به نام خدا

امیرحسین صفری _ 93106455

گزارش پروژه درس مبانی بیوانفورماتیک

سوال 1:

قسمت 1 :

VP40 باعث گسترش این بیماری در بدن میشود.

VP24 این پروتیین پس از سرکوب کردن مسیر های انتقال سیگنال در سلول در آخر باعث سرکوب سیستم ایمنی بدن می شود و در نتیجه عامل اصلی این ویروس که سبب ایجاد بیماری میشود این پروتیین هست.

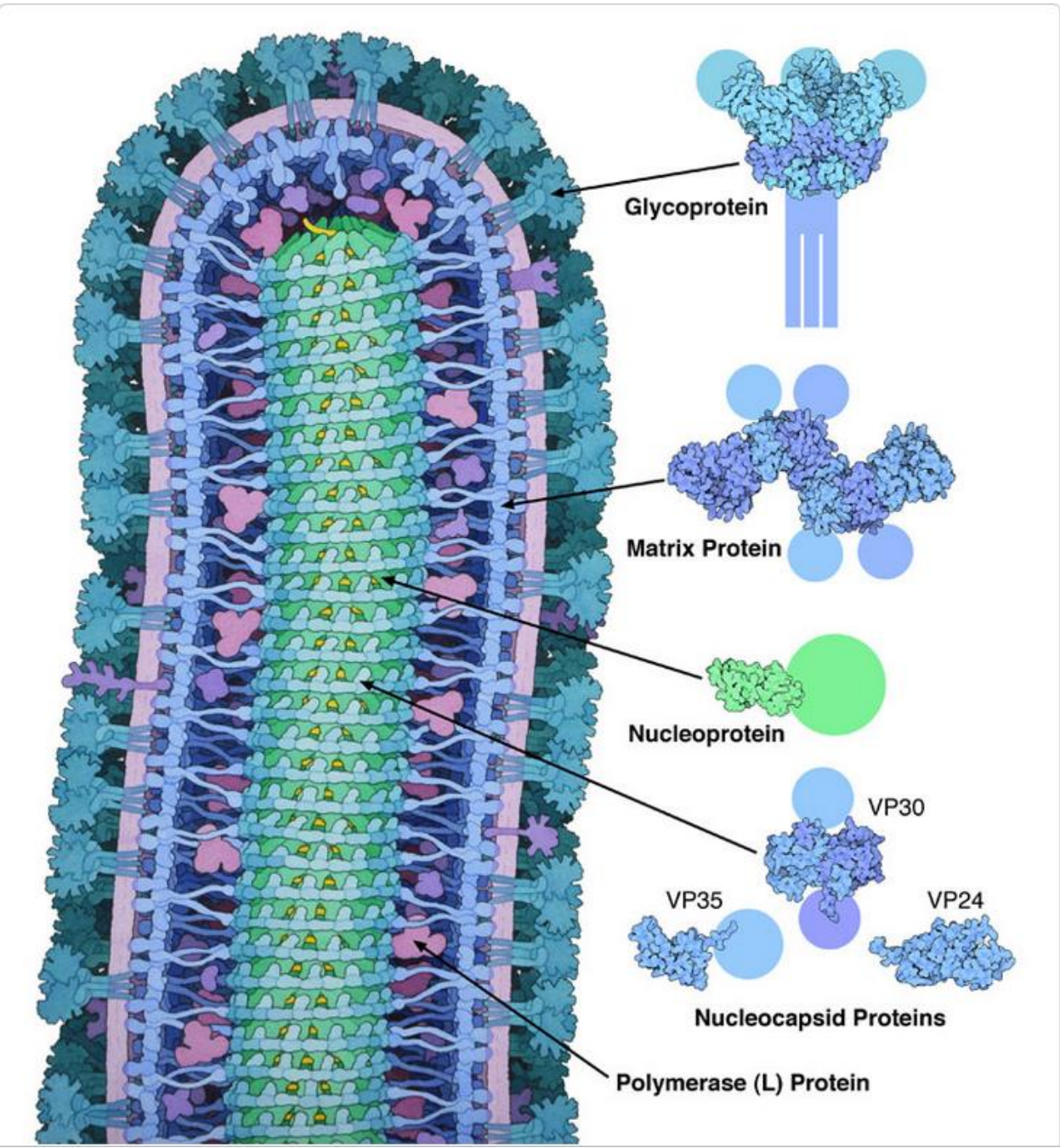
NP : انسداد ژنوم، محافظت از آن در مقابل هسته های هسته ای RNA .ژنومی محصور شده به نام nucleocapsid نامیده می شود و به عنوان الگو برای رونویسی و تکثیر استفاده می شود .در طی تکثیر و انباشته شدن توسط NP به سنتز RNA متصل می شود و تمام محصولات تکثیر کننده به نوکلئاز مقاوم هستند.

GP : مسئول اتصال به گیرنده (ها) در سلول های هدف است. تعامل با CD209 / DC-SIGN و CLEC4M / DC-SIGNR که به عنوان عامل سازنده برای ورود ویروس به سلول میزبان عمل می کند. اتصال به CD209 و CLEC4M، که به ترتیب بر روی سلول های دندریتیک (DCs)، و بر روی سلول های اندوتلیال سینوس های کبدی و سینوس های گره لنفاوی، عفونت ماکروفاژها و سلول های اندوتلیال را تسهیل می کند

L : RNA polymerase ای که باعث تسهیل در transcription برای viral mRNAs ها میشود. وظیفه آنها محدود کردن polyadenylation است

VP30 : بلافاصله بعد از شروع رونویسی به عنوان یک عامل ضد اتمام رونویسی عمل می کند اما بر طول عمر رونویسی تاثیر نمی گذارد .مشخص شده است که این تابع وابسته به تشکیلRNA stem-loop در محل شروع رونویسی ژن اول است .به RNA متصل می شود

VP35 : به عنوان یک cofactor در رونویس یا تکثیر آر ان ای پلیمراز عمل میکند، جلوگیری از ایجاد وضعیت ضد ویروسی سلولی بوسیله مسدود شدن فسفوریلاسیون ناشی از ویروس و فعال شدن عامل فاکتور 3 تنظیم کننده اینترفرون(IRF3) ، یک عامل رونویسی مهم برای القاء اینترفرون های آلفا و بتا است.



