Dr. Alexis Salas Burgos
2022
DIFAC version

Topics

Chapter 1 Introduction
Chapter 2 Quick Start – What can you do with Biopython?
Chapter 3 Sequence objects
Chapter 4 Sequence annotation objects
Chapter 5 Sequence Input/Output
Chapter 6 Multiple Sequence Alignment objects
Chapter 7 BLAST

Topics

Chapter 9 Accessing NCBI's Entrez databases
Chapter 10 Swiss-Prot and ExPASy
Chapter 11 Going 3D: The PDB module
Chapter 12 Bio.PopGen: Population genetics
Chapter 13 Phylogenetics with Bio.Phylo
Chapter 14 Sequence motif analysis using Bio.motifs
Chapter 15 Cluster analysis

Other Utils Modules

https://numpy.org
https://scipy.org
https://matplotlib.org
https://pandas.pydata.org
https://seaborn.pydata.org

pyData https://pydata.org

PyConference

https://pycon.cl

Pycon CL 2021 https://www.youtube.com/watch?v=18VvatmT8Qs

Pycon 2018 https://www.youtube.com/channel/UCsX05-2sVSH7Nx3zuk3NYuQ

PyCon 2019 https://www.youtube.com/channel/UCxs2IIVXaEHHA4BtTiWZ2mQ

pyCon 2021 US https://www.youtube.com/channel/UCMjMBMGt0WJQLeluw6qNJuA

Modules

Python functions are divided into three sets

- A small core set that are always available
- Some built-in modules such as math and os that can be imported from the basic install (ie. >>> import math)
- An number of optional modules that must be downloaded and installed before you can import them: code that uses such modules is said to have “dependencies”
- Most are available in different Linux distributions, or via pypy.org using pip (the Python Package Index)

Anyone can write new Python modules, and often several different modules are available that can do the same task

Object Oriented Code

Python implements object oriented programming

Classes bundle data and functionality

```
class Person:
    speech = "something in English"
    def __init__(self, name):
        """This creates a new instance of our Person class."""
        self.name = name

    def print_conv(self):
        """Prints out a conversation between two people"""
        print("P1: This person whose name is " + self.name
              + " is speaking something!")
        print("P2: But what is he speaking?")

    def answer(self):
        """Prints out what the person is saying"""
        print("P1: The person said " + self.speech
                  + " that was inaudible")
```

```
class Person:...

someone = Person("Bob") #1
someone.print_conv()        #2
someone.answer()
```

```
C:\Users\ak111\PycharmProjects\pythonProject\venv\Scripts\python.exe
P1: This person whose name is Bob is speaking something!
P2: But what is he speaking?
P1: The person said something in English that was inaudible

Process finished with exit code 0
```

Jupyter Lab and Notebook

Jupyter Lab and Notebook are free open source web applications that let you create and share documents that contain live code, data visualizations, text and equations. Jupyter fully supports both Python and R, and is particularly useful for interactive scientific programming and visualization.

Anaconda includes Jupyter Notebook (Miniconda se requiere instalar). Once Anaconda is installed, Notebook can be run from Terminal (on Macs) or Command Prompt (on Windows) by typing jupyter notebook. It can also be installed using pip, but installing with Anaconda instead is highly recommended.

Uso en Collaborate from Google https://colab.research.google.com.

Google Collaborate - https://colab.research.google.com

https://youtu.be/rNgswRZ2C1Y

Biopython - http://www.biopython.org

Biopython is an extensive package of Python tools, classes and functions for bioinformatics and computational biology. It was first released in 2000, and now contains over 300 modules for dealing with biological data. The current version, 1.79, was released in June of 2021, and requires Python 3.6 or later. A previous version, 1.76, supports Python 2.7 to 3.5.

In Biopython, sequence data is represented by a Seq class, which includes biological sequence methods such as transcribe or translate, and specifies the sequence alphabet used. The SeqRecord class describes sequences, with features described by SeqFeature objects.

Biopython handles importing and exporting biological data from a wide variety of formats, including Clustal, DNA Strider, FASTA, GenBank, mmCIF, Newick, NEXUS, PDB, PHYLIP and phyloXML using Bio.SeqIO and other modules. The Bio.Entrez module can download and import data directly from various NCBI databases. Phylogeny data can be imported into Tree and Clade objects and traversed and analyzed using the Bio.Phylo module. Molecular structure data can be imported into Structure objects and examined and analyzed using the Bio.PDB module.

