EXAMEN PARCIAL PYTHON

GBI6-2021II: BIOINFORMÁTICA

Apellidos, Nombres <--- CAMBIE POR LOS QUE CORRESPONDA A SUS DATOS

03-08-2022 Daniela Chávez

Color de texto

REQUERIMIENTOS PARA EL EXAMEN

Utilice de preferencia `Jupyter` de Anaconda, dado que tienen que hacer un control de cambios en cada pregunta.

Para este examen se requiere dos documentos:

1. Archivo `miningscience.py` donde tendrá dos funciones:
2. Archivo `2022I_GBI6_ExamenPython` donde se llamará las funciones y se obtendrá resultados.

Ejercicio 0 [0.5 puntos]

```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sb
! pip install biopython
```

Realice cambios al cuaderno de jupyter:

- Agregue el logo de la Universidad
- Coloque sus datos personales
- Escriba una **tabla** con las características de su computador

```
my-compt = pd.DataFrame({'Nombre del dispositivo': pd.Categorical(['LAPTOP-B6VEURON']),
                         'Procesador': pd.Categorical(['AMD Ryzen 5 3500U with Radeon']),
                         'RAM instaloda': [8,0],
                         'Sistema Operativo': ['64 bits']})
```

Ejercicio 1 [2 puntos]

my-compt

Cree el archivo `miningscience.py` con las siguientes dos funciones:

i. `download_pubmed` : para descargar la data de PubMed utilizando el **ENTREZ** de Biopython. El parámetro de entrada para la función es el `keyword` .

ii. `science_plots` : la función debe

- utilizar como argumento de entrada la data descargada por `download_pubmed`
- ordenar los conteos de autores por país en orden ascedente y
- seleccionar los cinco más abundantes. Con esta selección debe graficar un `pie_plot` . Como guía para el conteo por países puede usar el ejemplo de MapOfScience (https://github.com/CSB-book/CSB/blob/master/regex/solutions/MapOfScience_solution.ipynb).

iii *Cree un docstring para cada función.*

Luego de crear las funciones, cargue el módulo `miningscience` como `msc` e **imprima docstring de cada función.**

In [1]:

```
# Escriba aquí su código para el ejercicio 1
#Importando la librería
import miningscience as msc
#Docstrings funciones
help(msc.dowload_pubmed)
help(msc.science_plots)
```

Ejercicio 2 [2 puntos]

Utilice dos veces la función `download_pubmed` para:

- Descargar la data, utilizando los keyword de su preferencia.
- Guardar el archivo descargado en la carpeta `data` .

Para cada corrida, imprima lo siguiente:

`'El número artículos para KEYWORD es: XX'` *# Que se cargue con inserción de texto o valor que correspondea KEYWORD y XX*

In [2]:

```
# Escriba aquí su código para el ejercicio 2
# Primera data
import os
import re
w=msc.dowload_pubmed("monkey pox")
x=len(w)
print("El número de artículos para MONKEYPOXES:", x)
with open("Data/monkeypox.txt",w) as txt:
    txt.write(w)
```

```
#Segunda data
import os
import re
y = msc.dowload_pubmed("covid")
z= len(y)
print("El número de artículos para COVID es:",z)
with open("Data/covid.txt",w) as txt.
  txt.write(y)
```

Ejercicio 3 [1.5 puntos]

Utilice dos veces la función `science_plots` para:

- Visualizar un pie_plot para cada data descargada en el ejercicio 2.
- Guardar los pie_plot en la carpeta `img`

In [4]:

```
# Escriba aquí su código para el ejercicio 3
# Gráfica primera data
monkeypox = msc.science_plots(w)
with open ("img/monkeypox.jpg","w") as jpg
    monkeypox
```

```
#Gráfica segunda data
covid = msc.science_plots(y)
with open ("img/covid.jpg","w") as jpg
    covid
```

Ejercicio 4 [1 punto]

Interprete los resultados de las figuras del **ejercicio 3**

Escriba la respuesta del ejercicio 5 .

