage, dendrogram
from scipy.spatial.distance import pdist
```

```
pacbio_dist_matr = pdist(pacbio_feat_tab.transpose(), metric = 'braycurtis')
```

```
pacbio_link = linkage(pacbio_dist_matr, method = 'single', metric = 'braycurtis')
```

```
dendrogram(pacbio_link, orientation = 'top',
           labels = pacbio_feat_tab.transpose().index.tolist(),
          leaf_font_size=10)

plt.title('beta diversity')
plt.ylabel('bray-curtis')
plt.xlabel('samples')

plt.show()
```



Classification

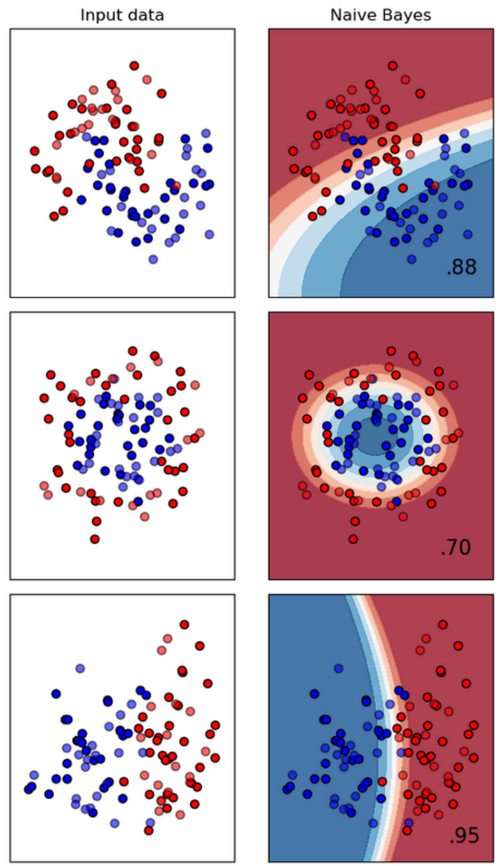
Identifying which category an object belongs to.

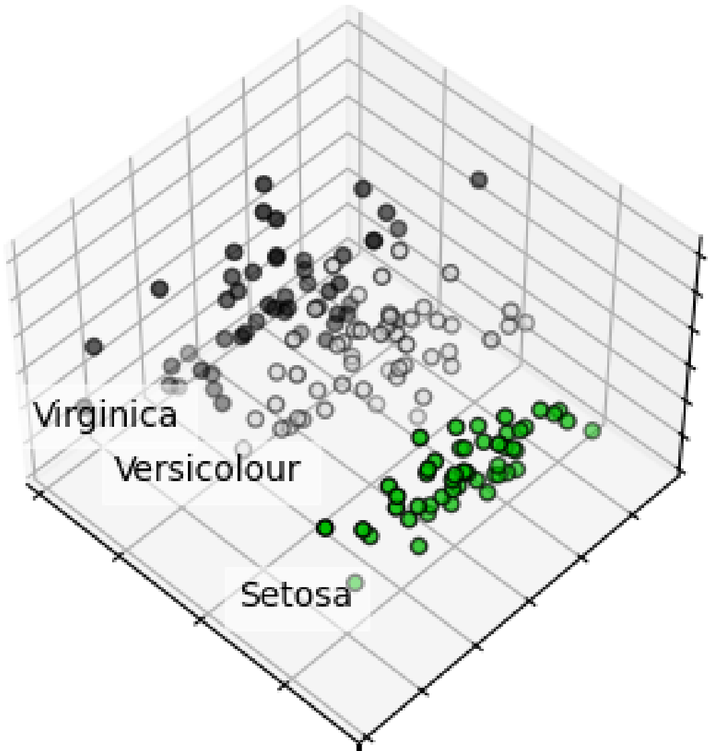
Taxonomy Classification

Dimensionality reduction

Reducing the number of variables to consider.

PCA

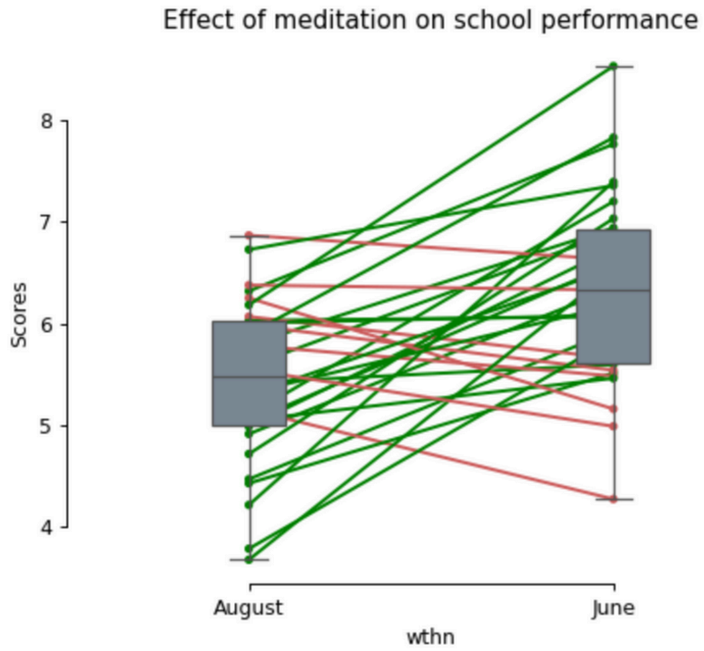




pingouin

Pingouin is an open-source **statistical** package based on **NumPy**, **Pandas**, and **SciPy** offering simple way to do statistics

