Graded Assignment Comparative Genomics

Comparative and Regulatory Genomics (B-KUL-I0U29A)

Mianyong Ding

R0823572

Introduction

Paralog and Ortholog

The paralogous genes in one organism are the genes arising by duplication events which are highly related. The orthologous genes of two species are the genes arising by speciation events. The paralogs can be divided as in-paralogs and out-paralogs. For the in-paralog, the speciation events happen before the lineage-specific duplication events where the paralogous genes arose. And for out-paralog, the lineage-specific duplication events for arising paralogous genes occur before the speciation events. When two in-paralogous genes are orthologous to the genes in other lineages. The two in-paralogous genes are co-orthologous to this third gene. The study on the paralog and ortholog can obtain information about the evolution. There are several common ways to find the paralog and ortholog such as BBH (Overbeek et al, 1999). and phylogenetic tree construction method.

Bidirectional best hits (BBH) method is a common method to find the ortholog of two species based on pairwise alignment score. The two genes from two species can be classified as one pair in BBH, if they are most similar to each other. Choosing the top-ranking matches from the output BLAST is the common way to estimate the pairwise BBH (Altschul et al, 1990). From the BBH, the in-paralogs can be found if the two genes from one species are more similar than the genes from other species. The construction of phylogenetic trees can also identity the orthologs and paralogs. The duplication events happen when the species overlapping in the branch and the speciation events can also be identified.

Conservation in the gene sequence.

The conservation region reserves the important function. There are some conserved positions in the gene sequences which may have important function such as acting as interaction sites.

To measure the conservation of sequence, several criteria can be used such as measurement of variability (Wu-Kabat, 1977), Shannon Entropy (Sander & Schneider 1991) and Simpson Diversity. For the Shannon entropy, the equation to calculate can be described as below with Pi representing fraction of residues of amino acid type and M presenting number of amino acid types.

$$H = -\sum_{i=1}^{M} P_i \log_2 P_i$$

Objective

For the assignment, the data of whole genome sequence of two species *Acetohalobium Arabaticum*(specie A) and *Hirschia baltica*(Specie B) are analyzed to find the orthologs of these two species with BBH methods. Phylogenetic tree method is used to check the ortholog based one co-orthologous gene sets picked from BBH. The species tree is constructed based on 16sRNA. And finally, the functional region of these sequence is identified.

Methods

Acetohalobium Arabaticum(GenBank assembly accession: GCA_000023785.1), and *Hirschia baltica*(GenBank assembly accession: GCF_000144695.1) were selected for finding ortholog. The Reference genome data of each species were obtained from NCBI GenBank FTP site. The protein.faa files of two species were downloaded which includes all protein sequence (last accessed Jan 14th, 2022).

Then the local blast + was run on the HPC, the makeblastdb command was used firstly to make the customized libraries of two species (library 1 of species A and library 2 of species B).

To find the ortholog, two blasts were carried firstly with blastp command. The protein sequence of specie A was blasted against library 2 and the protein sequence of species B was blasted against library 1. The generated blast files have the format of outfmt 6. To find the BBH, the python was used.

Two thresholds were conducted before generating BBH, threshold 1 for the cutoff of e-values and the threshold 2 for the cutoff the identity. The pairs lower than the two thresholds were filtered out. For each query of each species, the top one pair with the highest score were extracted. After extracting the best pairs of each query of two species, the same pairs were selected as the BBH.

To find the paralog, the protein sequence of species A was blasted on library 1, the protein sequence of species B was blasted on library 2. The two output files have the format of outfmt 6. The two thresholds for paralog were used to remove some pairs with low identity and e-cutoffs. For each query, the top one pair was extracted as paralog. To find the in-paralog from BBH, two genes can be considered as in-paralog if they are most similar than the pairs from BBH. To find sets co-orthologous genes, the in-paralogous genes were selected and one gene another species was found from BBH pairs.

The three genes WP_012778203.1, WP_015827394.1 and WP_013278500.1 which have the co-orthologous relationship were used. The WP_012778203.1 was used to extract 34 other homolog sequence from BLASTq website(https://blast.ncbi.nlm.nih.gov/Blast.cgi). These 33 sequences were merged with the sequences of WP_015827394.1 and WP_013278500.1. Using clustalw on HPC, the 37 sequences from 26 species were aligned and used to construct phylogenetic tree with 1000 bootstrap. To construct the species tree, the 16srna data of each species were extracted from silva website (https://www.arb-silva.de/). And all the 16sRna were aligned and the species tree were constructed by clustal W.

For the entropy, 37 sequences aligned by CLUSTAL 2.1 were used. The python was used to calculate the Shannon entropy as python script calculateentropy.py. The sites with gap were removed and sites with the presence of amino acid were calculated for Shannon entropy values.

Results are discussion

BBH for finding orthologs

Hirschia baltica has 2,297 protein coding sequences and *Acetohalobium Arabaticum* has 3,171 protein coding sequence. With the threshold of 10^ (-3) e-value and 30% identity, 714 BBH pairs was obtained. 575 paralog pairs in *Hirschia baltica* and 106 paralog *pairs Acetohalobium Arabaticum* were selected with 10^ (-40) for e-value and 50% for identity. The used threshold can remove some pairs with low similarity. The value of thresholds are based on other studies and modified here (Pérez-Bercoff et al., 2010). 20 in-paralogous pairs were extracted from *Hirschia baltica* and 0 pair was found in *Acetohalobium Arabaticum.* WP_012778203.1 and WP_015827394.1 in-paralogous with 55.128% identity from *Hirschia baltica*. They are co-orthologous to WP_013278500.1 of *Acetohalobium Arabaticum* with 43.2% identity. The 20-inparlog can detect 20 co-orthologous relationships. So, the final number of orthologous are 734 pairs.

