score_analysis

December 10, 2019

[106]:

```python
import os
from IPython.display import display, Image
import pandas as pd
import numpy as np
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt
from matplotlib import colors
from matplotlib.ticker import PercentFormatter
from scipy.stats import linregress
import math
from functools import reduce
import matplotlib
```

[2]:

```python
import argparse
from Bio import SeqIO, Entrez, pairwise2
Entrez.email = 'hongyingsun1101@gmail.com'
from Bio.SeqRecord import SeqRecord
import re, time
import os, sys, glob
import random
import uuid
# from skbio.tree import TreeNode
# from skbio import read
# from skbio.stats.distance import DistanceMatrix
# from skbio.stats.distance import DissimilarityMatrix

from scipy import stats
from ast import literal_eval
import sqlite3
```

[3]:

```python
class displayFancy(object):
    """Display HTML representation of multiple objects"""
    template = """<div style="float: left; padding: 10px;">
    <p style='font-family:"Courier New", Courier, monospace'>{0}</p>{1}
    </div>"""
    def __init__(self, *args):
        self.args = args

    def _repr_html_(self):
        return '\n'.join(self.template.format(a, eval(a)._repr_html_())
                         for a in self.args)

    def __repr__(self):
        return '\n\n'.join(a + '\n' + repr(eval(a))
                           for a in self.args)
```

[4]:

```
""" set up fonts here before importing matplotlib.pylab """
parms = {'font.family': 'serif', 'font.serif': 'Palatino', 'svg.fonttype':␣
 ↪'none'}
plt.rcParams.update(parms)
sns.set(context='talk', style='darkgrid', palette='deep', font='sans-serif')
```

[5]:

```
mock_seqtab=pd.read_csv("CC11.map.SeqTable.csv",index_col=0)
df_sv_list_names=['mock','rdp_10398','rdp_5224','rdp_1017','rdp_92','rdp_12']
taxaFiles=["AbundanceOfTaxIdInSamples_primaryTaxid.csv"]+[d+␣
 ↪"_"+"oneRankEachSV_keepBest.csv" for d in df_sv_list_names[1:]]
taxaDB="taxonomy.db"
nDlists=len(df_sv_list_names)
```

[6]:

```
#this function creats a list of files containing mock and the 5 rdp data in the␣
 ↪same list with the same format.
def prepareFiles():
    df_sv_list=[]
    for i, f in enumerate(taxaFiles):
        if i==0: #mock
            mock=mock_seqtab.copy()
            sample_ids=mock['community'].unique()
            sample_ids=['CC11CM'+str(i) for i in range(sample_ids.shape[0])]
            #not all tax_ids in the mock are primary
            translate={415850:1463164,195041:45634,592977:1680, 796939:796937,␣
 ↪41791:126333}
            mock.ncbi_tax_id.replace(translate, inplace=True)
            mock.rename(columns={'sourceSeq':'colind','organism':
 ↪'tax_name','ncbi_tax_id':'tax_id'}, inplace=True)
            sv_ids = mock.colind.unique()
            temp=pd.DataFrame(index=sv_ids,columns=['tax_id']+sample_ids)
            for s in sample_ids:
                mock_s = mock[mock.community==s]
                mock_s.set_index('colind', inplace=True)
                temp.loc[mock_s.index, 'tax_id']=mock_s['tax_id']
                temp.loc[mock_s.index, s]=mock_s['multiplicity']
            temp=temp.fillna(0)

        else: #analyzed using dada2/pplacer/RDP
```

```
            temp=pd.DataFrame(index=sv_ids,columns=['tax_id']+sample_ids)
            temp1=pd.read_csv(f)
            #drop rank, taxa_name and colind (SV index)
            temp1 = temp1.loc[:,['colind','tax_id']+sample_ids]
            temp1.set_index('colind',inplace=True)
            temp.loc[temp1.index,'tax_id']=temp1['tax_id']
            temp.loc[temp1.index,sample_ids]=temp1[sample_ids]
            temp=temp.fillna(0)

        ##very strange tax_id for the mock is not duplicated but when I set the␣
 ↪index as tax_id for the mock the index and the mock becomes duplicated!
        df_sv_list.append(temp)
        #print(temp.head())
    return df_sv_list
# generates the file list which has the mock data and the 5 data generated from␣
 ↪pplacer.
df_sv_list = prepareFiles()

dircs=[i+"/" for i in df_sv_list_names[1:]]
```

