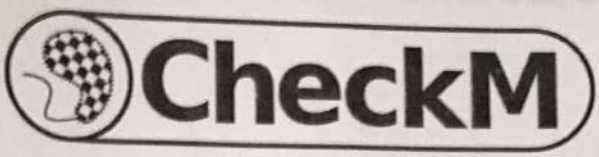

EXAMEN PARCIAL PYTHON

GBI6-2021II: BIOINFORMÁTICA

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

03-08-2022

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:

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).

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
import miningscience_g01 as msc
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
p1 = "bacteria"
busquedaA = msc.download_pubmed(p1)
print('El número articulos para', p1, 'es: ', len(busquedaA))
with open("Data/Bacteria.txt", "w") as txt:
    txt.write(busquedaA)
```

R// El número de articulos para bacteria es: 2384168

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`

Gen 1:

Escriba aquí su código para el ejercicio 3

```
gy =msc.science_plots (busqueda A)
gy =
```

Ejercicio 4 [1 punto]

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

En el pie plot se puede observar los cinco países más abundantes en publicaciones en pubmed de acuerdo a la palabra clave "Bacteria", el primer lugar con 60.99% se encuentra China, con 13.43% esta India, 8.85% Francia 8,44 % Canada y con un 8.26% Japon.

Escriba la respuesta del ejercicio 5

Con este resultado se puede decir que China está en primer lugar, en reportes científicos, en busca de respuesta sobre incertidumbres de la ciencia.

Pregunta 5

Ejercicio 5 [2 puntos]

guardar gy: glucosidasa. fact → se cay

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.

Ejercicio 5

Gen escrita elegida: G lactate dehydrogenase.

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

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)

ClistalAlign = 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)
rag2_tree = constructor.build_tree(alignment)
Phylo.draw(rag2_tree)

fig = plt.figure(figsize=(50,60), dpi=200)
matplotlib.rc('font', size=12)
matplotlib.rc('xtick', labelsize=20)
matplotlib.rc('ytick', labelsize=20)
axes = fig.add_subplot(1,1,1)
Phylo.draw(rag2_tree, axes=axes)
fig.savefig("img/rag2_tree")
```

from Bio
from
from
from
from
fr

PGBI6 – BIOINFORMÁTICA [2022I]
Examen Final [Python]

Ikiam
Universidad Regional Amazónica

Nombre [Apellido, Nombre]:

Construya las funciones del módulo miningscience.py

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

La función download_pubmed permite como ingreso una palabra de búsqueda "palabras clave" de artículos en pubmed

```
    """
    Entrez.email = 'A.N.Other@example.com'
    term = Entrez.email(Entrez.esearch(db="pubmed",
                                       term=Keyword,
                                       usehistory="y"))

    webenv = term["WebEnv"]
    query_key = term["QueryKey"]
    handle = Entrez.efetch(db="pubmed",
                           rettype="medline",
                           retmode="text",
                           retstart=0,
                           retmax=548, webenv=webenv, query_key=query_key)

    export = handle.read()
    imprdata = re.sub(r'\n\s{6}', '', export)
    return imprdata
```

Ikiam
Universidad Regional Amazónica

Nombre [Apellido, Nombre]:

```
def science_plots( archivo
    """                                    ):
```

la función science_plots permite ordenar los conteos de autores por país en orden ascendente y seleccionar los cinco más abundantes en función de una palabra de búsqueda "palabras clave" de artículos de pubmed.

```
    """
elimcorreos = re.sub(r'\S[\w.-]+@[\w.-]+\.[a-zA-Z]{1,4}',' ',archivo)
elimpuntos = re.sub(r'\..\d.\',' .',elimcorreos)
elimnumb = re.sub(r'\..\d.',' ', elimpuntos)
x = elimnumb[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(',')
            CountriesA.append(e[-1])


a=0
Countries B = [0] * leng(CountriesA).
for lis in CountriesA:
    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    #Saca las últimas palabras
        a = a+1
```


INVESTIGACIÓN · INNOVACIÓN · EDUCACIÓN | Parroquia Muyuna, kilómetro 7 vía a Alto Tena
Tena · Napo · Ecuador
Telf.: (06) 370 0040 · (06) 299 9160

Template = [aquí agrege en forma de diccionario, todos los países del mundo]

```
Countries C = Countries B
h= Template
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
  Countries Count[k] = d    #contador.
  k= k+1

Countries D = [ ]
Counter = [ ]
σ = 0
for line in Countries Count:
   if str(line) != '0'
      Counter.append(line)
      m= Template [σ]
      Countries D. append(m)
   σ =σ +1

Table A = pd. Data.Frame({'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)
a = sa.tolist ( )

prom = [ ]
for number in a :
     xa = (number /suma)* 100
     prom.append (xa)
TableB = pd. Data Frame ({'Country' : su,
                          'num_auth : s,
                          'Porcent' : prom})

fig1, ax1 = plt.subplots ( )
ax1.pie(prom, labels=su, autopct = '%1.1f%%',
        shadow = True , startangle = 90)
ax1.axis ('equal')
```

Aquí continúa el codigo.

```
plt.title ('grafica de pie')
plt.show( )
plt.save fig ('img/Graficadepie.... jpg',
              d pi = 300)

return (Table B)
```