The in paralogs can form a group of genes together and this group of genes are orthologous to a gene in another species, however the out-paralogs can never be orthologs. The BBH has its limits such as missing some orthologous relations if two species have frequent duplication events after the speciation events. The co-orthologous relationships can be formed by more than 3 genes, this is missed from BBH. For the study in 2013, 60% of orthologous relations are missed in plants and animals which have a high occurrence of duplication events (Dalquen & Dessimoz, 2013)). Gene loss can also not be detected.

Phylogenetic tree construction

The phylogenetic tree of homology tree is shown as figure s1(in appendix). The same species are in the same color (except black color). The bootstrap values can be seen. The bootstrap value close 1000 can indicate the splitting is very convinced. 26 species of 37 homologous sequences are shown in the figure. The stars with red colors are the duplication events and the blue stars are the duplication events. 9 duplication events and 25 speciation events are detected from the phylogenetic trees. The duplication events are identified by species overlapping. For our selected co-orthologous genes sets, WP_012778203.1, WP_015827394.1 and WP_013278500.1. The duplication event happened before two speciation events for WP_012778203.1, WP_015827394.1. The black star is the conflict event. The black star should be speciation events as the species overlapping of *Hischia sp*, however, the new species divergence of *Hischia sp* and *Hirschia baltica* suggests it should be speciation events. The conflict shows our in-paralog is plausible. And for WP_013278500.1 and other in-paralogous genes, the first root of speciation events make them separated.

Species tree construction

The species tree can be indicated as figure s2. Based on the 16sRNA, the genetic distance of species is shown here. From the species tree, we can see *Hirschia baltica* and *Acetohalobium arabaticum h*as the furthest distance in the tree which is consistent with homology tree. Some species with the same family are clustered together, such as Family Maricaulis. But *Hirschia baltica* is clustered with *Thalassobius mediterraneus*(CYSB01000034) with high bootstrap value other than other species from *Hirschia.*

The two strains were chosen for *Hyphomonas sp* to see how confident the species tree are (NCJT01000059.9650.11542 and CP017718.2300880.2303641). However, these two leaves are not clustered together, showing low confidence of this specie trees.

Conservation of sites of sequence

From the 37 sequences, 417 sites are detected, and 377 sites have amino acid in all 37 sequences. The site with lower entropy score means the same amino acid is always present in that site, representing higher conservation. The minimal value is 0, meaning the sites only contain one type of amino acid. Higher value of entropy means all amino acid are present with same frequency. From the figure, we can see the distribution of entropy of different sites. The summary of entropy score can be seen from figure. For the 377 sites, 84 highly conserved sites (22.2%) have the Shannon entropy with the value of 0. And 92 sites are between 0 and 0.5, 144 sites are larger than 1.5.

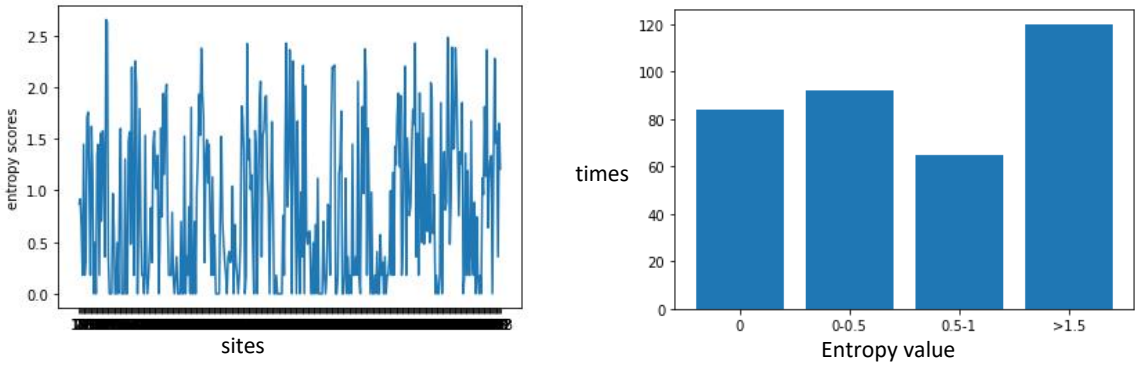


Figure 1. The relation between sites and entropy scores (left) and the frequency of each score(right).

We can see some clusters with several close sites from the figure 1. This WP_012778203.1 is relatively conserved with a large percent of sites with low Shannon entropy score. Checking the information about WP_012778203.1 from NCBI protein, it is aspartate aminotransferase family protein which can be involved in transamination or decarboxylation of various substrates.

This family protein is conserved in all organisms. The highly conserved sites may be the sites with interaction. Checking from NCBI, our aligned Site 232 and site 260 site which are site 217 and site 244 from the original sequence of WP_012778203, these two sites are conserved sites for pyridoxal 5'-phosphate binding and inhibitor-cofactor binding pocket.