[7]:

```
def getUniqueSet(alltaxids):
    out=set(alltaxids[0])
    for l in alltaxids[1:]:
        out=out.union(l)
    return out
allsv=[df_sv_list[i].index.astype(str) for i in range(nDlists)]
allt=[df_sv_list[i].tax_id.astype(str) for i in range(nDlists)]
alls=[df_sv_list[i].columns.astype(str) for i in range(nDlists)]
alltaxa=getUniqueSet(allt)
allsvs=getUniqueSet(allsv)
allsamples=getUniqueSet(alls)
```

[ ]:

```

```

[8]:

```
test_df =df_sv_list[0]
```

[9]:

```
all_index = ['mock','rdp_10398','rdp_5224','rdp_1017','rdp_92','rdp_12']
# merged_df_all = pd.concat([pd.DataFrame('df_sv_list[0]'),pd.
 ↪DataFrame('df_sv_list[1]')])
# merged_df_all.head()
pd.concat([df_sv_list[0],df_sv_list[1]], axis=1).head()
```

[9]:

```
                  tax_id  CC11CM0  CC11CM1  CC11CM2  CC11CM3  CC11CM4  CC11CM5  \
AB036759.1.1480  113287        9       11        0        0        0        0
AB253730.1.1456  376804     4656        0        0        0        0        0
AB253731.1.1463  376805     1579        0        0        0        0        0
AB298910.1.1471    1736     1066        0        0        0        0        0
AB510708.1.1476   46506      684        0        0        0        0        0
```

```
                 CC11CM6  CC11CM7  CC11CM8  ...  CC11CM90  CC11CM91  CC11CM92  \
AB036759.1.1480        0        0        0  ...       0.0       0.0       0.0
AB253730.1.1456        0        0        0  ...       0.0       0.0       0.0
AB253731.1.1463        0        0        0  ...       0.0       0.0       0.0
AB298910.1.1471        0        0        0  ...       0.0       0.0       0.0
AB510708.1.1476        0        0        0  ...       0.0       0.0       0.0

                 CC11CM93  CC11CM94  CC11CM95  CC11CM96  CC11CM97  CC11CM98  \
AB036759.1.1480       0.0       0.0       0.0       0.0       0.0       0.0
AB253730.1.1456       0.0       0.0       0.0       0.0       0.0       0.0
AB253731.1.1463       0.0       0.0       0.0       0.0       0.0       0.0
AB298910.1.1471       0.0       0.0       0.0       0.0       0.0       0.0
AB510708.1.1476       0.0       0.0       0.0       0.0       0.0       0.0

                 CC11CM99
AB036759.1.1480       0.0
AB253730.1.1456       0.0
AB253731.1.1463       0.0
AB298910.1.1471       0.0
AB510708.1.1476       0.0

[5 rows x 202 columns]
```

[10]:

```
def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by the db_file
    :param db_file: database file
    :return: Connection object or None
    """
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except ConnectionError as e:
        print(e)

    return None
```

[11]:

```
# generate tables for analysis.
mock=df_sv_list[0]
df_merge=pd.DataFrame(index=list(allsvs))
df_merge.index.name="sv_id"
mock_tab1=df_merge.merge(mock,how='left', left_index=True, right_index=True)
mock_tab1=mock_tab1.fillna(0)
ref_tabs={}
def generateFiles():
    for i, df in enumerate(df_sv_list[1:],1):
        ana_tab2=df_merge.merge(df, how="left", left_index=True,␣
 ↪right_index=True)
```

```
        ana_tab2=ana_tab2.fillna(0)
        ref_tabs[df_sv_list_names[i]]=ana_tab2
    return ref_tabs
ref_tabs = generateFiles()
```

