Notebook of DD2404's Project

Libo Xu

2018.12.23

I had read the paper written by William Stafford Noble once before I started to do the first lab. However, I didn't apply much of what I learned from the paper to the labs because the workload in each lab is small, compared to that in the project. So today I read this paper carefully before I started to finish the project. Besides, I decided to choose 'Reducing noise in protein multialignments' as my project of this course.

2018.12.28

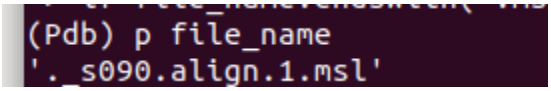
Today I began to write the code for the project. The whole structure of the code is that read the files, and then 'filter' the alignments and finally use FastPhylo and Dendrpy to assess the results of filtering.

Firstly, I decided to finish the noise-removing part and take one file in the test data to check whether my code is correct, which means I wouldn't care about the how to enter each directory to get all files in today's work.

For this part, I wrote 3 functions, which are used to read the input file, check the noisy columns and remove the noisy columns. Considered the error control, I set three error check parts in the code, which are empty files, wrong format of content and columns that are all noisy in a file.

2018.12.30

Yesterday I ran my code on the single input files and it worked well. Today I was going to refine the code to reduce the noise of all given files. In my code, the input is the name of the main folder which is 'appbio11' and the output is in a new folder called 'result'. However, when I ran the code errors occurred. So I used 'pdb' to debug the code and found that there were files that didn't display in the folder.



So I added a condition when recognized the file name to remove the file like the above and then I got the correct results stored in the new directory.

2018.1.2

Today I read the documentations of Fastphylo and Dendropy. I downloaded them successfully to my own computer and found the commands and functions I needed for analyzing the original data and filtering results.

fastprot

fastprot estimates the evolutionary distance between aligned protein sequences. It implements two methods for calculating the distance between protein sequences, the maximum likelihood of a distance and the expected distance

fnj

fnj implements the algorithm Fast Neighbor Joining.

Neighbor Joining (NJ) is a so-called distance-based method that, thanks to its good accuracy and speed, has been embraced by the phylogeny community. It takes the distances between n taxa and produces in Θ(n3) time a phylogenetic tree, i.e., a tree which aims to describe the evolutionary history of the taxa. In addition to performing well in practice, the NJ algorithm has optimal reconstruction radius.

DendroPy Phylogenetic Computing Library

DendroPy is a Python library for phylogenetic computing. It provides classes and functions for the simulation, processing, and manipulation of phylogenetic trees and character matrices, and supports the reading and writing of phylogenetic data in a range of formats.

Symmetric distance

The Robinson–Foulds metric is a way to measure the distance between unrooted phylogenetic trees. The Robinson–Foulds metric is also known as the symmetric difference metric.

The unweighted Robinson-Foulds distance (often referred to as just the Robinson-Foulds distance) is given by the `dendropy.calculate.treecompare.symmetric_difference` function:

```
import dendropy
from dendropy.calculate import treecompare

s1 = "(a,(b,(c,d)));"
s2 = "(a,(d,(b,c)));"

# establish common taxon namespace
tns = dendropy.TaxonNamespace()

# ensure all trees loaded use common namespace
tree1 = dendropy.Tree.get(
        data=s1,
        schema='newick',
        taxon_namespace=tns)
tree2 = dendropy.Tree.get(
        data=s2,
        schema='newick',
        taxon_namespace=tns)

## Unweighted Robinson-Foulds distance
print(treecompare.symmetric_difference(tree1, tree2))
```

2019.1.3

Today I began to write the code for analyzing the test data and filtering data. Firstly, I wrote a function named ‘evaluate’, which took a file as input and output the symmetric difference between the file and the reference tree. Then I wrote a ‘store’ function to store the symmetric difference of a sub-directory to a empty list. Hence, in the main function I applied the ‘store’ function to 2 directories which shared the same name. In this way, for example, I got the symmetric difference of ‘asymmetric_0.5’ including original alignment and the noise-reduced alignment in two list. However, when I used the difference between these 2 values to calculate the proportion of the cases where the symmetric difference of the tree after noise removal is smaller than that of the original tree, the result of proportion is 0.00%. By using pdb, I found that the ‘**os.listdir()**’ didn’t arrange the file names in the order of the file name, such as the number in the name,

which meant the differences I got were not correct because they were not calculated by subtracting two corresponding values.