Discussion

Here BBH is used for finding ortholog, the thresholds for e value and identity can make the result more stringent. It will lose some information, for instance, the large gene loss will be filtered out. The application of BBH have the limit of lacking co-orthologous relationships as the BBH only show the one-to-one orthologous relationship. So, the in paralogs are identified to find the co-orthologous relationship to have more orthologs. The threes genes with co-orthologous

relationship were chosen for constructing homology phylogenetic tree, the confidence of in-paralog is low. All the chosen homolog sequence are highly similar. Choosing some less similar homology sequence may produce better results.

The Shannon entropy was calculated for these homology sequence. The larger percentage of highly conserved sites shows the conservation. And some highly conserved sites are essential for the biological function.

Reference

Altschul, S.F. et al., 1990. Basic local alignment search tool. Journal of Molecular Biology, 215(3), pp.403–410.

Dalquen, D. A., & Dessimoz, C. (2013). Bidirectional best hits miss many orthologs in duplication-rich clades such as plants and animals. Genome Biology and Evolution, 5(10), 1800–1806. https://doi.org/10.1093/gbe/evt132
Pérez-Bercoff, Å., Makino, T., & McLysaght, A. (2010). Duplicability of self-interacting human genes. BMC Evolutionary Biology, 10(1). https://doi.org/10.1186/1471-2148-10-160

Overbeek, R. et al., 1999. The use of gene clusters to infer functional coupling. Proceedings of the National Academy of Sciences, 96(6), pp.2896–2901.
Kabat, D., 1977. Dividing the superconducting solenoid winding into cylindrical sections. Cryogenics, 17(11), pp.642–644.

Sander, C. & Schneider, R., 1991. Database of homology-derived protein structures and the structural meaning of sequence alignment. Proteins: Structure, Function, and Genetics, 9(1), pp.56–68.

Appendix

In this part, the python scripts for BBH and entropy calculation are shown.

The species tree and phylogenetic tree for homology are shown as figure s2 and figure s1.

The table s1 and table s2 are here to show the accession number and species name.

The entropy result of some sites, blast result, some orthologs can also be found.

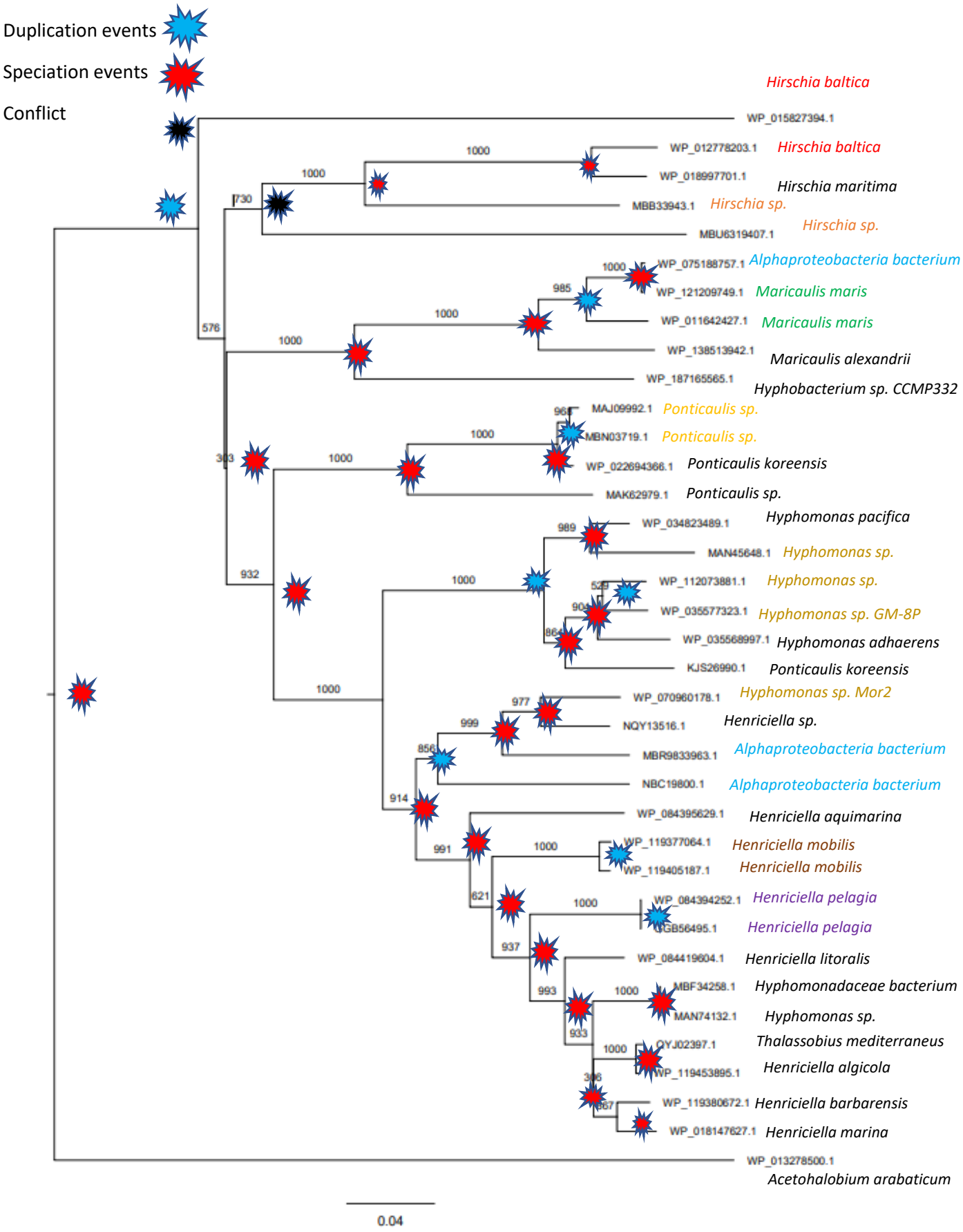


Figure s1. The phylogenetic tree of the homologs.