پروتیین ساختاری به پروتیین هایی که در ساختار سلول ها نقش دارند گفته میشود، پروتیین های غیر ساختاری نقش تنظیم کننده دارند مانند آنزیم ها. ژنی که تولید کننده پروتیین ساختاری است، ژن ساختاری نامیده میشود

ساختاری ها :NP, GP , VP24 , VP 30 , VP 35 , VP40

غیر ساختاری ها :L , (sGP)

پروتیینی که سبب ایجاد بیماری میشود VP24 هست، این پروتیین پس از سرکوب کردن مسیر های انتقال سیگنال در سلول در آخر باعث سرکوب سیستم ایمنی بدن می شود و در نتیجه عامل اصلی این ویروس که سبب ایجاد بیماری میشود این پروتیین هست.

همچنین پروتیین VP40 باعث گسترش این بیماری در بدن میشود. و میتوان به طریقی این پروتیین رو هم سبب ایجاد بیماری بدانیم.

منابع:

http://www.uniprot.org/uniprot/P18272

http://www.uniprot.org/uniprot/Q05127

http://www.uniprot.org/uniprot/Q05323

http://www.uniprot.org/uniprot/Q05318

http://www.uniprot.org/uniprot/A0A0E3H7K2

http://www.uniprot.org/uniprot/Q05320

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1440433/

قسمت 2 :

به صورت خلاصه این پروتیین پس از سرکوب کردن مسیر های انتقال سیگنال در سلول در آخر باعث سرکوب سیستم ایمنی بدن می شود. توضیح مفصل تر در ادامه آمده است:

این ویروس هنگام ورود به بدن در ابتدا سلول های ایمنی بدن را که اولین لایه دفاعی بدن هستند را هدف قرار می دهد. این ویروس، سلول های dendritic را تحت تاثیر قرار میدهند، این سلول ها به طور معمول سیگنال های عفونی را روی سطوح خود برای فعال کردن لنفوسیت های T نشان می دهند – گلبول های سفید خون که می توانند

سلول های آلوده دیگری را قبل از اینکه ویروس خودش را تکثیر کند، از بین ببرد .هنگامی که سلولهای دندریتی معیوب میشوند و قادر به ارسال سیگنال مناسب نیستند، سلول های T به عفونت پاسخ نمی دهند و همچنین آنتی بادی هایی که به آنها برای فعال شدن بستگی دارند، فعال نمیشوند. پس این ویروس می تواند بلافاصله و بسیار سریع تکرار شود.

ابولا، مانند بسیاری از ویروس ها، به مهار اینترفرون میپردازد، اینترفرون یک نوع مولکولی است که سلول ها برای جلوگیری از تولید بیشتر و ویروسی خود استفاده می کنند. برای مثال یکی از پروتئین های ابولا که VP24 نامیده می شود، مانع حرکت سلول immune میشود که این سلول نقش مهمی در اینترفرون ها دارد.

ابولا چگونه باعث خونریزی میشود:

همانطور که ویروس در خون به مکان های جدید می رود، دیگر سلول های ایمنی به نام ماکروفاژ ها آن را می خورند . پس از آلوده شدن آن ها، پروتئین ها را آزاد می کنند که باعث انعقاد، ایجاد لخته های کوچک در سراسر رگ های خونی و کاهش خون رسانی به اندام می شود .آنها همچنین پروتئین های سیگنال دهنده ی التهابی و اکسید نیتریک دیگری را تولید می کنند که باعث تخریب عروق خونی می شود و موجب نشت خون از آنها می شود.

ابولا چه ارگان هایی را به طور خاص هدف قرار میدهد:

ابولا باعث التهاب و تب در سراسر بدن می شود و همچنین می تواند به بسیاری از انواع بافت ها در بدن آسیب برساند، یا باعث ایجاد سلول های ایمنی مانند ماکروفاژها برای تولید مولکول های التهابی بشود و یا مستقیما باعث آسیب رسانی شود مثلا حمله به سلول ها و مصرف آنها از داخل .اما عواقب آن به ویژه در کبد خطرناک است، زیرا ابولا سلول هایی را که برای تولید پروتئین های انعقادی و سایر اجزای مهم پلاسما مورد استفاده قرار می گیرد را نابود می کند .سلول های آسیب دیده در دستگاه گوارش باعث اسهال می شوند که اغلب بیماران را در معرض خطر کم آبی قرار می دهد .و در غده آدرنال، این ویروس سلولهایی را ایجاد می کند که استروئید ها را تنظیم می کنند تا فشار خون را افزایش دهند و نارسایی گردش خون را ایجاد کنند که می تواند organs of oxygen را از بین ببرد.

در نهایت چیزی که باعث مرگ بیماران مبتلا به این ویروس میشود که است کهآسیب رسیدن به رگ های خونی باعث کاهش شدید فشار خون میشود و بیماران بر اثر شوک و یا از کار افتادگی چند ارگان جان خود را از دست میدهند.

منابع :

http://www.sciencemag.org/news/2014/08/what-does-ebola-actually-do

http://theconversation.com/what-happens-to-your-body-if-you-get-ebola-28116

سوال 2 :

قسمت 2 :

برای این قسمت از پروژه الگوریتم گلوبال را اینگونه پیاده سازی میکنیم که امتیاز گپ های میانی و عدم تطبیق برابر است با -1 ، امتیاز تطبیق برابر است با 1 و امتیاز گپ های پایانی و ابتدایی برابر با 0 است.

برای پیاده سازی این الگوریتم از کتابخانه BioPython در زبان برنامه نویسی پایتون استفاده میکنیم و از تابع pairwise2.align.globalms در این تابع اگر که پارامتر penalize_end_gaps=(False, False) را اینگونه مقدار دهی کنیم آنگاه گپ های اولیه و انتهایی را 0 در نظر میگیرد و الگوریتم کاملا مطابق خواسته ماست.

ابتدا به این روش عمل کردیم که با توجه به این که ترتیب ژن ها را داشتیم، ژن اول را الاین کردیم، سپس ژن دوم را به 100 نوکلیوتاید قبل از پایان ژن اول تا انتهای ژنوم را الاین کردیم و ....