[12]:

```
ref_tabs['rdp_10398'].head()
```

[12]:

```
                          tax_id  CC11CM0  CC11CM1  CC11CM2  CC11CM3  CC11CM4  \
sv_id
NR_044400.1                29465      0.0      0.0      0.0    184.0      0.0
CP001071.320473.321977    239934      0.0      0.0      0.0      0.0      0.0
JHYB01000010.85.1425        2147      0.0      0.0      0.0      0.0      0.0
ACIF01000047.49.1542         848      0.0      0.0      0.0      0.0      0.0
AGXH01000076.81191.82710     816      0.0      0.0      0.0      0.0      0.0

                          CC11CM5  CC11CM6  CC11CM7  CC11CM8  ...  CC11CM90  \
sv_id                                                          ...
NR_044400.1                   0.0      0.0      0.0      0.0  ...       0.0
CP001071.320473.321977        0.0      0.0      0.0      0.0  ...       0.0
JHYB01000010.85.1425          0.0      0.0      0.0      0.0  ...       0.0
ACIF01000047.49.1542          0.0      0.0      0.0      0.0  ...       0.0
AGXH01000076.81191.82710      0.0      0.0      0.0      0.0  ...       0.0

                          CC11CM91  CC11CM92  CC11CM93  CC11CM94  CC11CM95  \
sv_id
NR_044400.1                    0.0       0.0       0.0       0.0       0.0
CP001071.320473.321977         0.0       0.0       0.0       0.0       0.0
JHYB01000010.85.1425          15.0       0.0       0.0       0.0       0.0
ACIF01000047.49.1542           0.0       0.0       0.0       0.0       0.0
AGXH01000076.81191.82710       0.0       0.0       0.0       0.0       0.0

                          CC11CM96  CC11CM97  CC11CM98  CC11CM99
sv_id
NR_044400.1                    0.0     115.0       0.0       0.0
CP001071.320473.321977         0.0       0.0       0.0       0.0
JHYB01000010.85.1425           0.0       0.0       0.0       0.0
ACIF01000047.49.1542           0.0       0.0       0.0       0.0
AGXH01000076.81191.82710       0.0       0.0       0.0       0.0

[5 rows x 101 columns]
```

[13]:

```
df_sv_list[0].head()
```

[13]:

```
                  tax_id  CC11CM0  CC11CM1  CC11CM2  CC11CM3  CC11CM4  CC11CM5  \
AB036759.1.1480   113287        9       11        0        0        0        0
AB253730.1.1456   376804     4656        0        0        0        0        0
AB253731.1.1463   376805     1579        0        0        0        0        0
AB298910.1.1471     1736     1066        0        0        0        0        0
AB510708.1.1476    46506      684        0        0        0        0        0
```

```
                 CC11CM6  CC11CM7  CC11CM8  ...  CC11CM90  CC11CM91  CC11CM92  \
AB036759.1.1480        0        0        0  ...         0         0         0
AB253730.1.1456        0        0        0  ...         0         0         0
AB253731.1.1463        0        0        0  ...         0         0         0
AB298910.1.1471        0        0        0  ...         0         0         0
AB510708.1.1476        0        0        0  ...         0         0         0

                 CC11CM93  CC11CM94  CC11CM95  CC11CM96  CC11CM97  CC11CM98  \
AB036759.1.1480         0         0         0         0         0         0
AB253730.1.1456         0         0         0         0         0         0
AB253731.1.1463         0         0         0         0         0         0
AB298910.1.1471         0         0         0         0         0         0
AB510708.1.1476         0         0         0         0         0         0

                 CC11CM99
AB036759.1.1480         0
AB253730.1.1456         0
AB253731.1.1463         0
AB298910.1.1471         0
AB510708.1.1476         0

[5 rows x 101 columns]
```

[14]:
```python
rdp_10398_predicted = ref_tabs['rdp_10398']
rdp_5224_predicted = ref_tabs['rdp_5224']
rdp_10398_predicted.head()
```