>WP_012778203.1 [Hirschia baltica]
>WP_018997701.1 [Hirschia maritima]
>MBB33943.1 [Hirschia sp.]
>MAJ09992.1 [Ponticaulis sp.]
>MBN03719.1 [Ponticaulis sp.]
>WP_022694366.1 [Ponticaulis koreensis]
>MAK62979.1 [Ponticaulis sp.]
>WP_119377064.1 [Henriciella mobilis]
>WP_084419604.1 [Henriciella litoralis]
>WP_119405187.1 [Henriciella mobilis]
>WP_070960178.1 [Hyphomonas sp. Mor2]
>MBU6319407.1 [Alphaproteobacteria bacterium]
>WP_084395629.1 [Henriciella aquimarina]
>WP_084394252.1 [Henriciella pelagia]
>GGB56495.1 [Henriciella pelagia]
>QYJ02397.1 [Thalassobius mediterraneus]
>WP_119453895.1 [Henriciella algicola]
>WP_119380672.1 [Henriciella barbarensis]
>MBR9833963.1 [Alphaproteobacteria bacterium]
>WP_018147627.1 [Henriciella marina]

>WP_034823489.1 [Hyphomonas pacifica]
>WP_112073881.1 [Hyphomonas sp. GM-8P]
>WP_035577323.1 [Hyphomonas jannaschiana]
>MAN45648.1 [Hyphomonas sp.]
>MBF34258.1 [Hyphomonadaceae bacterium]
>WP_075188757.1 [Maricaulis sp. W15]
>WP_121209749.1 [Maricaulis maris]
>WP_035568997.1 [Hyphomonas adhaerens]
>MAN74132.1 [Henriciella sp.]
>WP_011642427.1 [Maricaulis maris]
>WP_187165565.1 [Hyphobacterium sp. CCMP332]
>NBC19800.1 [Alphaproteobacteria bacterium]
>WP_138513942.1 [Maricaulis alexandrii]
>KJS26990.1 [Hyphomonadaceae bacterium BRH_c29]
>WP_013278500.1 [Acetohalobium arabaticum]
>WP_015827394.1 [Hirschia baltica]

Table s1. The gene 37 accession number of genes from 26 species.

Figure s2. The species trees.

Table s2. The accession number of 16sRNA and the name of species

>PBNO01000018.1.378 Ponticaulis sp.
>PAXZ01000018.1.373 Hirschia;Hirschia sp.
>DOLV01000060.1.1732 Hyphomonas sp.
>NZQD01000024.1.982 Hyphomonadaceae bacterium
>LQXG01000015.237.3044 Maricaulis sp. W15
>RBIM01000011.251.3047 Maricaulis maris
>ATVJ01000005.441.3193 Ponticaulis koreensis DSM 19734
>AWFE01000035.1164.3933 Hyphomonas sp. GM-8P
>CYSB01000034.2783.4263 Thalassobius mediterraneus
>ARKC01000001.2826.5595 Hirschia;Hirschia maritima DSM 19733
>CYSB01000034.4296.5600 Thalassobius mediterraneus
>DQGZ01000036.2861.4717 Alphaproteobacteria bacterium
>NCJT01000059.9650.11542 Hyphomonas sp. 34-62-18
>LADW01000084.86430.86829 Hyphomonadaceae bacterium BRH_c29
>NCST01000059.34747.37510 Henriciella;Henriciella aquimarina
>ARYJ01000003.156043.158810 Hyphomonas jannaschiana VP2
>QWFX01000016.243571.246350 Henriciella;Henriciella sp. JN25
>QWGB01000005.130332.133114 Henriciella barbarensis
>CP002105.12486.15436 Acetohalobium arabaticum DSM 5501
>AJ421782.1.1419 Hirschia baltica
>CP017718.2300880.2303641 Hyphomonas sp. Mor2
>AQXT01000002.1565757.1568518 Henriciella marina DSM 19595
>ARYH01000001.2005542.2008309 Hyphomonas adhaerens MHS-3

