EXAMEN PARCIAL PYTHON

GBI6-2021II: BIOINFORMÁTICA

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

03-08-2022 Iturralde Bavahona Lina Arabell

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]

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

Ejercicio 1 [2 puntos]

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

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 ca función.**

In [1]:

```
# Escriba aquí su código para el ejercicio 1
# Lina Iturralde
# lista de librerías para el examen
from Bio import Entrez
  import re
  import csv
  import intertools.
# Se carga el modulo minigsciencia como msc
  import miningsciencia_g01 as msc.
# Se imprime el docstring de cada función
 help(msc.download_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
 import re
#se llama a las funciones
data_2 = msc.download_pubmed("Mycobacterium [title/Abstract]")
contar = len(data_2)
print("El número de artículos para Mycobacterium [Title/Abstract] es:", contar)
 with open("data/Mycobacterium.txt","w") as txt:
      txt.write(data_2)
data_3 = msc.download_pubmed("pulmonary.fibrosis [Title/Abstract]")
contar = len(data_3)
print("El número de artículos para la pulmonary fibrosis [Title/Abstract] es:"
    contar)
 with open("data/pulmonary fibrosis.txt","w") as txt:
      txt.write(data_3)
```

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`

[ ]:

Escriba aquí su código para el ejercicio 3

```
# Mycobacterium
print("\n\tDataFrames de los datos de los países y cantidades\n\t")
df_paT = msc_science_plots(data_2)
df_pa_T1 = df_pa_T.sort_values(by=['Número de autores'],
ascending=False)
df_pa_T2 = df_pa_T1.iloc[0:5]
df_pa_T2
import matplotlib.pyplot as plt
labels = 'China', 'USA', 'India', 'South Africa', 'Brazil'
sizes = [1987, 1706, 1246, 579, 521]
explode = (0, 0, 0, 0, 0)
fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%.1.1f%%',
shadow=True, startangle=0)
ax1.axis('equal')
plt.savefig("img/autores Mycobacterium.jpg")
```

Ejercicio 4 [1 punto]

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

Escriba la respuesta del ejercicio 5.

Interpretación Mycobacterium

- Se identifica que el país con mayor número de autores es liderado por China con un porcentaje de 32,9%, continua USA con 28,2% por último el país con menor autores es Brazil, ya que tiene un porcentaje de 8,6%.

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.

→ Continuación Ej 3.

```
#pulmonary fibrosis
print("\n\nIt DataFrames de los datos de los países y cantidades\n\t")
df_pa_T = msc.science_plots(data_3)
df_pa_T3 = df_pa_T.sort_values(by=['Numero de autores'],
    ascending=False)
df_pa_T4 = df_pa_T3.iloc[0:5]
df_pa_T4
```

```
import matplotlib.pyplot as plt
labels = 'China', 'USA', 'Japan', 'Italy', 'Germany'
sizes = [4458, 1810, 1152, 801, 678]
explode = (0, 0, 0, 0, 0)
fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct=
'%1.1f%%', shadow=True, startangle=10)
ax1.axis('equal')
plt.savefig("img/autorespulmonary_fibrosis.jpg")
```

- Interpretación Pulmonary Fibrosis.

Luego de realizar un pie plot con los datos obtenidos podemos decir que el país con mayor número de autores de "Pulmonary Fibrosis" es China con 4458, seguido por USA con aproximadamente 1810 autores, luego Japón continua con 1152 autores, mientras que en los últimos lugares se encuentra Italy con 801 y Alemania con 678 autores, es decir que tienen menor número de afluencia de autores.