Figura 1

Ultimamente un tema que ha mostrado relevancia es la viruela del mono. los países que poseen mayor número de publicaciones al respecto son: España, Italia, Estados Unidos, Reino Unido y Germania

Figura 2

los países que poseen mayor número de artículos publicados en relación al covid son: Estados Unidos, Germania, India, Italia y China.

Ejercicio 5 [2 puntos]

Para algún **gen de las enzimas que intervienen en la ruta metabolica de la gluconeogenesis** (Lista de genes por tipología (https://www.genome.jp/pathway/map00010+C00068)), realice lo siguiente:

1. Una búsqueda en la página del NCBI nucleotide (https://www.ncbi.nlm.nih.gov/nucleotide/).
2. Descargue el `Accession List` de su búsqueda y guarde en la carpeta `data` .
3. Cargue el `Accession List` en este notebook y haga una descarga de las secuencias de los **quince primeros** IDs de la accesión.
4. Arme un árbol filogenético para los resultados del paso 3.
5. Guarde su arbol filogénetico en la carpeta `img`
6. Interprete el árbol del paso 4.

Pregunta 5

```
from Bio import Entrez
from Bio import Phylo
import matplotlib.pyplot as plt
from Bio.Phylo.TreeConstruction import DistanceTreeConstruction
from Bio import SeqIO
from Bio.Phylo.TreeConstruction import DistanceCalculator
from Bio.Align.Applications import ClustalwCommandline
from Bio import AlignIO
import os

with open("Data/sequence.seq") as file:
    text = file.read()
text = text.split('\n')
text = ','.join(text[:15])
handle = Entrez.efetch(db="nucleotide", rettype="gb", retmode="text", id=text)
print(handle.url)
records = SeqIO.parse("Data/sequence.gb", "genbank")
count = SeqIO.write(records, "Data/sequence.fasta", "fasta")
clustalw_exe = r"c:\Program Files (x86)\Clustalw2\clustalw2.exe"
clustalw_cline = ClustalwCommandline(clustalw_exe, infile="Data/sequence.fasta")
assert os.path.isfile(clustalw_exe), "Clustal W executable is missing or not found"
stdout, stderr = clustalw_cline()
print(clustalw_cline)
ClustalAlign = AlignIO.read("Data/sequence.aln", "clustal")
with open('Data/sequence.aln', "r") as aln:
    alignment = AlignIO.read(aln, "clustal")
calculator = DistanceCalculator("identity")
distance_matrix = calculator.get_distance(alignment)
constructor = DistanceTreeConstructor(calculator)
tree = constructor.build_tree(alignment)
tree.rooted = True
fig = plt.figure(figsize=(10, 12), dpi=200)
plt.rc('font', size=12)
plt.rc('xtick', labelsize=20)
plt.rc('ytick', labelsize=20)
axes = fig.add_subplot(1, 1, 1)
Phylo.draw(tree, axes=axes)
fig.savefig("../img/arbol.jpg")
```

In [3]:

```
# Escriba aquí su código para el ejercicio 6
```

Escriba aquí la interpretación del árbol

Ejercicio 6 [1 punto]

1. Cree en GitHub un repositorio de nombre `GBI6_ExamenPython` .
2. Cree un archivo `Readme.md` que debe tener lo siguiente:

- Datos personales
- Características del computador
- Versión de Python/Anaconda y de cada uno de los módulos/paquetes y utilizados
- Explicación de la data utilizada
- Un diagrama de procesos del módulo `miningscience`

3. Asegurarse que su repositorio tiene las carpetas `data` e `img` con los archivos que ha ido guardando en las preguntas anteriores.
4. Realice al menos 1 control de la versión (commits) por cada ejercicio (del 1 al 5), con un mensaje que inicie como:

```
Carlitos Alimaña ha realizado el ejercicio 1

Carlitos Alimaña ha realizado el ejercicio 2

...
```

In [ ]:

```

```

GBI6 – BIOINFORMÁTICA [2022I]
Examen Final [Python]

Ikiam
Universidad Regional Amazónica

Nombre [Apellido, Nombre]:

Construya las funciones del módulo miningscience.py

```
from Bio import Entrez
from Bio import SeqIO
from Bio import Genbank
import re
import pandas as pd
import matplotlib as plt
import csv as csv
```

```
def download_pubmed( Keyword ):
    """
```

