EXAMEN PARCIAL PYTHON

GBI6-2021II: BIOINFORMÁTICA

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

03-08-2022 Slendy Alvarado

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 e 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 [illegible] función.

In [1]:

```
# Escriba aquí su código para el ejercicio 1
import miningscience as msc
help(download_pubmed)
help(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
busqueda = download_pubmed("peptidology")
busqueda_1 = (download_pubmed("R Ecuador"))
print("El número artículos para peptidology:", len(busqueda))
print("El número artículos para R Ecuador: ", len(busqueda_1))
```

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
```

```
science_plots("Peptidology")
science_plots("R Ecuador")
```

Ejercicio 4 [1 punto]

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

Escriba la respuesta del ejercicio 5 . En la p

. Busque1 = "Peptidology" se encontraron 22 artículos de la base de datos pubmed, mayor cantidad de autores de china y el segundo Mexico y finalmente españa e India.

. Busqueda 2 "R Ecuador" se encontro 321 artículos.

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.

Se obtiene las 15 secuencias de NCBI

```
from Bio import Phylo
from Bio import SeqIO
from Bio import AlignIO
from Bio.Phylo.TreeConstruction import DistanceCalculator
from Bio.Phylo.TreeConstruction import DistanceTreeConstructor
from Bio import Entrez
import re
import os
from Bio.Align.Applications import ClustalwCommandline
```

```
with open ("sequence.seq") as f:
    data = f.readlines () [0:15]
out_secuence = open ("secuencias.fasta", "w")
for linea in data:
    Entrez.email = "slendy.alvarado@est.ikiam.edu.ec"
    handle = Entrez.efetch (db="nucleotide", id=linea, rettype="fasta"
                                                    , retmode="text")
    data = (handle.read())
    out_sequence.write(data)
out_sequence.close()
```

Creamos alineamiento

```
clustalw_exe = r"C:\Program Files (x86)\clustalw2\clustalw2.exe"
clustalw_cline = clustalwCommandline (Clustalw_exe, infile = "secuencias
assert os.path.isfile (Clustalw_exe), "Clustal_w executable is missing or not
                                                    found"
stdout, stderr = clustalw_cline()
print (clustalw_cline)
clustalAlign = AlignIO.read ("secuencias.aln", "clustal")

from Bio import Phylo
tree = phylo.read ("secuencias.dnd", "newick")
```

Se crea el arbol filogenetico

```
with open ("secuencias.aln", "r") as aln:
    alignment = AlignIO.read (aln, "clustal")
from Bio.Phylo.TreeConstruction import DistanceCalculator
calculator = DistanceCalculator ('identity')
distance_matrix = calculator.get_distance (alignment)
from Bio.Phylo.TreeConstruction import DistanceTreeConstructor
constructor = DistanceTreeConstructor (Calculator)
oxa = constructor.build_tree (alignment)
oxa.rooted = True
phylo.write (oxa, "tree.xml", "phyloxml")
tdna_tree = phylo.read (file="tree.xml", format="phyloxml")
# arbol elemental en Matplotlib
import matplotlib
import matplotlib.pyplot.as plt
fig = plt.figure (figsize = (30,40), dpi = 100)
matplotlib.rc ('font', size=30)
matplotlib.rc ('xtick', labelsize = 20)
matplotlib.rc ('ytick', labelsize = 20)
axes = fig.add_subplot (1, 1, 1)
phylo.draw (oxa, axes = axes)
fig.savefig ("img/oxa.jpg")
```

Nombre [Apellido, Nombre]: Slendy Alvarado

Construya las funciones del módulo miningscience.py

```
def download_pubmed( keyword ):
    """
```

Función que de entrada pide al usuario la keyword tipo str y en output guarda un archivo que contiene los resultados de la busqueda en base a los titulos/resumen.

```
    """
    Entrez.email = "slendy.alvarado@est.iktam.edu.ec"
    handle = Entrez.esearch(db="pubmed",
                            term = keyword+"[Title/abstract]",
                            retmax = 1000,
                            usehistory = "y")
    record = Entrez.read(handle)
    id_list = record["IdList"]
    webenv = record["WebEnv"]
    query_key = record["QueryKey"]
    handle = Entrez.efetch(db="pubmed",
                           rettype="medline",
                           retmode="text",
                           webenv=webenv,
                           query_key=query_key)
    out_handle = open("data/"+keyword, "w")
    data = handle.read()
    print(id_list)
    handle.close()
    out_handle.write(data)
    out_handle.close()
    return id_list
```

Nombre [Apellido, Nombre]:

```
def science_plots(    data.                    ):
    """
    función que pide como entrada la data de la
    función download_pubmeds y como resultado un gráfico
    tipopastel que indican a los cinco países que
    aparecieron más veces.

    """
    with open("data/"+data, errors="ignore") as f:
        texto = f.read()
    texto = re.sub(r"\n\s{6}", " ", texto)
    pais = re.findall(r"AD\s{2}-\s[A-Za-z].*, \s([A-Za
                        -z]*)\.\s", texto)
    conteo = Counter(pais)
    resultado = {}
    for clave in conteo:
        valor = conteo[clave]
        if valor >1:
            resultado[clave] = valor
    ordenar = (sorted(resultado.values()))  ## ordena de forma ascende
    ordenar.sort(reverse = True)
    import operator
    countries = []
    counter = []
    reverse = sorted(resultado.items(), key= operator.itemgetter(1), reverse
                                                                  =True)
    for name in enumerate(reverse):
        countries.append(name[1][0])
        counter.append(resultado[name[1][0]])
    mas_pais = countries[0:5]  ## Selecciona los 5 primeros países
    mas_frec = counter[0:5] ## frecuencias respecto a los 5 países
    fig = plt.figure(figize=(10,7))
    plt.pie(mas_frec, labels= mas_pais)
    (plt.savefig("img/"+data, dpi=120, bbox_inches='tight'))
    plt.show()
```