In [3]:

```
# Escriba aquí su código para el ejercicio 6
from Bio.Phylo.TreeConstruction import DistanceTreeConstructor
from Bio.Phylo.TreeConstruction import DistanceCalculator
from Bio.Align.Applications import ClustalwCommandline
from Bio import AlignIO
from Bio import Phylo
from Bio import Entrez
from Bio import SeqIO
import Bio
import warnings
import os
```

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 [ ]:

```
import matplotlib
import matplotlib.pyplot as plt
# Primero se descarga el archivo Accession list de PubMed y se luego sus datos
# Cargue el Accession list en este notebook
with open ("data/phosphoglycerate_kinase.seq", "r") as f:
    seqPGK = f.read()
    seqPGKlist = seqPGK.split('\n')
seqPGKlistT = []
cont = 0
for i in seqPGKlist:
    if cont < 15:
        seqPGKlist.append(i[:])
        cont+=1
Entrez.email = "gualapuro.moises@gmail.com"
ofile = open ("data/phosphoglycerate_kinase.gb", 'w')
with Entrez.efetch (db="nucleotide", rettype="gb", retmode="text",
    id = seqPGKlist) as handle:
    for seq_record in SeqIO.parse(handle, "gb"):
    ofile.write (">"+ str (seq_record.id) + str (seq_record.description
    [:50])+'\n')
    ofile.write (str(seq_record.seq) + '\n')
    ofile.write ('\n')
#Archivos tipo FASTA y GB.
ffile = open ("data/phosphoglycerate_kinase.fasta", 'w')
    with open ("data/phosphoglycerate_kinase.gb", 'w') as genbank:
        c = genbank.read()
        for line in c:
            ffile.write (str (line))
crustalw_exe = r"C:\Program Files (x86)\Clustalw2\clustalw2.exe"
clustalw_cline = ClustalwCommandline (clustalw_exe, infile =
"data/phosphoglycerate_kinase.fasta")
assert os.path.isfile(clustalw_exe), "Clustal_wexecutable is
missing or not found"
stout, stderr = clustalw_cline()
ClustalAlign = AlignIO.read ("data/phosphoglycerate_kinase.aln",
"clustal")
```

```
calculator = DistanceCalculator ("identity")
distance_matrix = calculator.get_distance (ClustalAlign)
constructor = DistanceTreeConstructor (calculator)
Data_treePGK = constructor. build_tree (ClustalAlign)
Data_treePGK. rooted = true
Phylo.write (Data_treePGK, "data/treePGK .xlx", phyloxml")
# Árbol Filogenètico
fig = plt. figure ( figsize = (20, 25) , dpi = 200)
matplotlib.rc ('font', size = 12)
matplotlib.rc ('xtick', labelsize = 10)
matplotlib.rc ('ytick', labelsize = 10)
axes = fig. add_subplot (1,1,1)
Phylo. draw (Data_treePGK, axes = axes)
print ("\t\t\t\t\t\t\t Fig3. Arbol Filogenético
Phosphoglycerate_kinase\n\n")
```

No hay interpretación del árbol porque no se ejecuta por completo ¨̂ , se hizo lo que se pudo profe ¨̂ .

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

Ikiam
Universidad Regional Amazónica

Nombre [Apellido, Nombre]:

Construya las funciones del módulo miningscience.py

```
def download_pubmed(keyword)
    """                                                    ):
    Con la función Entrez me permite hacer directamente la búsqueda en
    la base de datos de PubMed y con las keyword que
    se emplearán en los ejercicios:
        Mycobacterium - Palmonary Fibrosis.

    """
Entrez.email = "guabapuro.moises@gmail.com"
handle = Entrez.research (db= "pubmed",
                          term= keyword,
record = Entrez.read (handle)   usehistory="y")
id-list = record ["IdList"]
webenv = record ["WebEnv"]
query-key = record ["QueryKey"]
handle = Entrez.efetch (db="pubmed",
                        rettype = "medline",
                        retmode = "text",
                        retstart = 0,
                        retmax = 1500,
                        webenv = webenv,
data = handle.read ()   query_key = query_key)
data_2 = re.sub (r'\n\s}6{', '', data)
return (data_2)
```