Python code 1 Build BBH to find ortholog

```
import pandas
import math
e_cutoff=-3
i_cutoff=50

# Read the blast output files with the format of outform 6, extracting the information as dataframe
def readout(file):
    dat=pandas.read_table(file,header=None)
    dat=dat.iloc[:,[0,1,2,10]]
    dat=dat.rename(columns={0: 'query',1:"subject",2:"identity",10:"evalue"})
    return dat

#Read the data of from readout()
#Find the top one pair of genes from two speices with the highest score
#whose evalue and identity are larger than the cutoffs
def findort(dat,e_cutoff,i_cutoff):
    dat=dat[dat["evalue"]<math.exp(e_cutoff)]
    dat=dat[dat["identity"]>i_cutoff]
    nrow=dat.shape[0]
    a1=[False]*nrow
    a1[0]=True
    for i in range(1,nrow-1):
        if dat.iloc[i,0]!=dat.iloc[i+1,0]:
            a1[i+1]=True
    dat=dat[a1]
    return dat

#Read the data of from readout()
#To find the top one pairs of self genes whose evalue and identity are larger than the cutoffs
def findpar(dat,e_cutoff,i_cutoff):
    dat=dat[dat["evalue"]<math.exp(e_cutoff)]
    dat=dat[dat["identity"]>i_cutoff]
```

```
    nrow=dat.shape[0]
    a1=[False]*nrow
    for i in range(0,nrow):
        if dat.iloc[i,0]!=dat.iloc[i,1]:
            a1[i]=True
    dat=dat[a1]
    return dat

#To find BBH based two top pairs of two species, dat1 and dat2 and the top pairs of each genes
#from different species
def findbbh(dat1,dat2):
    dat1=dat1.rename(columns={"subject": 'query',"query":"subject"})
    a1=dat1
    a2=dat2
    an=pandas.merge(a1, a2, how='inner',on=['query','subject'])
    an=an.rename(columns={"subject": 'gene1',"query":"gene2"})
    return an

#To find the inparalog of each genes
#a1_par is the paralogous gene pair generated by findpar() function
#to find the inparalog, select self pairs of genes are much more similar than the paris from BBH
def inpara(bbh,a1_par):
    a1_par
    nrow=a1_par.shape[0]
    inpar=[False]*nrow
    for i in range(0,nrow):
        if (a1_par.iloc[i,0]) in list(bbh.iloc[:,0]):
            a=list(bbh.iloc[:,0]).index(a1_par.iloc[i,0])
            inpar[i]=a1_par.iloc[i,:]["evalue"]<bbh.iloc[a,:]["evalue_y"]
    inpara=a1_par[inpar]

    return inpara
```

```
#read the blast result
a12=readout("s695vs785.txt")
a21=readout("s785vs695.txt")

a11=readout("s695vs695.txt")
a22=readout("s785vs785.txt")


#Find top pairs of two blast result for ortholog
a1_ort=findort(a12,-3,30)
a2_ort=findort(a21,-3,30)


# Find best self pairs for paralog
a1_par=findpar(a11,-40,50)
a2_par=findpar(a22,-40,50)

#la1 and la2 are the  lists of genes with paraplog
la1=list(a1_par.iloc[:,0])+list(a1_par.iloc[:,1])
la1=set(la1)
la2=list(a2_par.iloc[:,0])+list(a2_par.iloc[:,1])
la2=set(la2)

#print(a1_ort)
#print(a2_par)

#find the best bbh with the inparalog
bbh1=findbbh(a1_ort, a2_ort)
print(bbh1)

#inpar 2 is the inparalog of Hirschia baltica
inpar1=inpara(bbh1, a1_par)
```

```
inpar2=inpara(bbh1, a2_par)
cop=[None]*inpar2.shape[0]
#To find co-orthologous
for i in range(0, inpar2.shape[0]):
    b1=inpar2.iloc[i,0]
    i1=list(bbh1.iloc[:,0]).index(b1)
    o1=bbh1.iloc[i1,:]
    cop[i]=o1.iloc[1,]

df3 =pandas.concat([inpar2, pandas .DataFrame(cop)], axis=1)
df3=df3.iloc[:,[0,1,4]]
df3.to_csv("coorthologous.csv")
```