[14]:
```
                          tax_id  CC11CM0  CC11CM1  CC11CM2  CC11CM3  CC11CM4  \
sv_id
NR_044400.1                29465      0.0      0.0      0.0    184.0      0.0
CP001071.320473.321977    239934      0.0      0.0      0.0      0.0      0.0
JHYB01000010.85.1425        2147      0.0      0.0      0.0      0.0      0.0
ACIF01000047.49.1542         848      0.0      0.0      0.0      0.0      0.0
AGXH01000076.81191.82710     816      0.0      0.0      0.0      0.0      0.0

                          CC11CM5  CC11CM6  CC11CM7  CC11CM8  ...  CC11CM90  \
sv_id                                                         ...
NR_044400.1                   0.0      0.0      0.0      0.0  ...       0.0
CP001071.320473.321977        0.0      0.0      0.0      0.0  ...       0.0
JHYB01000010.85.1425          0.0      0.0      0.0      0.0  ...       0.0
ACIF01000047.49.1542          0.0      0.0      0.0      0.0  ...       0.0
AGXH01000076.81191.82710      0.0      0.0      0.0      0.0  ...       0.0

                          CC11CM91  CC11CM92  CC11CM93  CC11CM94  CC11CM95  \
sv_id
NR_044400.1                    0.0       0.0       0.0       0.0       0.0
CP001071.320473.321977         0.0       0.0       0.0       0.0       0.0
```

```
JHYB01000010.85.1425         15.0       0.0       0.0       0.0       0.0
ACIF01000047.49.1542          0.0       0.0       0.0       0.0       0.0
AGXH01000076.81191.82710      0.0       0.0       0.0       0.0       0.0

                         CC11CM96  CC11CM97  CC11CM98  CC11CM99
sv_id
NR_044400.1                   0.0     115.0       0.0       0.0
CP001071.320473.321977        0.0       0.0       0.0       0.0
JHYB01000010.85.1425          0.0       0.0       0.0       0.0
ACIF01000047.49.1542          0.0       0.0       0.0       0.0
AGXH01000076.81191.82710      0.0       0.0       0.0       0.0

[5 rows x 101 columns]
```

[ ]:

```

```

[ ]:

```

```

[15]:

```
conn = create_connection(taxaDB)
```

[16]:

```
# read csv files
score_table = pd.read_csv('score_table2.csv', index_col=0)
adcl_table = pd.read_csv('adcl_bySV_allsamples.csv', index_col=0)
score_table.describe()
```

[16]:

```
         CC11CM0    CC11CM1    CC11CM2    CC11CM3    CC11CM4    CC11CM5  \
count  55.000000  73.000000  40.000000  59.000000  52.000000  61.000000
mean    4.618182   4.410959   3.250000   3.152542   2.230769   4.754098
std     8.910025   8.055081   5.776833   5.148865   1.352056   8.564765
min     0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
25%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
50%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
75%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
max    38.000000  38.000000  38.000000  38.000000   6.000000  38.000000

         CC11CM6    CC11CM7    CC11CM8    CC11CM9  ...   CC11CM90   CC11CM91  \
count  54.000000  67.000000  65.000000  57.000000  ...  48.000000  59.000000
mean    3.111111   4.089552   4.184615   3.473684  ...   3.583333   2.135593
std     5.265289   7.314849   7.405429   6.375470  ...   7.310247   1.332066
min     0.000000   0.000000   0.000000   0.000000  ...   0.000000   0.000000
25%     2.000000   2.000000   2.000000   2.000000  ...   2.000000   2.000000
50%     2.000000   2.000000   2.000000   2.000000  ...   2.000000   2.000000
75%     2.000000   2.000000   2.000000   2.000000  ...   2.000000   2.000000
max    38.000000  38.000000  38.000000  38.000000  ...  38.000000   6.000000

        CC11CM92   CC11CM93   CC11CM94   CC11CM95   CC11CM96   CC11CM97  \
count  53.000000  61.000000  53.000000  61.000000  50.000000  57.000000
mean    3.207547   3.475410   4.226415   4.163934   3.160000   3.824561
std     5.238021   6.130814   8.130289   7.644563   5.658153   6.636272
```

```
min     0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
25%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
50%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
75%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
max    38.000000  38.000000  38.000000  38.000000  38.000000  38.000000

         CC11CM98   CC11CM99
count   62.000000  56.000000
mean     3.806452   3.285714
std      6.216949   5.207387
min      0.000000   0.000000
25%      2.000000   2.000000
50%      2.000000   2.000000
75%      2.000000   2.000000
max     38.000000  38.000000

[8 rows x 100 columns]
```