```
import pingouin as pg
import numpy as np
df = pg.read_dataset('mixed_anova').query("Group == 'Meditation' and Time != 'Januar
ax = pg.plot_paired(data=df, dv='Scores', within='Time', subject='Subject')
ax.set_title("Effect of meditation on school performance")
```



Pearson's correlation

```
pg.corr(x, y)
```

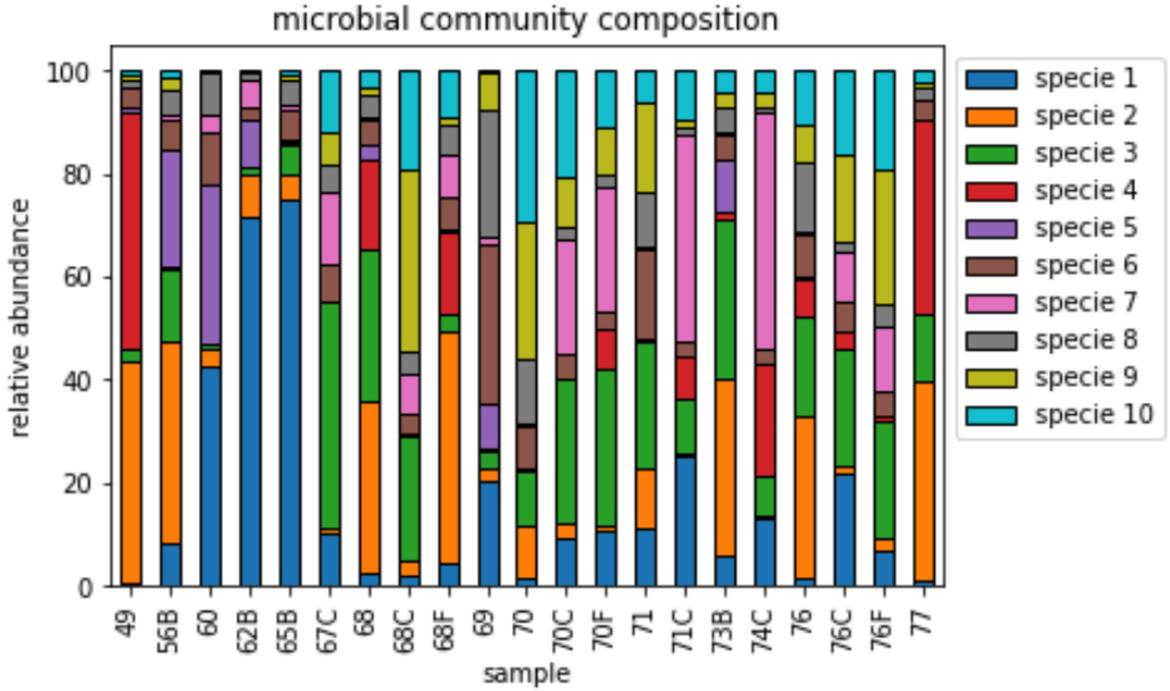
Output

n	r	CI95%	p-val	BF10	power
30	0.595	[0.3 0.79]	0.001	69.723	0.950

matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive **visualizations** in Python.