Other Biopython features include a GenomeDiagram module for visualizing sequence and genome data, a Bio.PopGen module for interacting with Genepop, support for the BioSQL model and schema, and a number of command line wrappers which allow for Python interaction with commonly used bioinformatics tools such as BLAST, Clustal and EMBOSS.

Basic Biopython

```
> sudo pip install "biopython==1.79"
#(or sudo pip install biopython if running Python 3.6 or later)
python
>>> import Bio
>>> from Bio.Seq import Seq
>>> my_seq = Seq('ATGCATTAG')
>>> print ('Sequence %s is %i bases long' % (my_seq, len(my_seq))
>>> print ('Reverse complement is %s' % my_seq.reverse_complement())
>>> print ('Protein translation is %s' % my_seq.translate())
```

Biopython and Sequences

```
#!/usr/bin/python
from Bio import SeqIO
from Bio.SeqUtils import GC
for sr in SeqIO.parse ("test.fasta", "fasta"):
        print (sr.id)
        print (repr(sr.seq))
        print (len(sr))
        print (sr.seq)
        print GC(sr.seq)
        print (sr.seq.transcribe())
        print (sr.seq.translate())
        print (sr.seq.translate(to_stop=True))
```

Biopython and Parsing

```
#!/usr/bin/python
from Bio import Entrez
Entrez.email = "mi@columbia.edu"
handle = Entrez.efetch(db="nucleotide", rettype="gb", retmode="text", id="2765658")
save_file = open("2765658.gbk", 'w')
save_file.write(handle.read()) handle.close()
save_file.close()

#!/usr/bin/python
from Bio import SeqIO
SeqIO.convert("2765658.gbk", "genbank", "2765658.fasta", "fasta")

#!/usr/bin/python
from Bio import SeqIO
recs = SeqIO.parse("cosmids1.fasta", "fasta")
for rec in recs:
print (rec.id)
```

Biopython BLAST

```
#!/usr/bin/python
from Bio.Blast import NCBIWWW
result_handle = NCBIWWW.qblast("blastn", "nt", "8332116")
from Bio.Blast import NCBIXML
blast_record = NCBIXML.read(result_handle)
E_VALUE_THRESH = 0.04
for alignment in blast_record.alignments:
        for hsp in alignment.hsps:
                if hsp.expect < E_VALUE_THRESH:
                        print('\n****Alignment****')
                        print('*Sequence:', alignment.title)
                        print('*Length:', hsp.align_length)
                        print('*Identities:', hsp.identities)
                        id = (100.00 * hsp.identities / hsp.align_length)
                        print ('*Precent identity:', id)
                        print('*E-value:', hsp.expect)
                        print(hsp.query[0:75] + '...')
                        print(hsp.match[0:75] + '...')
                        print(hsp.sbjct[0:75] + '...')
```

Pandas - https://pandas.pydata.org

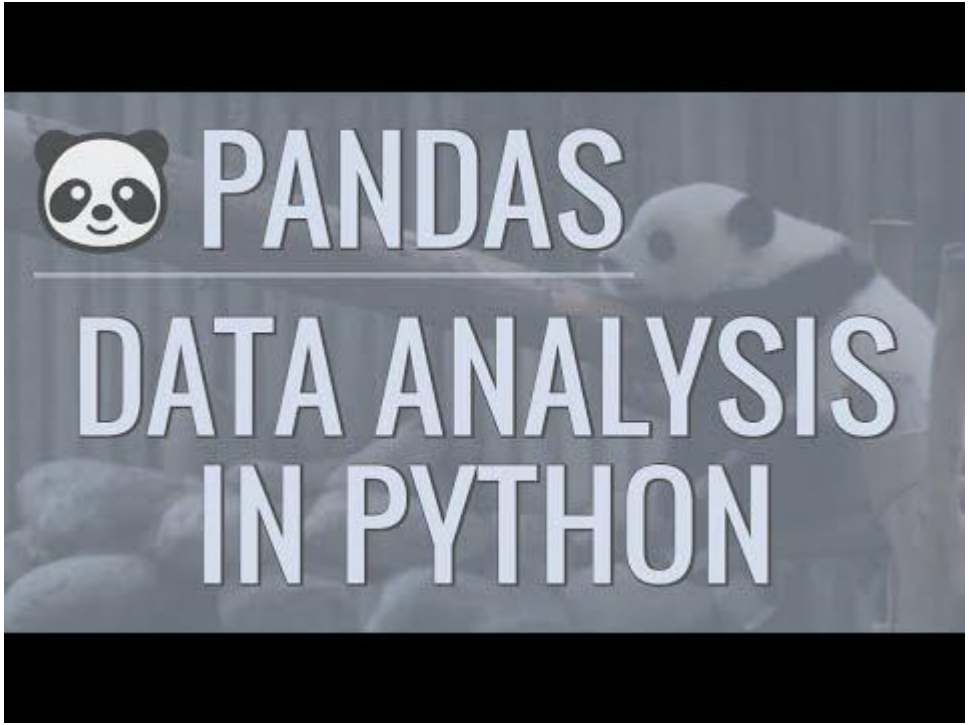
Pandas provides a set of particularly powerful data structures and functions for working with structured data. It is named after panel data, which in statistics and econometrics refers to multidimensional data that frequently changes over multiple time periods.