0.1 Sensitivity Analylsis

[17]:

```
def percentageCorrect(mock_seqtab, taxaFiles,df_sv_list_names,␣
 ↪withMultiplicity=True):
    mock=mock_seqtab.copy()
    #not all tax_ids in the mock are primary
    translate={415850:1463164,195041:45634,592977:1680, 796939:796937, 41791:
 ↪126333}
    mock.ncbi_tax_id.replace(translate, inplace=True)
    mock.rename(columns={'sourceSeq':'colind'}, inplace=True)
    mock = mock[['community', 'colind','organism', 'ncbi_tax_id',␣
 ↪'multiplicity']]
    mock_gpbySample = mock.groupby('community', as_index=True)
    sample_ids=[list(mock_gpbySample)[i][0] for i in␣
 ↪range(len(list(mock_gpbySample)))]
    df_pcorrect=pd.DataFrame(index=sample_ids)
    for i, f in enumerate(taxaFiles[1:],1):
        temp = pd.read_csv(f, index_col=0) #multisample assignment
        for s in  sample_ids:
            merged=mock_gpbySample.get_group(s).merge(temp[['tax_id',␣
 ↪'tax_name', 'colind']+[s]], how='left', on='colind') #merge on SVs=sourceSeq
            merged.loc[:,"iscorrect"]=(merged['ncbi_tax_id'].
 ↪astype(str)==merged['tax_id'].astype(str)).astype(int)
            if withMultiplicity:
                df_pcorrect.
 ↪loc[s,df_sv_list_names[i]]=((merged['iscorrect']*merged['multiplicity'])/
 ↪merged['multiplicity'].sum()).sum()*100.0 #percentage correct
            else:
                df_pcorrect.loc[s,df_sv_list_names[i]]=(merged['iscorrect']).
 ↪mean()*100.0 #percentage correct
    return sample_ids, df_pcorrect

 sample_ids, df_pcorrect=percentageCorrect(mock_seqtab,␣
 ↪taxaFiles,df_sv_list_names)
 sample_ids_woMulti, df_pcorrect_woMulti=percentageCorrect(mock_seqtab,␣
 ↪taxaFiles,df_sv_list_names, withMultiplicity=False)
```

merged0=mock$_{g}pbySample.get_{g}roup(s0).merge(temp0[['tax_{i}d',' tax_{n}ame',' colind'] + [s0]], how =' left', on =' colind')merged0.loc[:, "iscorrect"] = (merged0['ncbi_{t}ax_{i}d'].astype(str) == merged0['tax_{i}d'].astype(str)).astype(int)$

merged0['iscorrect'].mean()

[18]:
```
df_pcorrect.mean()
```

[18]:
```
rdp_10398    3.763598
rdp_5224     3.446103
rdp_1017     2.619771
rdp_92       1.100746
rdp_12       0.000000
dtype: float64
```

[19]:
```
rdp_10398_predicted.head()
```

[19]:
```
                          tax_id  CC11CM0  CC11CM1  CC11CM2  CC11CM3  CC11CM4  \
sv_id
NR_044400.1                29465      0.0      0.0      0.0    184.0      0.0
CP001071.320473.321977    239934      0.0      0.0      0.0      0.0      0.0
JHYB01000010.85.1425        2147      0.0      0.0      0.0      0.0      0.0
ACIF01000047.49.1542         848      0.0      0.0      0.0      0.0      0.0
AGXH01000076.81191.82710     816      0.0      0.0      0.0      0.0      0.0

                          CC11CM5  CC11CM6  CC11CM7  CC11CM8  ...  CC11CM90  \
sv_id                                                         ...
NR_044400.1                   0.0      0.0      0.0      0.0  ...       0.0
CP001071.320473.321977        0.0      0.0      0.0      0.0  ...       0.0
JHYB01000010.85.1425          0.0      0.0      0.0      0.0  ...       0.0
ACIF01000047.49.1542          0.0      0.0      0.0      0.0  ...       0.0
AGXH01000076.81191.82710      0.0      0.0      0.0      0.0  ...       0.0

                          CC11CM91  CC11CM92  CC11CM93  CC11CM94  CC11CM95  \
sv_id
NR_044400.1                    0.0       0.0       0.0       0.0       0.0
CP001071.320473.321977         0.0       0.0       0.0       0.0       0.0
JHYB01000010.85.1425          15.0       0.0       0.0       0.0       0.0
ACIF01000047.49.1542           0.0       0.0       0.0       0.0       0.0
AGXH01000076.81191.82710       0.0       0.0       0.0       0.0       0.0

                          CC11CM96  CC11CM97  CC11CM98  CC11CM99
```

```
sv_id
NR_044400.1                    0.0     115.0       0.0       0.0
CP001071.320473.321977         0.0       0.0       0.0       0.0
JHYB01000010.85.1425           0.0       0.0       0.0       0.0
ACIF01000047.49.1542           0.0       0.0       0.0       0.0
AGXH01000076.81191.82710       0.0       0.0       0.0       0.0

[5 rows x 101 columns]
```