نتایج این قسمت را ثبت کردیم اما نتایج به نظر خوب و منطقی نمی آمد، سپس این کار را کردیم که هر ژن را به کل ژنوم الاین کردیم، نتایج این قسمت بسیار بهتر و منطقی تر از قسمت قبل بود پس از این به بعد تمامی کار ها با استفاده از نتایج این قسمت انجام میشود.

همچنین الگوریتم گلوبال الاینمنت به صورت داینامیک پروگرمینگ هست و در آن در هر مرحله خانه های جدول خود را با توجه به شرایط خانه ای که در آن قرار داریم پر میکنیم. سودوکد الگوریتم:

```
for i = 1 to rows:
    for j =1 to cols:
        if ith character in seqA == jth character in seqB:
```

```
            score=match
        else
            score = mismatch
        # Now we have 3 choices
        choice1=a[i-1,j-1]+score     # If characters are aligned
        choice2=a[i-1,j] + gap       # Gap in seqB
        choice3=a[i,j-1] + gap       # Gap in seqA
        Let a[i,j] be max(choice1,choice2,choice3)
        if a[i,j] is choice1:
            let ptr[i,j] be 0        # Chars i and j aligned
        else if a[i,j] is choice2:
            let ptr[i,j] be 1        # Gap in seqB
        else
            let ptr[i,j] be -1       # Gap in seqA


i=length of seq A
j=length of seq B
while(i>0 or j>0):
    if ptr[i,j]==0:
        add ith character of seqA to alnseqA
        add jth character of seqB to alnseqB
        decrement i by 1
        decrement j by 1
    elif ptr[i,j]==1:
        add ith character of seqA to alnseqA
        add '-' to alnseqB
        decrement i by 1
    else:
        add '-' to alnseqA
        add jth character of seqB to alnseqB
        decrement j by 1
```

منابع:

http://www.iiserpune.ac.in/~farhat/courses/idc205/lab12.html

https://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm

قسمت 3 :

برای این قسمت هم از کتابخانه editdistance در زبان پایتون استفاده میکنیم و از تابع editdistance.eval استفاده میکنیم، این تابع به این روش عمل میکند که امتیاز تطبیق برابر با 0 است، امتیاز عدم تطبیق برابر با 1- و در آخر امتیاز حاصل را ضرب در 1- میکند و به عنوان خروجی به ما می دهد، برای مثال

```
print(editdistance.eval("CGGTT" , "CGGT"))
```

به ما عدد 1 را خروجی میدهد ، این روش از روشی که تطبیق را 1 در نظر بگیریم و در انتها تعداد تطبیق منهای عدم تطبیق هارا خروجی بدهیم منطقی تر است زیرا که برای ساختن درخت ها نیاز به ماتریس فاصله داریم و در روش اول که ما استفاده کرده ایم هر چه فاصله بیشتر، خروجی تابع هم عدد بزرگتری است که این مطلوب ما هست اما در روش دوم برعکس هست که به همین دلیل این روش مناسب نمیباشد.

الگوریتم کلی :

برای حل این قسمت هم مانند قسمت قبل از داینامیک پروگرمینگ استفاده میکنیم و برای پر کردن جدول خود اینگونه عمل میکنیم که اگر کاراکترهای دو رشته که الان در خانه آن ها هستیم برابر بودند، کار خاصی نمیکنیم در غیر اینصورت هر سه حالت ممکن یعنی درج ، حذف و ریپلیس را در نظر میگیریم و سعی میکنیم تا کمترین هزینه را بپردازیم (مینیمم میگیریم)، سودوکد:

```
dp = [[0 for x in range(n + 1)] for x in range(m + 1)]

# Fill d[][] in bottom up manner
for i in range(m + 1):
    for j in range(n + 1):

        if i == 0:
            dp[i][j] = j
        elif j == 0:
            dp[i][j] = i

        elif str1[i - 1] == str2[j - 1]:
            dp[i][j] = dp[i - 1][j - 1]
        else:
            dp[i][j] = 1 + min(dp[i][j - 1],
                               dp[i - 1][j],
                               dp[i - 1][j - 1])

return dp[m][n]
```

سپس خروجی های این کد را به صورت دستی وارد فایل csv. کردیم، که این فایل ها در فولدر P2.3 موجود می باشند. سطر اول همان هدر هست و 5 سطر دیگر عدد های ما.

منابع:

https://www.geeksforgeeks.org/dynamic-programming-set-5-edit-distance/

سوال 3:

عدد 1 نشانگر Tai

عدد 2 نشانگر Zaire

عدد 3 نشانگر Sudan

عدد 4 نشانگر Bundibugyo

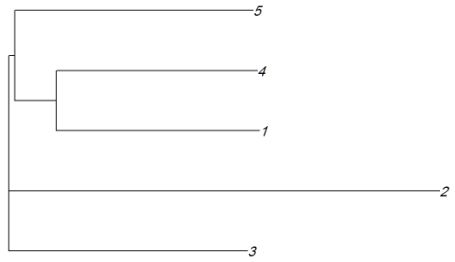
عدد 5 نشانگر Reston

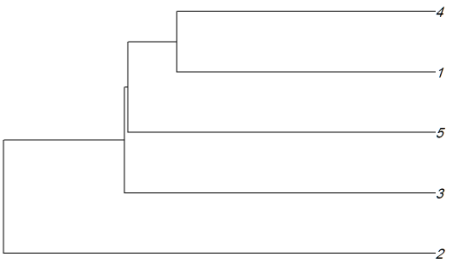
قسمت اول:

الف)

در همه قسمت ها تصویر بالا برای NJ هست و پایین برای UPGMA

برای ژن GP داریم:



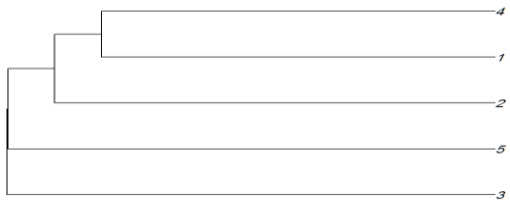


جد مشترک برای 1 و 4 در دو الگوریتم یکی هست همچنین برای (1و4) و 5 اما برای 2 و 3 بین دو الگوریتم اختلاف وجود دارد.