```
rel_ab_ex.transpose().plot(kind='bar',
                           stacked = True,
                           edgecolor='black',
                           legend = False)

plt.title('microbial community composition')

plt.legend([f'specie {i}' for i in range(1,len(rel_ab_ex.columns))],
           bbox_to_anchor = (1,1))

plt.xlabel('sample')
plt.ylabel('relative abundance')

plt.show()
```

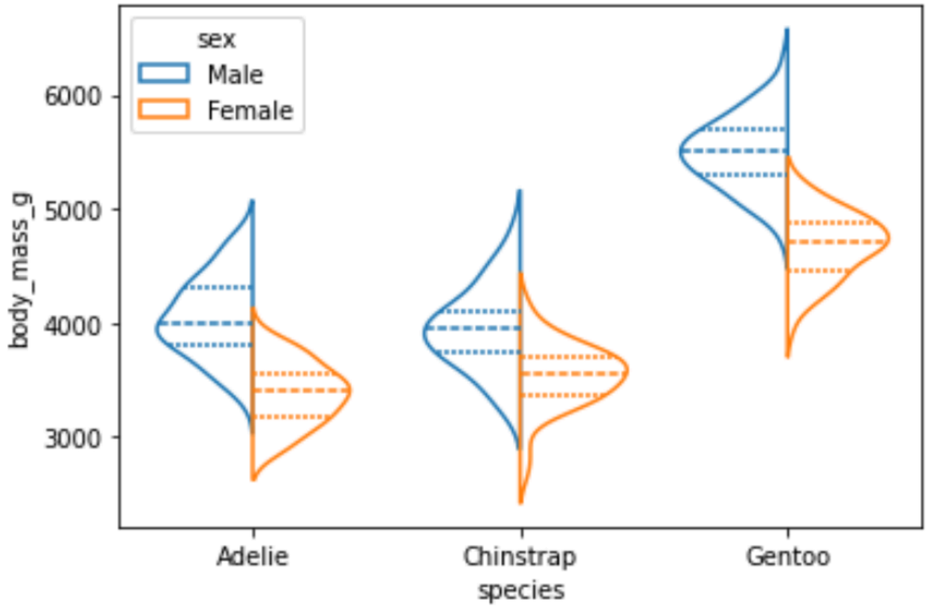


seaborn

Seaborn is a Python data **visualization** library based on matplotlib. It provides a **high-level interface** for drawing **attractive** and **informative** graphics.

More importantly, it **save** both **lines of code** and **time**. Losing, of course, in **flexibility**.

```
sns.violinplot(data = data, x = 'species',
                y='body_mass_g', hue='sex',
               split=True, fill = False,
               inner='quart')
plt.show()
```



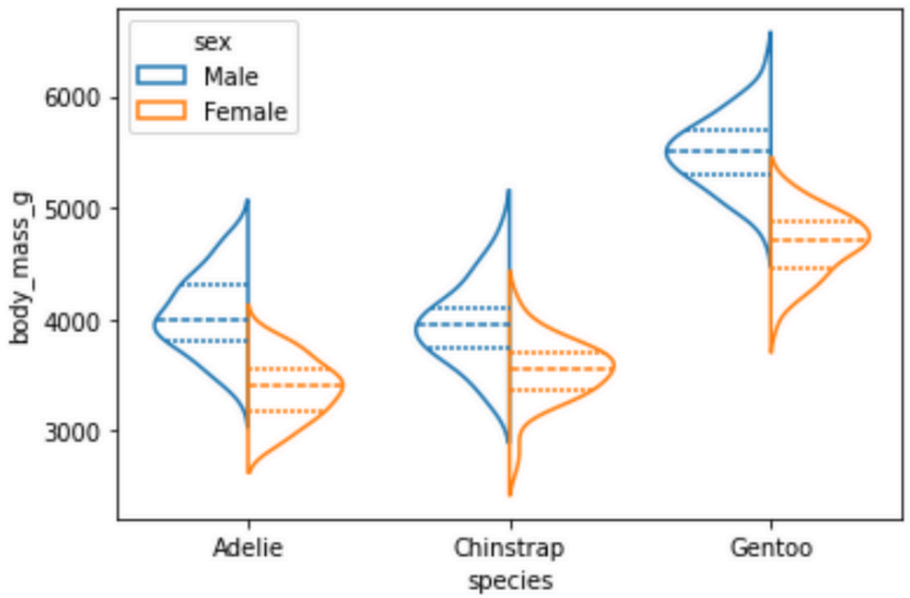
seaborn