Nombre [Apellido, Nombre]:

```
def science_plots( data_2                    ):
    """ Mediante esta función vamos a buscar y los autores de la base PubMed
    con el fin de que contabilice el número de autores por país y los
    que obtienen con mayor repetición, ya que la variable
    país será pa, es así como mediante expresiones regulares se delimitará
    y se obtendrá las repeticiones de los autores.


    """
    AD = [ ]
    pa1 = [ ]
    pa2 = [ ]
    pa3 = [ ]
    pa4 = [ ]
    pa5 = [ ]
    pa6 = [ ]
    pa7 = [ ]
    pa8 = [ ]
    pa9 = [ ]
    pa10 = [ ]
f   for line in data_2.splitlines( ):
        if line.startswith ("AD -"):
            AD.append(line [:])
    for line in data_2.splitlines( ):
        if line. startswith ("AD -"):
            AD = line [:]
            p1 = re. findall (r' \,\s(\w}2,16{)\.', AD)
            pa1.append (p1)
            p2 = re.findall(r'\,\s(\w}2,16{[^0-9\,]\s\w}2,16{[^0-9]})\.', AD)
            pa2.append (p2)
            p3 = re.findall (r' \,\s(\w}3,16{[^0-9\,]\s\w}2,3{[^0-9\,]
            \s\w}3,16{[^0-9\,])\.', AD)
            pa3. append (p3)
            p4 = re.findall (r'\,\s(\w}2,16{)\.\s[a-z0-9_\.-]+
            @ [\da-z\.-]+\.[a-z\.]}2,6{', AD)
            pa4.append (p4)
            p5 = re.findall (r'\,\s(\w}2,16{[^0-9\,]s\w}2,16{
            [^0-9])\.\s[a-z0-9_\.-]+@[\da-z\.-]
            +\.[a-z\.]}2,6{', AD)
            p6 = re.findall (r'\,\s(\w}3,16{[^0-9\,]
            \s\w}2,3{[^0-9\,]\s\w}3,16{[^0-9\,])
```

```
\.\s[a-z0-9_\.-]+@[\da-z\.-]+\.[a-z\.]
}2,6}', AD)
    pa6.append(p6)
    p7 = re.findall(r'\,\s(\w}2,16}).Electronic address: \s
    [a-z0-9_\.-]+@[\da-z\.-]+\.[a-z\.]}2,6}\
    .', AD)
    pa7.append(p7)
    p8 = re.findall(r'\,\s(\w}2,16}[^0-9\,]\s\w}2,16}[^0-9])
    \. Electronic address: \s[a-z0-9_\.-]+@[\da-z\.-]
    +\.[a-z\.]}2,6}\.', AD)
    pa8.append(p8)
    p9 = re.findall(r'\,\s(\w}3,16}[^0-9\,]\s\w}2,3}
    [^0-9\,]\s\w}3,16}[^0-9\,])\. Electronic address: \s
    [a-z0-9_\.-]+@[\da-z\.-]+\.[a-z\.]}2,6}\.', AD)
    pa9.append(p9)
    p10 = re.findall(r'\,\s\w}3,9}[0-9\-]\,\s(\w}2,16})\.', AD)
    pa10.append(p10)
    pa_T = pa1 + pa2 + pa3 + pa4 + pa5 + pa6 + pa7 + pa8 +
           pa9 + pa10.
    pa_T = list(intertools.chain.from_iterable(pa_T))
    len(pa_T)
    unique_pa_T = list(set(pa_T))
    unique_pa_T.sort()
    len(unique_pa_T)
    import csv
    coordenadas = {}
    with open('data/ubi pais.txt') as f:
    cs      csvr = csv.DictReader(f)
            for row in csvr:
            coordenadas[row['Name']] = [row['Latitude'],
            row['Longitude']]
    country = []
    longitude = []
    latitude = []
    almacen = []
    for z in unique_pa_T
        if z in coordenadas.keys():
            country.appena(z)
            latitude.append(float(coordenadas[z][0]))
            longitude.append(pa_T.count(z))
            almacen.append(pa_T.count(z))
```

```
df_pa_T = pd.DataFrame()
df_pa_T ["Pais"] = country
df_pa_T ["Numero de autores"] = almacen
return (df_pa_T)
```