هر دو این الگوریتم ها این موضوع که کدام گونه از کدام گونه جدا شده است را مانند یکدیگر توصیف کرده اند

اما اینکه در ترتیب پیدایش چگونه بوده اند بین این دو الگوریتم اختلاف است و برای ترتیب پیدایش دو گونه شماره 2 و 3 اختلاف داریم.

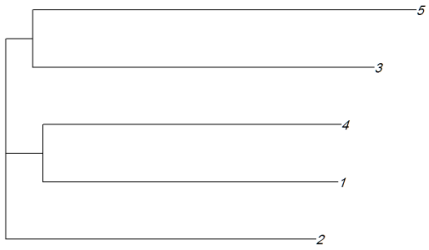
برای ژن L داریم:



هر دو این الگوریتم ها این موضوع که کدام گونه از کدام گونه جدا شده است را مانند یکدیگر توصیف کرده اند

همچنین اینکه در ترتیب پیدایش چگونه بوده اند هم بین این دو الگوریتم اختلافی نیست. تنها تفاوتی که میتوان گفت این است که گونه شماره 3 در الگوریتم پایینی جد مشترک متفاوتی دارد که با توجه به rates of evolution بسیار کم آن قابل چشم پوشی میباشد.

برای NP :



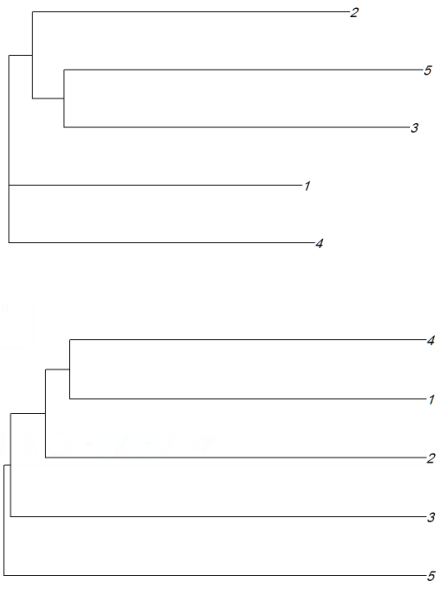
هر دو این الگوریتم ها این موضوع که کدام گونه از کدام گونه جدا شده است اختلاف دارند

همچنین اینکه در ترتیب پیدایش چگونه بوده اند بین این دو الگوریتم اختلاف هست

در کل الگوریتم nj فرض کرده است که 3 انشعاب از سلول ابتدایی داشته ایم و سپس یکی از آنها تبدیل به 2 شده دیگری به 1 و 4 و دیگری به 3 و 5

اما در الگوریتم دیگر در ابتدا دو انشعاب داشته ایم یکی به 5 تبدیل شده و دیگری باز دچار انشعاب شده و ....

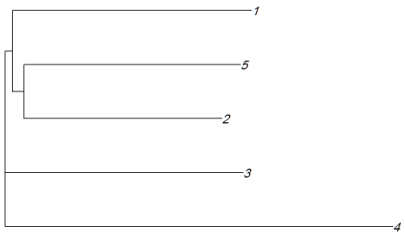
برای VP24 :

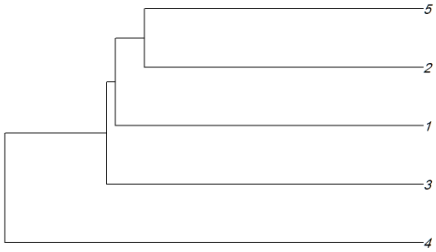


در جد مشترک این دو الگوریتم تفاوت هایی دیده میشود برای مثال در الگوریتم بالایی 1 و 4 و (2و3و5) مستقیما از گونه اصلی جدا شده اند اما در پایین 5 و بقیه از گونه اصلی جدا شده اند

اما در ترتیب پیدایش اختلاف است برای مثال الگوریتم اول پیدایش 4 و 1 را مستقیما از سلول اصلی دانسته اما دومی همچین چیزی را فرض نکرده

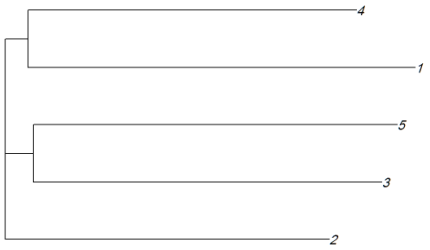
برای VP30 :





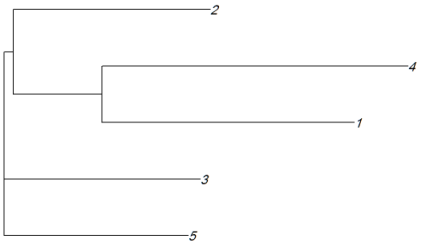
دو الگوریتم تا حد خوبی مانند یکدیگر عمل کرده اند فقط الگوریتم اول فرض کرده که 3 هم مستقیما از سلول اولیه منشعب شده است اما در الگوریتم دوم فرض شده که این سلول از انشعاب سلول اولیه به دست آمده است

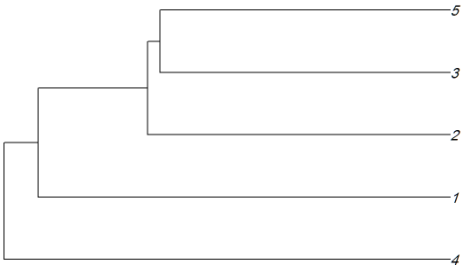
برای VP35:



در دو الگوریتم تفاوت های زیادی مشاهده میشود. در الگوریتم بالایی (2و4) و 3 دارای جد مشترک هستند اما در پایینی 3 و 5 دارای جد مشترک هستند! سیر تفاوت ها رو هم از روی شکل میتوان متوجه شد.