```
sns.violinplot(data = data, x = 'species',
               y='body_mass_g', hue='sex',
              split=True, fill = False,
              inner='quart')
plt.show()
```

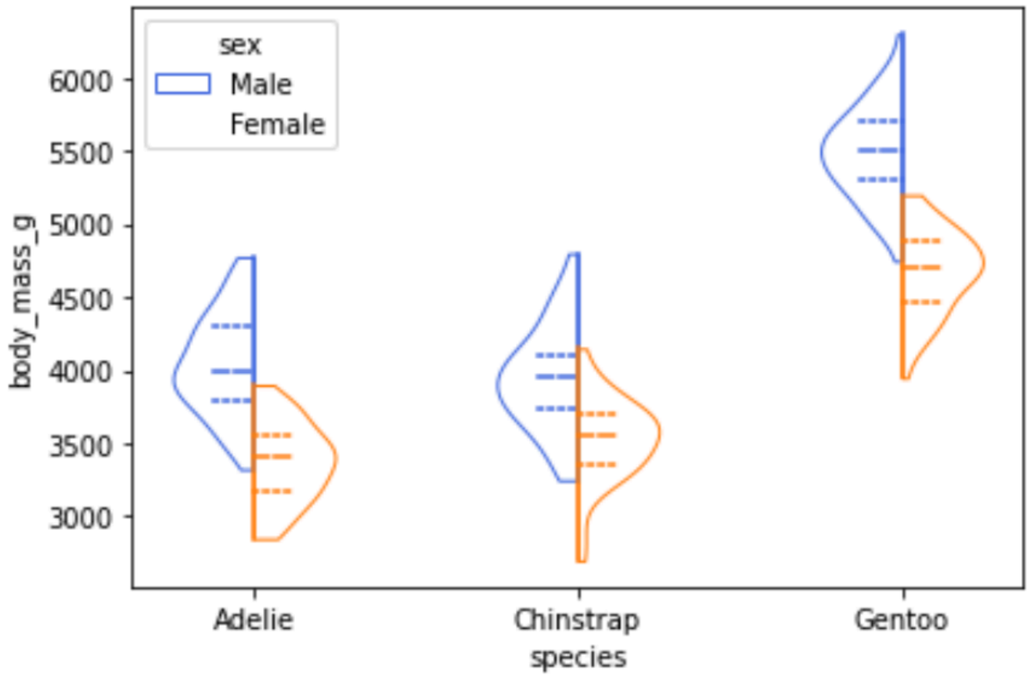
matplotlib

```
c={'Male':'royalblue','Female':'tab:orange'}
sd={'Male':'low','Female':'high'}

i = 0
for s in ['Adelie','Chinstrap','Gentoo']:
    for x in ['Male','Female']:

        part33 = plt.violinplot(dataset=data.query('species==@s&sex==@x')['body_mass_g'].dropna(),
                                quantiles=[0.25,0.5,0.75],
                                positions=[i],side=sd[x])
        for pc in part33['bodies']:
            pc.set_color('white')
            pc.set_edgecolor(c[x])
            pc.set_alpha(1)

        part33['cmins'].set_linewidth(0)
        part33['cmaxes'].set_linewidth(0)
        part33['cbars'].set_color(c[x])

        part33['cquantiles'].set_color(c[x])
        part33['cquantiles'].set_linestyle(['dotted','dashed','dotted'])


        i+=0
    i+=1

# for partname in ('cbars','cmins','cmaxes','cmeans','cmedians'):

plt.ylabel('body_mass_g')
plt.xlabel('species')
plt.xticks([0,1,2],['Adelie','Chinstrap','Gentoo'])

plt.legend(['Male','Female'],loc='upper left', title ='sex')

plt.show()
```

seaborn



matplotlib



CODING TIME

useful links

https://www.python.org/

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

https://jupyter-notebook.readthedocs.io/en/stable/

https://biopython.org/docs/latest/index.html

https://numpy.org/about/

https://pandas.pydata.org/

https://scikit.bio/index.html

https://scipy.org/

https://scikit-learn.org/stable/

https://biom-format.org/

https://matplotlib.org/

https://seaborn.pydata.org/

thank YOU

for your

ATTENTION

ciimar

BIOPORTUGAL S.A.
Químico, Farmacêutica

SPECO