Python code 2 calculate the entropy

```
#This script is to read the alignment file generated from clustalw2 and calculate the shannon entropy of each sites.
import matplotlib.pyplot as plt
import pandas as pd
import math as ma
import string
#function to read file and make a dataframe of each line. Extract line start from "W" and extract the
#sequence of each gene, make the dataframe.
def readalign(file):
    alphabet_string = string.ascii_uppercase
    alphabet_list = list(alphabet_string)
    align=open(file).readlines()
    an=[]
    for i in range(1,len(align)):
       if list(align[i])[0]in alphabet_list:
           an.append(i)
    align = [align[i] for i in an]
    lena=len(align)
    name="name"
    seq1=[None]
    da = {name:seq1}
    df1= pd.DataFrame(da)
    for i in range(0,lena):
        name1=align[i].split(' ')[0]
        l11=list(align[i])
        indices1 = [index for index, element in enumerate(l11) if element == " "]
        length1=len(l11)
        seq2=l11[(indices1[-1]+1):(length1-1)]
        da1= {name1:seq2}
        df= pd.DataFrame(da1)
        df1 = pd.concat([df1, df], axis=1)
    dft=df1.transpose()
    dft=dft.drop("name")
```

```
    return dft


#the getseqnum is to get the number of sequences we have in this dataframe,the dft is object get from readalign
def getseqnum(dft):
    ilist=dft.index
    q1=dft.index[2]
    ind1 = [index for index, element in enumerate(ilist) if element == q1]
    times=len(ind1)
    numseq=int(dft.shape[0]/times)
    return numseq


#part means which part of alignment, the txt file is divided  in several parts, which is separeted with"    . *: .... ..
*:.***:  . :  .*  ::: :            "
#To know how many times of a sequence occuring in the file can know the number of parts.
def findpart(dft,part):
    numseq=getseqnum(dft)
    inn=list(range(numseq*(part-1),numseq*(part)))
    df3=dft.iloc[inn,:]
    return df3


# To get the final dataframe , each row represent the protein name and each column represents the alignment sites
def mergeall(dft):
    times=int(dft.shape[0]/getseqnum(dft))
    df=findpart(dft, 1)
    for i in range(1,times):
        df=pd.concat([df,findpart(dft,i+1)], axis=1)
    df= df.dropna(axis=1)
    ncol=df.shape[1]
    column=list(range(1,ncol+1))
    col=[str(x) for x in column]
    df.columns=col#to re-index the column name, making each colunmn name represent the position of sides
    return df
```

```
#this function two retrun the sites which have amino acid in each sequence, dff is the object getting from mergall()
def allhave(dff):
    ann=[]
    for i in range(0,dff.shape[1]):
        c1=dff.iloc[:,i]
        ntype=set(list(c1))
        if ('-' in ntype)==False:ann.append(i)
    return ann


#this function can calculate the shanno entropy of the sites whihc have all amino acid in each sequence.
def calentropy(dff):
    l1=allhave(dff)
    df2=dff.iloc[:,l1]
    allentropy=[]
    for i in range(0,df2.shape[1]):
        s1=list(df2.iloc[:,i])
        nt=list(set(s1))
        namino=len(nt)
        n=dff.shape[0]
        fre=[s1.count(nt[i]) for i in range(0,namino)]
        #with a list of frequency of each amino acid, now entropy is calcualted
        entropy=0
        for i in range(0,len(fre)):
            p=fre[i]/n
            entropy=-p*ma.log2(p)+entropy
        allentropy.append(entropy)
    df2.loc["entropy"]=allentropy
    return df2


# This function return the sites whose entropy is between the lower and upper, when lower=upper, it turn the value with certain entropy which is equal to lower and upper.
def getsites(withentro,lower,upper):
    onlyentro=entropy.iloc[-1,:]
    loca=[onlyentro.index[i] for i in range(0,onlyentro.shape[0]) if (onlyentro[i]>=lower and onlyentro[i]<=upper)]
```

```
    return loca


#Have a look at the result and functiion
dft1=readalign("allalign.txt")
print(dft1)
dff1=mergeall(dft1)
print(dff1)

print("number of sequence" , dff1.shape[0])
print("number of aligned sites", dff1.shape[1])



#print the sites which have the amino acid in each sequence
sites=allhave(dff1)
print("number of sites where seuqneces all have amino acids" , len(sites))

#the dataframe with shanno entropy in the last row
entropy=calentropy(dff1)
print(entropy)
entropy.to_csv("entropy.csv")

#the dataframe only have the entropy, with the index,we can know which set
onlyentro=entropy.iloc[-1,:]

#to know the sites with the given range
s1=getsites(entropy,0,0)
print("the sites whose entropy is 0 is ",s1)


plt.plot(pd.DataFrame(onlyentro))
plt.ylabel('entropy scores')
```

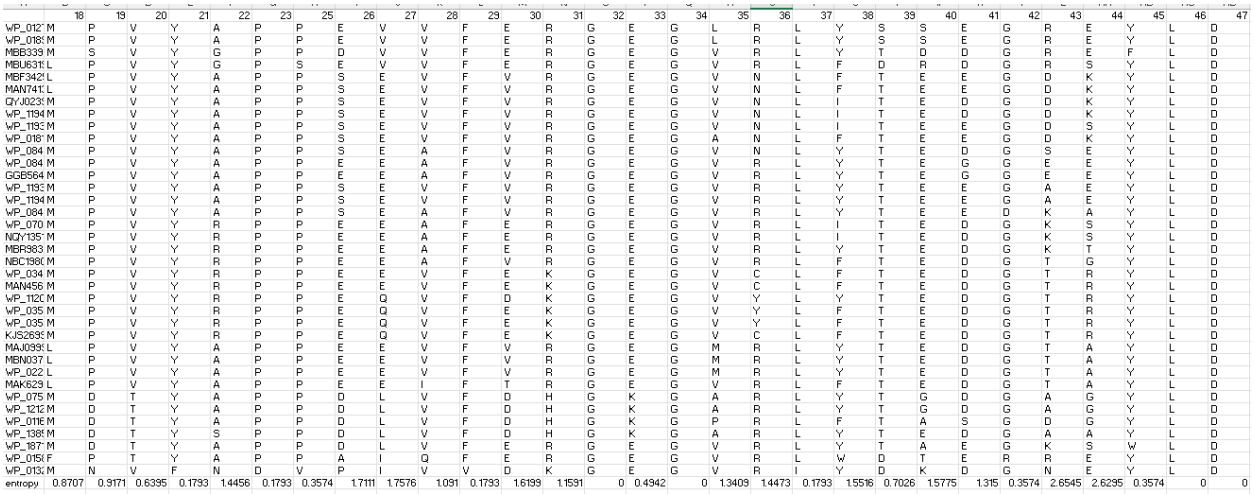
```
plt.show()


print(len(getsites(entropy,0.0000001,0.5)))
print(len(getsites(entropy,0.50001,1)))
```