برای VP40:





در دو الگوریتم تفاوت هایی وجود دارد برای مثال در بالایی 5 و3 و بقیه از گونه اصلی جدا شده اند اما در پایینی 4 و بقیه مستقیما از گونه اصلی جدا شده اند و یا در بالایی 2 و (1و4) جد مشترک دارند اما در پایینی 2 و (3و5)

ب)

بله تفاوت های چشمگیری بین دو الگوریتم وجود دارند، واضح ترین تفاوت این است که در بعضی از ژن ها، جد مشترک و ترتیب حضور گونه ها متفاوت است مثلا برای VP24 اینگونه است که در الگوریتم NJ 3 و 5 جد مشترک دارند و سپس (3و5) و 2 جد مشترک دارند و سپس 1 و 4 و (2و3و5) جد مشترک دارند اما در الگوریتم دیگر ترتیب کاملا متفاوت است و در این الگوریتم 1 و 4 جز آخرین گونه های ایجاد شده اند!

ج)

علت این تفاوت این است که این دو از الگوریتم های متفاوتی برای ساختن درخت استفاده میکنند،UPGMA

یک rate of evolution ثابتی را در نظر میگیرد که این کار خوبی برای ساخت phylogenetic trees نیست همچنین این الگوریتم یک hierarchical clustering method ساده هست،

الگوریتم UPGMA یک درخت ریشه دار را ایجاد می کند که ساختار موجود در یک pairwise similarity matrix را نشان می دهد. در هر مرحله، نزدیکترین دو خوشه به یک خوشه سطح بالایی ترکیب می شوند. و در هر مرحله، فاصله رو به بالا برای کلایتر جوین شده A و B و یک کلاستر جدید به اسم ایکس، اینگوه تعریف میشود:

$$d_{(\mathcal{A}\cup\mathcal{B}),X} = \frac{|\mathcal{A}| \cdot d_{\mathcal{A},X} + |\mathcal{B}| \cdot d_{\mathcal{B},X}}{|\mathcal{A}| + |\mathcal{B}|}$$

الگوریتم NJ یک bottom-up clustering method است که برای ساخت درخت استفاده میشود، این الگوریتم به عنوان ورودی ماتریس فاصله را مشخص می کند که فاصله بین هر جفت تاکسون را مشخص می کند. الگوریتم با یک درخت کاملا حل نشده آغاز می شود که توپولوژی آن مربوط به یک شبکه ستاره است و در طی مراحل زیر تکرار می شود تا زمانی که درخت کاملا حل شود و تمام طول شاخه ها شناخته شده است. مراحل آن به شرح زیر است:

1 _ ساخت ماتریس Q

2 _ پیدا کردم جفت هایی که Q(I,j) کمترین مقدار را دارد

3_ محاسبه فاصله از هر Taxa در جفت به این گره جدید.

4 _ محاسبه فاصله از هر Taxa خارج از این جفت تا گره جدید.

5 _ الگوریتم را دوباره راه اندازی میکنیم، جای جفت همسایگان با گره جدید و با استفاده از فاصله های محاسبه شده در مرحله قبلی.

منابع :

https://en.wikipedia.org/wiki/UPGMA

https://answers.yahoo.com/question/index?qid=20100525075826AAqtR46

https://en.wikipedia.org/wiki/Neighbor_joining#The_algorithm

د)

اگر که در حال استفاده از فاصله های تصحیح شده و تعدیل شده هستیم (تصحیح شده توسط مدل درست تکامل) و یا تکامل strict molecular clock را دنبال میکند به طور کلی اگر که فاصله های بین گونه های ما دقیق و تعدیل شده هستند آنگاه UPGMA بهتر است. در غیر اینصورت NJ بهتر است

اما در حالت کلی الگوریتم UPGMA الگوریتم معتبر و قابل اعتمادی به حساب نمی آید و برای همین در اکثر مواقع از NJ استفاده میشود.

قسمت 2 :

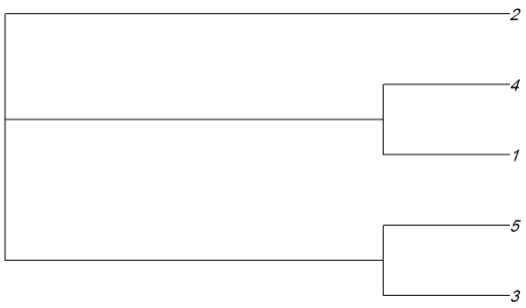
عدد 1 نشانگر Tai

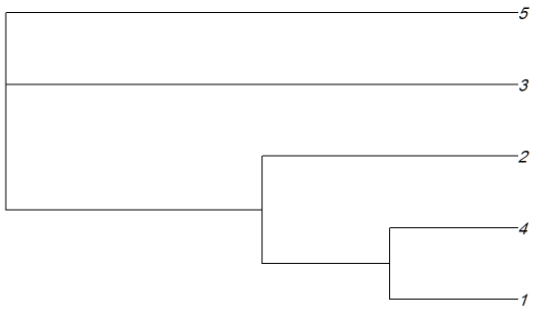
عدد 2 نشانگر Zaire

عدد 3 نشانگر Sudan

عدد 4 نشانگر Bundibugyo

عدد 5 نشانگر Reston





برای حل این قسمت اینگونه عمل میکنیم که از تابع consensus موجود در لایبرری ape در زبان برنامه نویسی R استفاده میکنیم. روش کار این تابع به این شکل هست که چند درخت را در یکدیگر مرج میکند. الگوریتم آن به شکل زیر است:

Input: A set $\mathcal{S} = \{T_1, T_2, \ldots, T_k\}$ of trees with $\Lambda(T_1) = \Lambda(T_2) = \cdots = \Lambda(T_k)$.

Output: The frequency difference consensus tree of $\mathcal{S}$.

```
    /* Preprocessing */
1   Compute w(C) for every cluster C occurring in S.

    /* Main algorithm */
2   T := T_1
3   for j := 2 to k do
        A := Filter_Clusters(T, T_j);  B := Filter_Clusters(T_j, T)
        T := Merge_Trees(A, B)
    endfor
4   for j := 1 to k do T := Filter_Clusters(T, T_j)
5   return T
```

این الگوریتم، الگوریتم کاملی است زیرا که نسبتا همه حالات محتمل را چک میکند و w را برای همه حالات چک میکند و سپس با اعمال فیلتر کلاسترهایی را حذف کرده و پس از آن با مرج کردن حالت های باقیمانده، درخت جواب را بدست می آورد. برای همین این الگوریتم، الگوریتمی کامل و درست هست و از آن استفاده کردیم.