DataFrames

The primary data structure in pandas is a DataFrame, a two dimensional column oriented structure with row and column labels that can be thought of as a table of data, similar to the R programming language data.frame object. Pandas also supports one dimensional array like structures called a Series, containing an array of data and an associated array of labels.

Pandas allows for data to be loaded into very large DataFrame structures and quickly and efficiently manipulated in a variety of ways: cleaned, transformed, merged, reshaped, pivoted, etc. It also offers high-level plotting functions that supplement those offered by matplotlib, and simplifies the visualization of large, complex data sets.

PANDAS - https://pandas.pydata.org



https://youtu.be/ZyhVh-qRZPA

From min 7.45

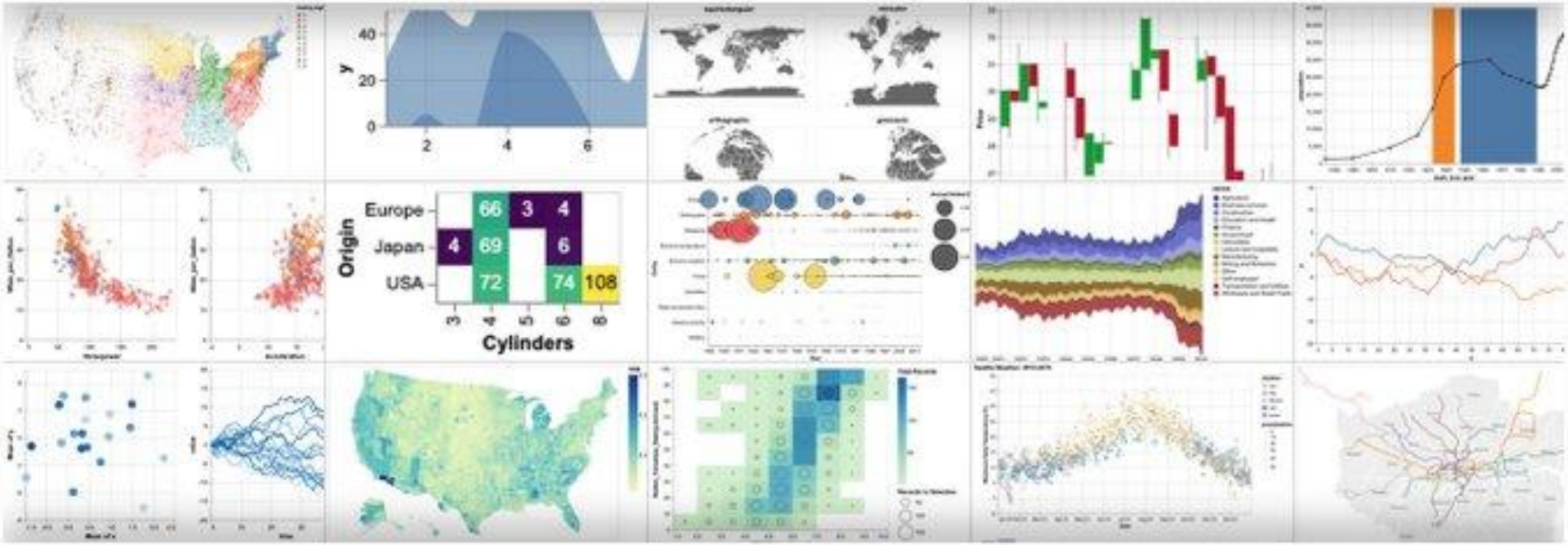
Pandas en 10 minutos

https://pandas.pydata.org/docs/user_guide/10min.html

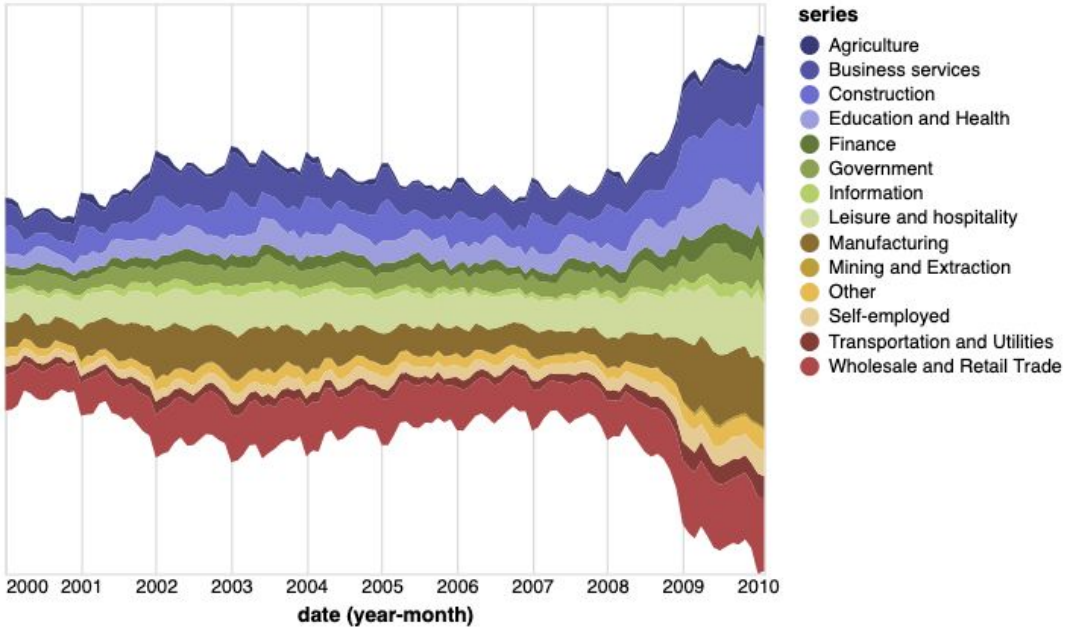
Altair https://altair-viz.github.io

```
import altair as alt
from vega_datasets import data

source = data.unemployment_across_industries.url

selection = alt.selection_multi(fields=['series'], bind='legend')

alt.Chart(source).mark_area().encode(
    alt.X('yearmonth(date):T', axis=alt.Axis(domain=False, format='%Y', tickSize=0)),
    alt.Y('sum(count):Q', stack='center', axis=None),
    alt.Color('series:N', scale=alt.Scale(scheme='category20b')),
    opacity=alt.condition(selection, alt.value(1), alt.value(0.2))
).add_selection(
    selection
)
```

Altair Gallery https://altair-viz.github.io/gallery/index.html



https://altair-viz.github.io/gallery/interactive_legend.html#gallery-interactive-legend



Extras

Metodología ágil https://www.redhat.com/es/devops/what-is-agile-methodology

Microservicio https://www.redhat.com/es/topics/microservices

Manejo de subversiones GIT https://git-scm.com

StackOverFlow https://stackoverflow.com

Biostarts https://www.biostars.org

References

Practical Computing for Biologists free at:
http://people.duke.edu/~ccc14/pcf/index.html

Biopython Tutorial and Cookbook free at:
http://biopython.org/DIST/docs/tutorial/Tutorial.html

Biopython Documentation free at: https://biopython.org/wiki/Documentation
Introduction to Computation and Programming Using Python by John V. Guttag

Python for Data Analysis: Data Wrangling with Pandas, NumPy and iPython by Wes McKinney