Python 大作业：基因工厂小程序

余谷风 519111910244

生命科学技术学院

一、 设计背景

有许多人谈到，21 世纪是生命科学的世纪，近些年里，生命科学犹如一个新兴学科，飞速发展。但实际上，生命科学已经发展了数百年甚至数千年之久，在很长一段内，生命科学包括医学的发展比较缓慢，即使是飞速发展的今天，其真实发展水平也是比较落后的。相比于其他学科，生命科学有着许多特点，如体系不全、模型不足、机制不完整，故很多时候，生命科学的研究就像古代的炼金术，需要运气和偶尔，无法有效地前进。

近二十年来，随着科技尤其是信息领域的飞速发展，为生命科学的进步提供了巨大的助推，并因此催生出许多新兴交叉学科如生物信息学、计算生物学等，这些学科的研究方向很广泛，但是其核心思想都是利用现代的计算机技术和数学思想来解决一些生物问题，为科学家的后续探究提供便利和较为明确的思路。

本人是生物信息学专业的一名学生，而基因在生物信息学领域属于主流，且覆盖了多个方面，故本项目取名为“基因工厂”，实现生物信息学中的一些简单功能。

二、 主要功能和实现思想

本小程序将实现三个简单功能，以下将进行分别介绍，并会对其中的背景和原理进行简单解释。

1. 序列比对的全局比对

在自然界中，几乎所有生物的 DNA 序列都由 ATGC 四种碱基组成，即在计算机中，DNA 是一条 ATGC 四种字符随机组成的字符串。生物的遗传信息均储存在 DNA 序列中，四种碱基的排列方式和比例储存着不同的信息。当同时提供两条序列时，通过比较其相似度，可以变相地比较两条序列所代表的遗传信息的差异度。

进行比对时，需要设置一个评分规则，两条序列之间完全匹配得 10 分，不全得 5 分，匹配到空位-5 分，使用动态规划算法，得到最大比对得分，并通过回溯，输出相应的最佳比对方式。

其具体公式形式以下所示：

$$F(0,0)=0$$

$$F(i,j)=\max\begin{cases}F(i-1,j-1)+s(x_i,y_j)\\F(i-1,j)+d\\F(i,j-1)+d\end{cases}$$

当然，这个程序还可以比对其它生物信息，如蛋白序列，RNA 等其他核酸序列，在此我们以 DNA 为例即可。

2. 差异表达分析

在生物体中，DNA 虽然储存了遗传信息，但是并不直接构成生物体，根据一系列的调控网络和表达机制如中心法则等，DNA 才能将遗传信息表达出来，构成现实中我们能实际观察得到的生物体。一般来说，生物体中的 miRNA、mRNA 等物质的含量或表达浓度取决于 DNA 基因，并且可以影响到生物性状。我们可以通过分析几个组别这些指标的差异来找出真正影响到生物性状的指标和因素，进而更好的进行后续研究。

近年来，随着医疗健康领域的发展，差异表达分析变得越来越重要，通过差异分析筛选出病人样本和正常人样本中差异较大的指标，进行后续的湿实验验证，找出引起疾病的基因或者物质，这可以大大提高医学研究的效率，极大减少了资源和时间的浪费。

本程序中的差异表达分析采用 fold change+统计检验的方式进行。首先设置差异倍数（fold change）的阈值（多以 2 为基准），筛选出差异超过阈值的指标，后进行的差异的显著性检验（t-test），挑出符合统计概念下有显著差异的指标，通常置信度设置为 0.05。

筛选出具有显著差异的特征后，将这些特征的名字储存在文件 feature_names.txt 中，在文件结尾指明了特征的数量，这个文件可以进行后续的使用。同时，这些特征对应的 fold change 值也将进行可视化，绘制出一个横坐标为特征名，纵坐标为 fold change 值的一个直方图，其中正负主要是代表相

对大小，即病人比正常人高为正，反之则为负，这个图片被保存为 bar.png。

3. 产生随机序列

生物数据是当今最大型的数据之一，很多时候下载和处理数据十分费功夫，在进行一些模拟实验和操作时，我们可能并不需要使用真正的序列，使用随机序列验证即可，故在本程序的最后附上这个小功能，根据需求产生对应条数和长度的随机序列，并储存在文件中，以备后续使用。

三、 使用方法

以下部分是本程序的使用方法，也可以称之为程序的帮助文档。

程序以一个压缩包的形式下载，解压之后打开主文件即 main.py 文件，运行这个文件，小程序开始运行。运行初始如下所示：

可以看到页面上有四个按键，分别对应开机键和三个功能，以及五个输入框，点击 On 按钮程序进入运行状态，接下来将对三个功能进行一一介绍。

1. 功能一：进行序列比对

在两个输入框中随机输入两个序列，点击“进行序列比对”按钮，计算其最大相似得分，并输出其最佳匹配方式。

这里的评分规则为两条序列之间完全匹配得 10 分，不全得 5 分，匹配到空位-5 分，如果你需要设置不同的规则，那么很简单，将源代码中的 dp_program.py 文件进行修改即可。

```python
# 一行一行的计算矩阵中的每个各自中的最优结果。当前格子中的最优结果由它的三个来源推出
def compute(array, seq1, seq2):
    row, col = len(seq2), len(seq1)
    for i in range(1, row+1):
        for j in range(1, col+1):
            if seq1[j-1] == seq2[i-1]:  # 这里简化了得分矩阵，完全匹配得10分，不完全得5分，有gap减5分
                s = 10
            else:
                s = 5
            lu = [array[i-1][j-1].score+s, [i-1, j-1]] # idx 0：最大得分，idx 1：来源坐标
            left = [array[i-1][j].score-5, [i-1, j]]
            up = [array[i][j-1].score-5, [i, j-1]]
            max_choice = max([lu,left, up], key=lambda x: x[0])
            score= max_choice[0]
            pointer = max_choice[1]
            array[i][j] = F(score, pointer)  # 在当前保存最大得分，和来源坐标，方便回溯。
    return array
```