Partial Output of BALST

WP_013277000.1	WP_013277000.1	100.000	449	0	0	1	449	1	449	0.0	909
WP_013277000.1	WP_013277217.1	28.947	76	48	3	113	188	117	186	0.060	31.6
WP_013277000.1	WP_013279048.1	28.571	42	30	0	176	217	225	266	2.5	26.2
WP_013277000.1	WP_013277934.1	26.437	87	44	2	200	286	36	102	2.6	26.2
WP_013277000.1	WP_013278883.1	25.714	70	50	1	97	164	219	288	2.8	26.2
WP_013277000.1	WP_013277038.1	22.807	57	37	1	113	169	8	57	3.6	25.8
WP_013277000.1	WP_013279036.1	32.911	79	48	2	293	367	236	313	5.5	25.4
WP_013277000.1	WP_013277689.1	32.394	71	41	3	97	166	9	73	5.9	25.0
WP_013277000.1	WP_013277439.1	52.381	21	10	0	120	140	269	289	8.5	24.6
WP_013277000.1	WP_013278942.1	35.484	31	20	0	35	65	485	515	9.1	24.6
WP_013277000.1	WP_013278305.1	44.444	18	10	0	149	166	142	159	9.4	24.6
WP_013277001.1	WP_013277001.1	100.000	365	0	0	1	365	1	365	0.0	731
WP_013277001.1	WP_013277241.1	36.111	36	23	0	271	306	20	55	0.98	25.8
WP_013277001.1	WP_013278909.1	39.024	41	23	2	216	254	255	295	2.2	26.2
WP_013277001.1	WP_013277090.1	27.184	103	69	3	201	298	362	463	2.5	26.2
WP_013277001.1	WP_013278852.1	30.667	75	44	2	87	153	171	245	3.5	25.4
WP_013277001.1	WP_013278627.1	38.095	42	21	1	143	179	49	90	4.0	25.4
WP_013277001.1	WP_013277734.1	22.277	202	128	9	163	342	103	297	5.0	25.0
WP_013277001.1	WP_013278130.1	29.310	58	38	1	221	278	13	67	5.2	23.5
WP_013277001.1	WP_013277450.1	25.758	66	48	1	242	306	436	501	8.9	24.3
WP_013277002.1	WP_013277002.1	100.000	69	0	0	1	69	1	69	2.89e-46	
WP_013277002.1	WP_013277871.1	34.426	61	37	2	2	60	5	64	0.003	30.0
WP_013277002.1	WP_013278626.1	32.727	55	35	2	13	66	4	57	0.066	26.6

Shannon entropy of some sites



	18	19	20	21	22	23	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
WP_012	M	P	V	Y	A	P	P	E	V	V	F	E	R	G	E	G	L	R	L	Y	S	S	E	G	R	E	Y	L	D
WP_018	M	P	V	Y	A	P	P	E	V	V	F	E	R	G	E	G	L	R	L	Y	S	S	E	G	R	E	Y	L	D
MBB339	M	S	V	Y	G	P	P	D	V	V	F	E	R	G	E	G	V	R	L	Y	T	D	D	G	R	E	F	L	D
MBU631	L	P	V	Y	G	P	S	E	V	V	F	E	R	G	E	G	V	R	L	F	D	R	D	G	R	S	Y	L	D
MBF342	L	P	V	Y	A	P	P	S	E	V	F	V	R	G	E	G	V	N	L	F	T	E	E	G	D	K	Y	L	D
MAN741	L	P	V	Y	A	P	P	S	E	V	F	V	R	G	E	G	V	N	L	F	T	E	E	G	D	K	Y	L	D
OYJ023	M	P	V	Y	A	P	P	S	E	V	F	V	R	G	E	G	V	N	L	I	T	E	D	G	D	K	Y	L	D
WP_119	M	P	V	Y	A	P	P	S	E	V	F	V	R	G	E	G	V	N	L	I	T	E	D	G	D	K	Y	L	D
WP_119	M	P	V	Y	A	P	P	S	E	V	F	V	R	G	E	G	V	N	L	I	T	E	E	G	D	S	Y	L	D
WP_018	M	P	V	Y	A	P	P	S	E	V	F	V	R	G	E	G	A	N	L	F	T	E	E	G	D	K	Y	L	D
WP_084	M	P	V	Y	A	P	P	S	E	A	F	V	R	G	E	G	V	N	L	Y	T	E	D	G	S	E	Y	L	D
WP_084	M	P	V	Y	A	P	P	E	E	A	F	V	R	G	E	G	V	R	L	Y	T	E	G	G	E	E	Y	L	D
GGB564	M	P	V	Y	A	P	P	E	E	A	F	V	R	G	E	G	V	R	L	Y	T	E	G	G	E	E	Y	L	D
WP_119	M	P	V	Y	A	P	P	S	E	V	F	V	R	G	E	G	V	R	L	Y	T	E	E	G	A	E	Y	L	D
WP_119	M	P	V	Y	A	P	P	S	E	V	F	V	R	G	E	G	V	R	L	Y	T	E	E	G	A	E	Y	L	D
WP_084	M	P	V	Y	A	P	P	S	E	A	F	V	R	G	E	G	V	R	L	Y	T	E	E	D	K	A	Y	L	D
WP_070	M	P	V	Y	R	P	P	E	E	A	F	E	R	G	E	G	V	R	L	I	T	E	D	G	K	S	Y	L	D
NQY135	M	P	V	Y	R	P	P	E	E	A	F	E	R	G	E	G	V	R	L	I	T	E	D	G	K	S	Y	L	D
MBR983	M	P	V	Y	R	P	P	E	E	A	F	E	R	G	E	G	V	R	L	Y	T	E	D	G	K	T	Y	L	D
NBC198	M	P	V	Y	R	P	P	E	E	A	F	V	R	G	E	G	V	R	L	F	T	E	D	G	T	G	Y	L	D
WP_034	M	P	V	Y	R	P	P	E	E	V	F	E	K	G	E	G	V	C	L	F	T	E	D	G	T	R	Y	L	D
MAN456	M	P	V	Y	R	P	P	E	E	V	F	E	K	G	E	G	V	C	L	F	T	E	D	G	T	R	Y	L	D
WP_112	M	P	V	Y	R	P	P	E	Q	V	F	D	K	G	E	G	V	Y	L	Y	T	E	D	G	T	R	Y	L	D
WP_035	M	P	V	Y	R	P	P	E	Q	V	F	E	K	G	E	G	V	Y	L	F	T	E	D	G	T	R	Y	L	D
WP_035	M	P	V	Y	R	P	P	E	Q	V	F	E	K	G	E	G	V	Y	L	F	T	E	D	G	T	R	Y	L	D
KJS269	M	P	V	Y	R	P	P	E	Q	V	F	E	K	G	E	G	V	C	L	F	T	E	D	G	T	R	Y	L	D
MAJ099	L	P	V	Y	A	P	P	E	E	V	F	V	R	G	E	G	M	R	L	Y	T	E	D	G	T	A	Y	L	D
MBN037	L	P	V	Y	A	P	P	E	E	V	F	V	R	G	E	G	M	R	L	Y	T	E	D	G	T	A	Y	L	D
WP_022	L	P	V	Y	A	P	P	E	E	V	F	V	R	G	E	G	M	R	L	Y	T	E	D	G	T	A	Y	L	D
MAK629	L	P	V	Y	A	P	P	E	E	I	F	T	R	G	E	G	V	R	L	F	T	E	D	G	T	A	Y	L	D
WP_075	M	D	T	Y	A	P	P	D	L	V	F	D	H	G	K	G	A	R	L	Y	T	G	D	G	A	G	Y	L	D
WP_121	M	D	T	Y	A	P	P	D	L	V	F	D	H	G	K	G	A	R	L	Y	T	G	D	G	A	G	Y	L	D
WP_011	M	D	T	Y	A	P	P	D	L	V	F	D	H	G	K	G	P	R	L	F	T	A	S	G	D	G	Y	L	D
WP_138	M	D	T	Y	S	P	P	D	L	V	F	D	H	G	K	G	A	R	L	Y	T	E	D	G	A	A	Y	L	D
WP_187	M	D	T	Y	A	P	P	D	L	V	F	E	R	G	E	G	V	R	L	Y	T	A	E	G	K	S	W	L	D
WP_015	F	P	T	Y	A	P	P	A	I	Q	F	E	R	G	E	G	V	R	L	W	D	T	E	R	R	E	Y	L	D
WP_013	M	N	V	F	N	D	V	P	I	V	V	D	K	G	E	G	V	R	I	Y	D	K	D	G	N	E	Y	L	D
entropy		0.8707	0.9171	0.6395	0.1793	1.4456	0.1793	0.3574	1.7111	1.7576	1.091	0.1793	1.6199	1.1591	0	0.4942	0	1.3409	1.4473	0.1793	1.5516	0.7026	1.5775	1.315	0.3574	2.6545	2.6295	0.3574	0	0