[20]:
```
mock=mock_seqtab.copy()
translate={415850:1463164,195041:45634,592977:1680, 796939:796937, 41791:126333}
mock.ncbi_tax_id.replace(translate, inplace=True)
mock.rename(columns={'sourceSeq':'colind'}, inplace=True)
mock = mock[['community', 'colind','organism', 'ncbi_tax_id', 'multiplicity']]
mock_gpbySample = mock.groupby('community', as_index=True)
# df_pcorrect=pd.DataFrame(index=sample_ids)
temp0 = pd.read_csv("rdp_10398_oneRankEachSV_keepBest.csv", index_col=0)
temp0.head(3)
```

[20]:
```
    rank  tax_id         tax_name             colind  CC11CM15  CC11CM49  \
0  genus     816      Bacteroides  AB050110.1.1425     718.0       NaN
1  genus     816      Bacteroides  AB260025.1.1492     812.0       NaN
2  genus    1678  Bifidobacterium  AB437350.1.1505      32.0       NaN

   CC11CM87  CC11CM64  CC11CM85  CC11CM97  ...  CC11CM21  CC11CM92  CC11CM29  \
0       NaN       NaN       NaN       NaN  ...       NaN       NaN       NaN
1       NaN       NaN       NaN       NaN  ...       NaN       NaN       NaN
2       NaN       NaN       NaN       NaN  ...       NaN       NaN       NaN

   CC11CM56  CC11CM52  CC11CM96  CC11CM47  CC11CM31  CC11CM40  CC11CM7
0       NaN       NaN       NaN       NaN       NaN       NaN      NaN
1       NaN       NaN       NaN       NaN       NaN       NaN      NaN
2       NaN       NaN       NaN       NaN       NaN       NaN      NaN

[3 rows x 104 columns]
```

[21]:
```
sample_ids=[list(mock_gpbySample)[i][0] for i in␣
 ↪range(len(list(mock_gpbySample)))]
```

[22]:
```
mock_seqtab.head()
```

[22]:
```
  community        sourceSeq                                       seqID  \
0   CC11CM0  AB036759.1.1480  CC11CM0SCR1e06de32f41c414aaa57f33949f4905c
1   CC11CM0  AB253730.1.1456  CC11CM0SCR3e39a5fc61b6420cac1c2dd465292aec
2   CC11CM0  AB253731.1.1463  CC11CM0SCR3823cad73fba408b8b4a7cda1eb5e493
3   CC11CM0  AB298910.1.1471  CC11CM0SCR6f14135fa1ba41f49b480f6973684fb2
4   CC11CM0  AB510708.1.1476  CC11CM0SCRc168f76390fb4e7c9f4809f8d0100c39

                         organism  ncbi_tax_id  multiplicity
```

```
0  Pseudoramibacter alactolyticus       113287             9
1           Bacteroides barnesiae       376804          4656
2        Bacteroides salanitronis       376805          1579
3             Eubacterium limosum         1736          1066
4           Bacteroides stercoris        46506           684
```

```
[23]: displayFancy('df_pcorrect.head()', 'df_pcorrect_woMulti.head()')
```

```
[23]: df_pcorrect.head()
                rdp_10398  rdp_5224  rdp_1017    rdp_92  rdp_12
      CC11CM0    5.765636  5.765636  3.159084  2.504251     0.0
      CC11CM1    0.363896  0.378621  0.000000  0.000000     0.0
      CC11CM10   7.533649  7.499137  5.743113  3.020768     0.0
      CC11CM11   0.623034  0.036293  0.036293  0.000000     0.0
      CC11CM12   5.770894  2.601227  2.601227  0.000000     0.0

      df_pcorrect_woMulti.head()
                rdp_10398  rdp_5224  rdp_1017    rdp_92  rdp_12
      CC11CM0    5.454545  5.454545  3.636364  1.754386     0.0
      CC11CM1    2.702703  4.054054  0.000000  0.000000     0.0
      CC11CM10  10.606061  9.090909  9.090909  3.030303     0.0
      CC11CM11   2.985075  1.492537  1.492537  0.000000     0.0
      CC11CM12   6.153846  3.076923  3.076923  0.000000     0.0
```

```
[24]: displayFancy('df_pcorrect.describe()','df_pcorrect_woMulti.describe()')
```