قسمت 3 :

عدد 1 نشانگر Tai

عدد 2 نشانگر Zaire

عدد 3 نشانگر Sudan

عدد 4 نشانگر Bundibugyo

عدد 5 نشانگر Reston

برای حل این قسمت از همان روش قسمت 2.3 استفاده کرده ایم با این تفاوت که این بار کل رشته را به آن میدادیم که نتایج آن را میتوانید از فولدر P3.2 مشاهده کنید.

اولین تفاوتی که بین خروجی های این بخش و بخش قبل مشاهده میشود rates of evolution هاست که در دو قسمت تفاوت زیادی دارند.

همچنین در قسمت قبل فاصله های بسیار اندک بین نسل ها در الگوریتم UPGMAدر نظر گرفته نشده است برای مثال برای قسمت قبل 5 و 3 و (1و2و4) از یک جد مشترک هستند اما در این قسمت آن فرق اندک هم در نظر گرفته شده و در نتیجه برای (3و5) و (1و2و4) یک جد مشترک داریم

اما در الگوریتم NJ تفاوت ها برعکس الگوریتم قبل است! در قسمت قبل 2 و (1و4) و (3و5) یه جد مشترک داشتند اما در این قسمت 3 و 5 و (1و2و4)

در نهایت اگر که هر 4 درخت را با هم مقایسه کنیم به نظر میرسد که الگوریتم consensus برای درخت های حاصل از الگوریتم NJ خوب عمل نکرده است زیرا که تفاوت های آن درخت با 3 درخت دیگر زیاد است.

در نهایت به عنوان جمع بندی میتوان این مورد ها را با اطمینان گفت (اشتراکات هر 4 درخت) که 1 و 4 از یک نسل مشترک هستند، 3 و 5 از یک نسل مشترک هستند و با احتمال زیادی 2 و (1و4) از یک نسل مشترک هستند و (3و5) و (1و2و4) هم داری یک نسل مشترک میباشند.

همچنین درباره تفاوت دو الگوریتم این بخش میتوانیم بگوییم که تفاوت این دو برای جد مشترک 3 و 5 هست که در NJ نسبتا برابر با 0 هست و در نتیجه فرض میکنیم که این ها از همان ویروس ابتدایی منشعب شده اند اما در الگوریتم دیگر این مقدار قابل تشخیص و تمایز است.

قسمت 4:

عدد 1 نشانگر Marburg

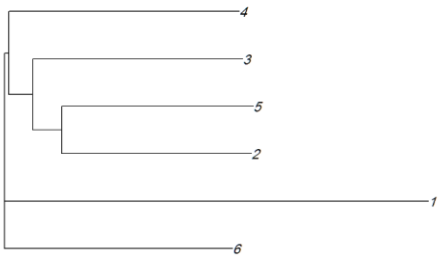
عدد 2 نشانگر Tai

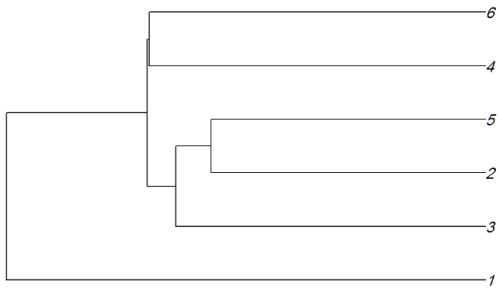
عدد 3 نشانگر Zaire

عدد 4 نشانگر Sudan

عدد 5 نشانگر Bundibugyo

عدد 6 نشانگر Reston





الگوریتم UPGMA از الگوریتم دیگر جواب منطقی تری میدهد زیرا که در درخت این الگوریتم مشاهده می شود که جد مشترک ابولا و ماربرگ یک ویروس است که پیش بینی ما هم همین بود و پیش بینی میکردیم که چون این دو به شدت به هم دیگر نزدیک هستند از لحاظ ساختار ژنوم پس باید از یک جد مشترک باشند، چیزی که از الگوریتم NJ میتوان برداشت کرد این است که ماربرگ و رستون و دیگر ویروسی های ابولا از یک جد مشترک هستند که به همین دلیل UPGMA بهتر عمل کرده است.

از هر دو الگوریتم میتوان این برداشت را کرد که **اولین** ویروس ابولا **رستون** بوده است. زیرا در NJ اگر که یال ها را بر حسب طولشان مرتب کنیم آنگاه رستون کوتاه ترین یال را دارد پس اولین ویروس بوده است، در UPGMA هم همانطور که میبینید با توجه به روند انشعابات انتظار میرودد که 4 یا 6 اولین ویروس باشد که با توجه به طول یال ها رستون جواب است.

اما با توجه به مطالبی که گفته شد، الگوریتم UPGMA درخت منطقی تر و درست تری را با توجه به وجود ماربرگ کشیده است اما در NJ تشخیص ویروس اول واضح تر است.

در نتیجه به نظر میرسد که در این مساله **UPGMA بهتر و قابل اعتماد تر** است. اما باید این را در خاطر داشته باشیم که در حالت کلی این الگوریتم غیر قابل اعتماد است!

همچنین همانطور که در **3.1** هم گفته شد و با توجه به شرایط مساله ما استفاده از **UPGMA** بهتر است زیرا که دیتا ما تا حد خوبی تصحیح شده است یا حداقل فرض من بر این است. اما در اینجا خروجی هر دو الگوریتم را برای دقت بیشتر آورده ایم.

سوال 4:

عدد 1 نشانگر Marburg

عدد 2 نشانگر Tai

عدد 3 نشانگر Zaire

عدد 4 نشانگر Sudan

عدد 5 نشانگر Bundibugyo

عدد 6 نشانگر Reston

قسمت 1:

در این قسمت از روش آماری Jukes and cantor استفاده میکنیم. این مدل روشی ست که احتمال جایگزینی از یک حالت را محاسبه میکند و به صورت خاص این مدل برای نوکلیوتاید ها تعریف شده است که برای البته به سادگی میتواند برای کدن ها و آمینو اسید ها هم استفاده بشود.

ایده اصلی این روش این است که احتمال اینکه از یک حالت به حالت دیگری برویم همیشه برای حالت های مختلف این احتمال برابر است، برای مثال اینکه از حالت "C" به "G" برویم برابر است با اینکه از "C"به "T" برویم. که البته میدانیم در واقعیت این فرض ممکن است غلط باشد.