The wrong code and running result were as follows:

```
def store(path):

    #Create a empty list to store the symmetric difference between input tree and reference tree.
    sd_list = []

    # Find the reference tree.
    for filename in os.listdir(path):
        if filename.endswith('.tree') and filename[0] != '.':
            ref_path = path + '/' + filename

    # Compare each files in the sub-directory and store the results in a list.
    for filename in os.listdir(path):
        if filename.endswith('.msl') and filename[0] != '.':
            detail_path = path + '/' + filename
            sd_list.append(evaluate(ref_path,detail_path))

    return sd_list
```

```
# The input should be the name of sub-directorys in the 'result' directory, like 'asymmetric_0.5'
sub_name = sys.argv[1]

path1 = '/home/xlbbbbbb/project/appbio11' + '/' + sub_name
path2 = '/home/xlbbbbbb/project/result' + '/' + sub_name

# The symmetric difference between orginal alignment tree and reference tree.
original_sd = store(path1)
# The symmetric difference between noise-reduced tree and reference tree.
reduced_sd = store(path2)
#pdb.set_trace()
# Create a empty list to store the difference between the 2 values above
diff = []
for i in range(0,300):
    diff.append(original_sd[i] - reduced_sd[i])
```

```
xlbbbbbb@xlbbbbbb-virtual-machine:~/project$ python data.py asymmetric_0.5
('asymmetric_0.5', 'original:', 1, 'reduced:', 1)
Proportion:0.00%
```

So, I deleted the ‘store’ function and made it into the main function, but the result of proportion was still 0.00%. The modified code was as follows:

```
# Create 3 empty lists to store the symmetric difference between the original alignment tree and reference tree,
# the symmetric difference between noise-reduced tree and reference tree,
# and the difference between these 2 values.
original_sd = []
reduced_sd = []
diff = []

# Find the reference tree.
for filename in os.listdir(path1):
    if filename.endswith('.tree') and filename[0] != '.':
        ref_path = path1 + '/' + filename
#pdb.set_trace()
# Compare each files in the sub-directory and store the results in the corresponding list.
for filename in os.listdir(path1):
    if filename.endswith('.msl') and filename[0] != '.':
        detail_path1 = path1 + '/' + filename
        original = evaluate(ref_path,detail_path1)
        original_sd.append(original)
        for _filename in os.listdir(path2):
            if _filename == 'unnoisy_' + str(filename):
                detail_path2 = path2 + '/' + _filename
                reduced = evaluate(ref_path,detail_path2)
                reduced_sd.append(reduced)
                diff.append(original - reduced)
```

Then I found out that it was because the difference between python 2.7 and python 3.7 on the division operation that caused the mistake. In my Ubuntu system, I used python

2.7 and the result of (1/3) would be 0. If I wanted to get the correct result, (float(1)/3) was needed. The result of **(float(1)/3)** would be 0.333333333333. Just as follows.

```
print('%.2f%%'%(33/300*100))
print('%.2f%%'%(float(33)/300*100))
```

```
0.00%
11.00%
```

Then I got the correct analyzing data.

2019.1.4

Yesterday I thought I got the right data, but today when I was going through my code I found that I made the same mistake in the noise checking code that I didn't add 'float; when doing division. So, the filtering result I got before was definitely inaccurate.

```
# Check the column according to the 3 conditions.
if indels/len(column) > 0.5: # Condition 1
    noisy = True
if aa_unique/aa_total >= 0.5: # Condition 2
    noisy = True
if aa_morethan2 == 0: #Condition 3
    noisy = True
```

```
# Check the column according to the 3 conditions.
if float(indels)/len(column) > 0.5: # Condition 1
    noisy = True
if float(aa_unique)/aa_total >= 0.5: # Condition 2
    noisy = True
if aa_morethan2 == 0: #Condition 3
    noisy = True
```

So, I modified the code and got the right data.

```
xlbbbbbb@xlbbbbbb-virtual-machine:~/project$ python analyse.py asymmetric_0.5
asymmetric_0.5 :  original:1 reduced:1
Average of difference: 0.07
Proportion: 15.33%
xlbbbbbb@xlbbbbbb-virtual-machine:~/project$ python analyse.py asymmetric_1.0
asymmetric_1.0 :  original:0 reduced:0
Average of difference: 0.07
Proportion: 19.00%
xlbbbbbb@xlbbbbbb-virtual-machine:~/project$ python analyse.py asymmetric_2.0
asymmetric_2.0 :  original:0 reduced:0
Average of difference: 0.39
Proportion: 31.00%
xlbbbbbb@xlbbbbbb-virtual-machine:~/project$ python analyse.py symmetric_0.5
symmetric_0.5 :  original:23 reduced:21
Average of difference: 0.07
Proportion: 15.33%
xlbbbbbb@xlbbbbbb-virtual-machine:~/project$ python analyse.py symmetric_1.0
symmetric_1.0 :  original:16 reduced:20
Average of difference: 0.15
Proportion: 20.33%
xlbbbbbb@xlbbbbbb-virtual-machine:~/project$ python analyse.py symmetric_2.0
symmetric_2.0 :  original:4 reduced:8
Average of difference: 0.37
Proportion: 31.67%
```

2019.1.5

Today I began to write my report.

2019.1.7

Report finished.