Some list of ortholog from BBH

gene1	gene2
WP_012777861.1	WP_013279269.1
WP_012777863.1	WP_013279265.1
WP_012777864.1	WP_013279264.1
WP_012777867.1	WP_013277570.1
WP_012777877.1	WP_148217699.1
WP_012777879.1	WP_013277379.1
WP_012777887.1	WP_013277636.1
WP_012777888.1	WP_013278433.1
WP_012777889.1	WP_013278240.1
WP_012777891.1	WP_013278703.1
WP_012777899.1	WP_013277686.1
WP_012777900.1	WP_013278340.1
WP_012777909.1	WP_013278260.1
WP_012777910.1	WP_013277391.1
WP_012777912.1	WP_013277390.1
WP_012777916.1	WP_013277383.1
WP_012777917.1	WP_013277382.1
WP_012777921.1	WP_013277381.1
WP_012777933.1	WP_013277532.1
WP_012777935.1	WP_013278019.1
WP_012777938.1	WP_013277877.1
WP_012777942.1	WP_013278391.1
WP_012777966.1	WP_013278962.1
WP_012777968.1	WP_013278807.1
WP_012777969.1	WP_013278968.1

The list of co-orthologs

WP_012778203.1	WP_015827394.1	WP_013278500.1
WP_015826687.1	WP_041301940.1	WP_013277225.1
WP_015827344.1	WP_015826739.1	WP_013277206.1
WP_015827575.1	WP_015827589.1	WP_013277956.1
WP_015827585.1	WP_015827329.1	WP_013277469.1
WP_015827611.1	WP_015827271.1	WP_013277449.1
WP_015827616.1	WP_015826803.1	WP_013278446.1
WP_015827641.1	WP_015828184.1	WP_013278778.1
WP_015827967.1	WP_015827266.1	WP_013278076.1
WP_015828160.1	WP_015828159.1	WP_013277258.1
WP_015828160.1	WP_015828149.1	WP_013277258.1
WP_015828160.1	WP_015828158.1	WP_013277258.1
WP_015828160.1	WP_174256054.1	WP_013277258.1
WP_015828160.1	WP_015828147.1	WP_013277258.1
WP_015828279.1	WP_015828626.1	WP_013277915.1
WP_015828503.1	WP_015828583.1	WP_013277991.1
WP_015828800.1	WP_015828651.1	WP_013277278.1
WP_041302020.1	WP_015828889.1	WP_013277047.1
WP_041302020.1	WP_015827204.1	WP_013277047.1
WP_041302020.1	WP_015826272.1	WP_013277047.1