Permite buscar articulos cientificos en Pubmed con filtrado mediante el uso de palabras clave (keywords)

```
    """
    Entrez.email = "A.N.Other@example.com"
    busq = Entrez.read (Entrez.esearch (db = "pubmed")
                        term = Keyword,
                        usehistory = "y"))
    webenv = busq["Webenv"]
    query-key = busq["Query Key"]
    handle = Entrez.efetch (db = "pubmed",
                            rettype = "medline",
                            retmode = "text",
                            retstart = 0,
                            retmax = 543, webenv = webenv, query-key = query-key
    data-pubmed = handle.read()
    data = re.sub (r'\n\s{6}', '', data-pubmed)
    return data
```

Nombre [Apellido, Nombre]:

```
def science_plots( archivo                                    ):
    """
    Permite generar un diagrama de pastel de los países
    a los que pertenecen los autores que han realizado
    publicaciones sobre un tema en particular.
    En la gráfica se muestran los 5 países con mayor
    número de publicaciones.
    """
    Correos = re.sub(r'\s[\w._%+-]+@[\w.-]+\.[a-zA-Z]{1,4}',"",archivo)
    puntos = re.sub(r'\..\d.\',',',',correos)
    numb = re.sub(r'\..\d.',"",puntos)
    x = numb[1:].split('PMID-')

    Countries_A = []
    for PMID in x:
        q = PMID.split('\n')
        for fila in q:
            w=fila.split(' ')
            if w[0] == 'AD':
                e = fila.split(',')
                Countries_A.append(e[-1])

    a=0
    Countries_B = [0]*len(Countries_A)
    for lis in Countries_A:
        bytes(lis,encoding="utf8")
        if lis != '':
            w = lis
            if w[0] == ' ':
                w = re.sub(r'^\s',"",w)
            if w[-1] == '.':
                w = re.sub(r'\.$',"",w)
            w = re.sub(r'\.$',"",w)
            w = re.sub(r'\s$',"",w)
            Countries_B[a] = w
            a = a+1
```

```
Countries_all=['Andorra','United Arab Emirates','Afghanistan','Antigua and Barbuda',
'Anguilla','Albania','Armenia','Netherlands Antilles','Angola','Antarctica','Argentina','American
Samoa','Austria','Australia','Aruba','Azerbaijan','Bosnia and Herzegovina','Barbados',
'Bangladesh','Belgium','Burkina Faso','Bulgaria','Bahrain','Burundi','Benin','Bermuda',
'Brunei','Bolivia','Brazil','Bahamas','Bhutan','Bouvet Island','Botswana','Belarus','Belize',
'Canada','Cocos [Keeling] Islands','Congo [DRC]','Central African Republic',------,'Zambia',
'Zimbabwe']

    Countries_C=Countries_B
    h=Contries_all
    f=len(h)
    CountriesCount=[0]*f
    K=0
    for elem in h
        d=0
        for comp in Countries_C:
            if elem==str(comp):
                d=d+1
        CountriesCount[K]=d
        K=K+1
    Countries_D=[]
    Counter=[]
    o=0
    for line in CountriesCount:
        if str(line)!='0':
            Counter.append(line)
            m=Countries_all[o]
            Countries_D.append(m)
        o=o+1
    Table_A=pd.DataFrame({'Country':Countries_D,
                          'num_auth':Counter})
    Order=Table_A.sort_values(by=['num_auth'],ascending=[False])
    Taken=Order.iloc[0:5]
    suma=Taken['num_auth'].sum()
    iu=Taken.iloc[:,0]
    su=pd.series(iu)
    i=Taken.iloc[:,1]
    sa=pd.Series(i)
    s=sa.tolist()
    prom=[]
    for number in s:
        xa=(number/suma)*100
        prom.append(xa)
Table_B=pd.DataFrame({'Country':su,'Porcent':prom})
fig1,ax1=plt.subplots()
ax1.pie(prom,labels=su,autopct='%1.1f%%',shadow=True,startangle=90
ax1.axis('equal')
plt.show()
plt.savefig('img/Graficadepie.jpg',dpi=500)
return(Table_B)
```