همچنین فرض میکنیم که سایت های متفاوت (different sites) ها مستقل از یکدیگر هستند. بنابراین با توجه به شرایط این مدل و تطابق آنها با شرایط مساله ما و همچنین اینکه این مدل، مدلی معروف برای این نوع مسایل هست و همچنین در کلاس تدریس شده است پس از این مدل استفاده میکنیم.

در این قسمت ابتدا از زبان R برای برنامه نویسی استفاده کردیم و در آن از تابع dist.dna استفاده شد. در این قسمت اشکالات زیادی از جمله فرمت ورودی گرفتن این تابع و یکسان نبودن طول رشته ها وجود داشت.

برای فرمت ورودی ها از کتابخانه Biostrings و دستور DNAStringSet موجود در این کتابخانه استفاده شد برای مشکل یکسان نبودن طول رشته ها هم رشته های پس از الاینمنت را استفاده کردیم برای مثال رشته "CGT" را به شکل "C-GT" استفاده کردیم.

جواب ها با توجه به نرخ جهشی که در صورت سوال گفته شده است اصلا منطقی نبود و جواب های بدست آمده از عمر تخمینی کیهان بیشتر بود!

جواب های بدست آمده در شکل زیر آمده است:

```
             1            2            3            4
2 4.232437e+89
3 6.491437e+86 8.953401e+82
4 2.811414e+70 2.664033e+88 7.982871e+54
5 5.342286e+56 8.674071e+92 2.382864e+78 5.814544e+79
```

با توجه به اینکه به نظر میرسید که *گاما* که وردی این تابع بود تعریفی متفاوت با برداشت *ما* یعنی نرخ جهش دارد لذا از روش دیگری به شرح زیر استفاده شد:

در پایتون، کدی زده ایم که فاصله دو به دو ویروس ها را به *ما* می دهد، از طرفی با استفاده از روش جکس کانتور، فرومول زیر را میدانیم :

$$T = -3/4 * \ln(1 - 4P/3)$$

که در این فرمول :

$$P = \text{number of different} / \text{length of longest genome}$$

در پیاتزا گفته شد که آوردن اثبات ها نیازی نیست اما برای اثبات این قضیه میتوانید به منبع اول مراجعه کنید.

با استفاده از کد پایتونی که زده ایم، number of different را بدست می آوریم سپس P را بدست می آوریم و سپس T را.

در انتها با توجه به نرخ جهشی که گفته شده است، جواب *ما* تقسیم عدد به دست آمده تقسیم بر نرخ جهش هست.

کد مربوط به محاسبه آن در پایتون را میتوانید در تصویر زیر مشاهده کنید.

```
for i in range(0,5):
    for j in range(i,5):
        str1 = data[i]
        str2 = data[j]
        score = editdistance.eval(str1,str2)
        if len(str1) > len(str2):
            tmp = -3/4*math.log(1-(4/3)*(score/len(str1)))
            print (tmp / 0.0019)
        else:
            tmp = -3 / 4 * math.log(1 - (4 / 3) * (score / len(str2)))
            print(tmp / 0.0019)
```

نتایج حاصل از این بخش به شرح زیر است که این نتایج به نظر منطقی می آیند :

	A	B	C	D	E
1	0	226.52949031130447	257.4559260718033	195.05050646345248	256.3368981300215
2	226.52949031130447	0	253.60427399546967	229.0011607775698	251.34133619634574
3	257.4559260718033	253.60427399546967	0	258.3575115229352	257.4458022416669
4	195.05050646345248	229.0011607775698	258.3575115229352	0	257.39682105984525
5	256.3368981300215	251.34133619634574	257.4458022416669	257.39682105984525	0

همچنین اگر علاوه بر گونه های ابولا، ماربرگ را هم اضافه کنیم آنگاه جواب ها به شکل زیر است :

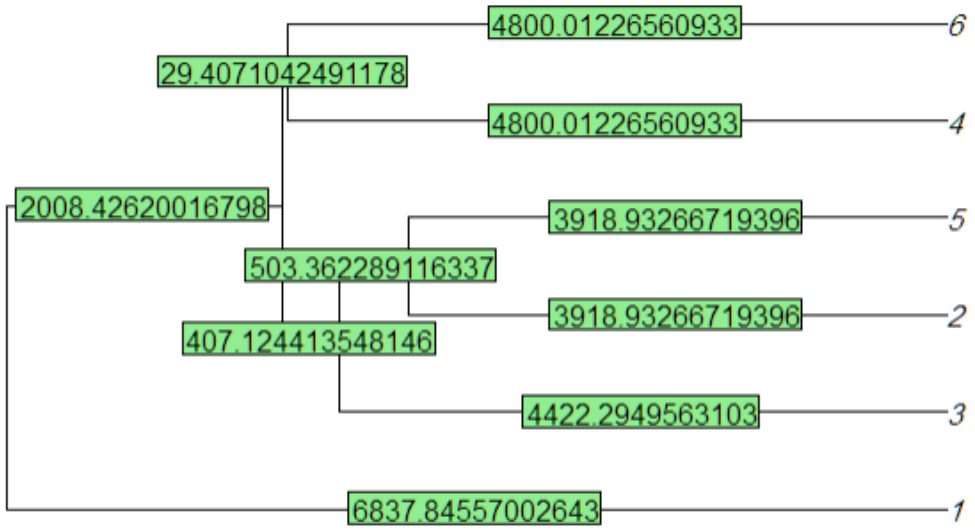
	A	B	C	D	E	F
1	0	393.66463860171353	389.21036960196017	389.431893346504	395.91076449015264	384.80580290675886
2	393.66463860171353	0	226.52949031130447	257.4559260718033	195.05050646345248	256.3368981300215
3	389.21036960196017	226.52949031130447	0	253.60427399546967	229.0011607775698	251.34133619634574
4	389.431893346504	257.4559260718033	253.60427399546967	0	258.3575115229352	257.4458022416669
5	395.91076449015264	195.05050646345248	229.0011607775698	258.3575115229352	0	257.39682105984525
6	384.80580290675886	256.3368981300215	251.34133619634574	257.4458022416669	257.39682105984525	0