10、5、-5 分别对应完全匹配、不全、空位，修改时改掉这四个值即可。

2. 功能二：进行差异分析

在输入框中输入需要读取的文件的名字（需要输入文件的格式），点击“进行差异分析”按钮，进行差异分析。由于往往读取的文件数据量较大，这个功能实现需要花一定时间，请耐心等待。

此外，这个功能的实现并不是全自动的，需要进行相应的手动操作。

```python
# 每个基因（行）BH样本（正常人员）的表达平均值和方差
bh = data.loc[:, 'BH11399-2_NBZX27_(HG-U133_Plus_2)' : 'BH11399-2_NZMY24_(HG-U133_Plus_2)']
bh_mean = data.loc[:, 'BH11399-2_NBZX27_(HG-U133_Plus_2)' : 'BH11399-2_NZMY24_(HG-U133_Plus_2)'].mean(axis=1)
bh_var = data.loc[:, 'BH11399-2_NBZX27_(HG-U133_Plus_2)' : 'BH11399-2_NZMY24_(HG-U133_Plus_2)'].var(axis=1)

# 每个基因（行）US样本（病人）的表达平均值和方差
us = data.loc[:, 'US-1047659':'US-1137753']
us_mean = data.loc[:, 'US-1047659':'US-1137753'].mean(axis=1)
us_var = data.loc[:, 'US-1047659':'US-1137753'].var(axis=1)
```

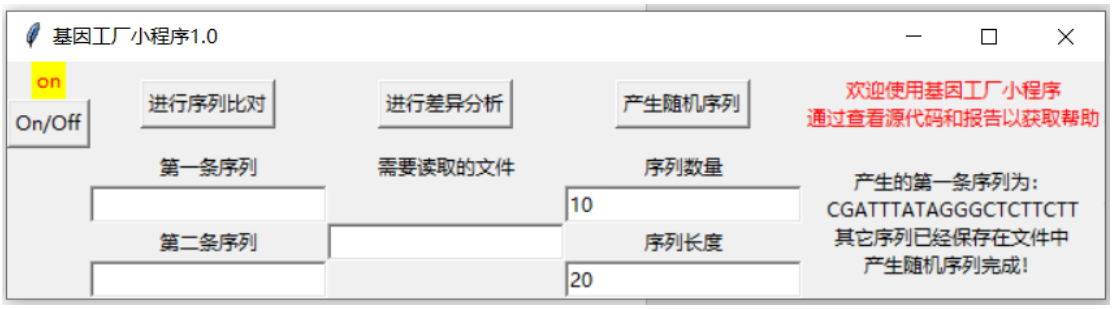
标黄的部分对应的是正常人员和病人的编号，由于在处理时我们是将样本看待成两个群体，所以要进行先一步的整合。将源代码中的 RDA.py 文件进行修改，一般来说，该类型的文件不会存在病人和正常人交替出现的情况，故将编号改成两个样本的首尾编号即可。

当然，这个函数还可以实现更加细致的功能，比如分别展示 up 组和 down 组。如果有需要，进行进一步的修改即可。

最终的结果保存在 bar.png 图片和 feature_names.txt 中，打开查看即可。

3. 功能三：产生随机序列

在输入框中分别输入需要的序列数量以及每一条序列的长度，点击“产生随机序列”按钮，随机产生 n 条长度为 m 的序列。



产生的随机序列保存在 seqs.txt 文件中，程序主界面会显示第一条序列。

当然，这个功能也能够进行进一步的修改，在这里我们默认产生的是 DNA 单链，如果你想得到其他的序列，比如说产生 RNA 序列或者是蛋白质序列，也是十分简单的。产生 RNA 序列时，将对应关系进行修改即可，即将 random_seq.py 中的 Tranverse 函数中的 T 改成 U。

```
def Tranverse(a):
    """将数字序列转化为ATGC"""
    seq=''
    for i in range(len(a)):
        if a[i]==0:
            seq+='A'
        elif a[i]==1:
            seq+='T'
        elif a[i]==2:
            seq+='G'
        else:
            seq+='C'
    return seq
```

由于蛋白质由氨基酸组成，而氨基酸的种类比碱基种类要明显多，故对应关系修改幅度较大，这里就不进行展示了。

四、 代码实现

代码文件和其他文件一起被压缩，一同上传。

本项目已上传至作者的Github上，项目名称为Gene_factory，如果感兴趣或想提出一些建议的话,可以给本人的Github留言或直接参与修改,我的Github账号是GufengYu520。