EXAMEN PARCIAL PYTHON

GBI6-2021II: BIOINFORMÁTICA

Apellidos, Nombres <--- CAMBIE POR LOS QUE CORRESPONDA A SUS DATOS
Calva Moreno Ibeth Calva
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).

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
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
journal = dowload_pubmed("Ecuador SARS-CoV-2")
journal1= (download_pubmed("Ecuador shrimp"))
print("El número artículos para Ecuador SARS-CoV-2 es:", len(jaurnal))
print("El número artículos para Ecuador shrimp:", len(journal1))
```

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

    science_plots("Ecuador SARS-CoV-2")
    science_plots("Ecuador shrimp")
```

Ejercicio 4 [1 punto]

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

Escriba la respuesta del ejercicio 5 .

- En la primera busqueda se obtuvieron 126 articulos de SARS-CoV2, es un número alto dado el descubrimiento del virus a finales del 2019. Sin embargo, es el esperado debido a los años de pandemia que soporto la población mundial. En el grafico se puede observar que por "Keyword" elegido el pais que presento un mayor número de frecuencias fue Ecuador. Los siguientes paises fueron UK, Spain y Usa, los motivos de la frecuencia puede deberse a multiples factores, tales como la producción de farmacos en UK y USA que hicieron que realizen investigaciones en colaboración en Ecuador y puede el intercambio de datos para identificar posibles brotes
- En la segunda busqueda se obtuvieron 3 articulos de "Ecuador shrimp", siendo un número bajo dado la importancia del camaron en la industria productora del Ecuador. Dado el "keyword" se espera que el pais que presente mayor frecuencia sea Ecuador. El segundo pais fue USA que podría deberse a que es uno de los más importantes importadores del camaron ecuatoriano. Finalmente los paises latinos restantes, tales como México, Colombia y Chile puede deberse a la cooperación cientifica que existe en areas investigativas de interes.

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.

```
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] ## Seleccionamos las primeras 15 secuencias
out_sequence = open("secuencias.fasta", "w")
for linea in data:
    Entrez.email = "ibeth.calva@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()
```

In [3]:

```
# Escriba aquí su código para el ejercicio 6
Crustalw_exe = r"C:\Program Files (x86)\Crustalw2\clustalw2.exe"
Crustalw_cline = ClustalwCommandline(clustalw_exe, infile = "secuencias.fasta")
assert os.path.isfile(clustalw_exe), "Crustal_w executable is missing or not found"
stdout, stderr = clustalw_cline()
print(clustalw_cline)
ClustalAlign = AlignIO.read("secuencias.aln", "clustal")
print(ClustalAlign)

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

with open("secuencias.dn", "r") as aln:
    alignment = AlignIO.read(aln, "clustal")
from Bio.Phylo.TreeConstruction import DistanceCalculator
calculator = DistanceCalculator('identity')
distance_matriz = calculator.get_distance(alignment)
from Bio.Phylo.TreeConstruction import DistanceTreeConstructor
constructor = DistanceTreeConstructor(calculator)
#Construir el arbol
glut_1 = constructor.build_Tree(alignment)
glut_1.rooted = True
Phylo.write(glut_1, "tree_glut_1.xml", "phyloxml")
glut_1 = Phylo.read(file = "tree_glut_1.xml", format = "phyloxml")
```

```
# Arbol elemental en Matplotlib
import matplotlib
import matplotlib.pyplot as plt
#fig = Phylo.draw(as_tree)
fig = plt.figure(figsize=(30,40), dpi = 100) #create figure y set thesize
matplotlib.rc('font', size=30) #fontsize of the leaf and node labels
matplotlib.rc('xtick', labelsize=20) #fontsize of the tick labels
matplotlib.rc('ytick', labelsize=20) #fontsize of the tick labels
axes = fig.add_subplot(1, 1, 1)
Phylo.draw(glut_1, axes=axes)
fig.savefig("img/tree_glut_1.jpeg")
```

Escriba aquí la interpretación del árbol

El arbol filogenetico se puede distinguir a 13 clados definidos del gen GLUT-1 que codifica a la glucose transporter protein que esta involucrada en la ruta glucogenesis / glucolitica. Las secuencias con el codigo NM_174602.2, NM_001105687.1 fueron las que presentaron mayor similitud y compartieron 9 clados, pudiendo indicar que ambas secuencias provienen de un mismo organismo.

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

```
Import Bio
from Bio.Seq import Seq
from Bio import Entrez
import re

def download_pubmed(Keyword):
    """
    Función que entrada pide la frase clave de busqueda y en output guarda un
    archivo que contiene los resultados de la busqueda, considerando los titulos.

    """
    Entrez.email = "ibeth.calva@est.ikiam.edu.ec"
    handle = Entrez.esearch(db="pubmed",
                            term=Keyword+"[Title]",
                            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()
    (id_list)
    handle.close()
    out_handle.write(data)
    out_handle.close()
    return id_list
```

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

Nombre [Apellido, Nombre]:

Ikiam
Universidad Regional Amazónica

```
import re
import csv
import matplotlib.pyplot as plt
from collection import Counter

def science_plots(data):
    """
```

Función que pide como entrada la busqueda de la función download_pubmeds y como resultado muestra un grafico de pastel indicando a los cinco paises de origen de autores que aparecieron mas veces.

```
    """
    with open("data/"+data, errors="ignore") as f:
        texto = f.read()
    texto = re.sub(r"\n\s{6}", " ", texto)
    countries_1 = re.findall(r"AD\s{2}-\s[A-Za-z].*, \s([A-Za-z]*)\.\s", texto)
    unique_countries = list(set(countries_1))
    conteo=Counter(countries_1)
    resultado={} #Hacemos un diccionario vacio
    ##Bucle agregamos los valores del diccionario
    for clave in conteo:
        valor=conteo[clave]
        if valor > 1:
            resultado[clave] = valor
    ##ordenamos de forma ascendente
    ##ordenamos de forma ascendente
    sort = (sorted(resultado.values()))
    sort.sort(reverse=True)
    import operator
    ##creamos dos listas vacias
    pais = []
    frecuencia =[]
    ## bucle que añade los valores pais y frecuencia a la listas vacias país y frecuencia
    reverse = sorted(resultado.items(), key=operator.itemgetter(1), reverse=True)
    for name in enumerate(reverse):
        pais.append(name[1][0])
        frecuencia.append(resultado[name[1][0]])
    max_país = pais[0:5] ## seleccionamos los cinco primeros países
    max_frec = frecuencia[0:5] ## Seleccionamos las cinco primero frecuencia respecto a los paises
    fig = plt.figure(figsize=(10,8))
    plt.pie(max_frec, labels=max_pais)
    (plt.savefig("img/"+data, dpi=100, bbox_inches='tight'))
    plt.show()
```