حال برای تخمین زدن زمان جدا شدن گونه ها از جد خود داریم :

در درخت 3.4 دستور زیر را اضافه میکنیم :

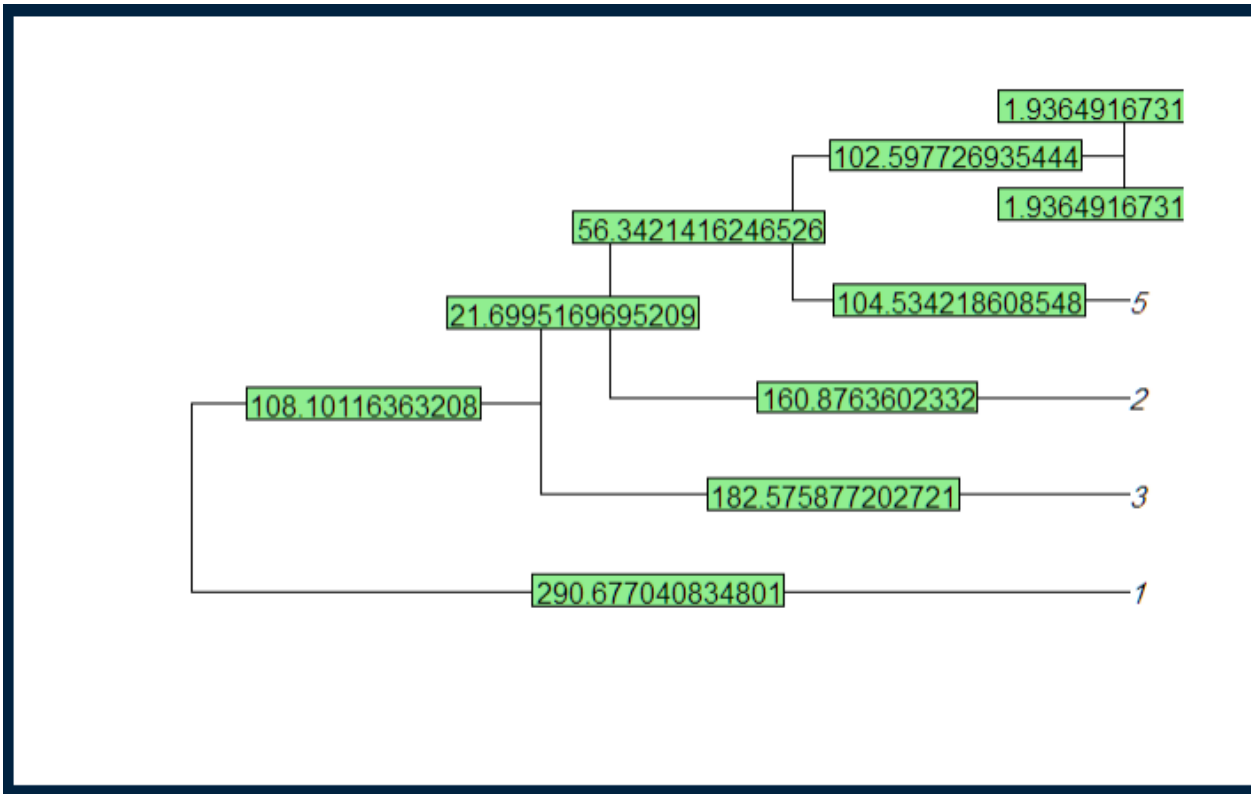
```
edgelabels(tree$edge.length,  col="black", font=1)
```

حال میتوانیم فاصله ها را مشاهده کنیم



حالا به این روش عمل میکنیم که اگر این عدد ها را به جای پی در فرمول گفته شده بگذاریم میتوانیم به جواب برسیم.

اما روش مطمین تر و بهتر این هست که اعدادی که از این بخش به دست آوردیم یا همان ماتریس زمانی را روی آنها این الگوریتم UPGMA را اجرا کنیم، خروجی :



حال در این شکل میتوانیم به صورت حدودی زمان جدا شدن گونه ها از جد خود را بگوییم، جد اصلی 290 سال پیش می زیسته است و یا اینکه zaire حدودا 180 سال پیش از جد خود جدا شده است و ...

منابع :

www.montefiore.ulg.ac.be/~kvansteen/GBIO0009-1/ac20132014/T4/jc.pdf

https://mathcs.clarku.edu/~djoyce/java/Phyltree/mutations.html

www.cs.utoronto.ca/~brudno/csc2427/Lec4Notes.pdf

قسمت 2 :

برای این بخش اینگونه عمل میکنیم که میتوانیم با بررسی گونه های جدا شده، مکان جهش هایی که باعث ایجاد گونه جدید میشوند را استخراج کرد، زیرا که میدانیم جد مشترک گونه ها به چه شکل بوده است و کدام دو گونه از

جدی مشترک جدا شده اند. حال اگر بیاییم اینگونه تحقیق کنیم که جهش در کدام مکان ها و در کدام ژن ها باعث ایجاد نسل جدید میشود میتوانیم آن مکان ها و ژن ها را بدست بیاوریم

یا برای مثال قسمت هایی از ژنوم که خارج از ژن هستن اصلا برای ما اهمیتی ندارند و درنتیجه میتوان آنها را نادیده گرفت

برای مثال فرض کنیم که در همه گونه های جدید، یکی از مکان ها جهش داشته است، اینگونه میتوان متوجه شد که احتمالا ان مکان و آن ژن نقشی کلیدی در جهش داشته است.

حال که این مکان های کلیدی را داریم میتوانیم اینگونه بررسی کنیم که احتمال اینکه تمامی این مکان ها با همدیگر دچار جهش بشوند چقدر است، از روی این احتمال میتوان یک توزیع احتمال مناسب به دست آورد.

منابع جهت الهام گرفتن:

https://www.theatlantic.com/health/archive/2017/06/evolution-of-the-flu/531693/

https://theconversation.com/scientists-create-accurate-predictor-of-the-next-flu-virus-23577

مخزن گیت هاب :

https://github.com/amirhoseinsa/Bio

کدهای پایتون و قسمت های خوبی از کد های آر کامنت گذاری شده اند.

پایان

امیرحسین