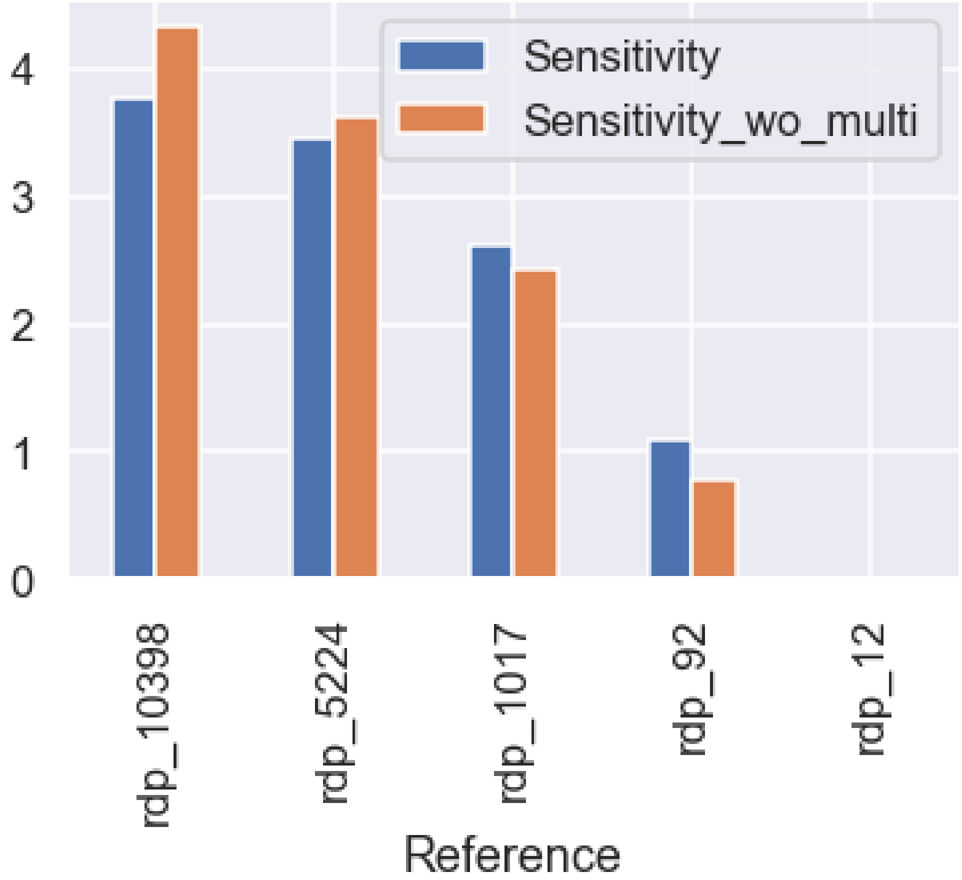
```
[24]: df_pcorrect.describe()
              rdp_10398    rdp_5224    rdp_1017      rdp_92  rdp_12
      count  100.000000  100.000000  100.000000  100.000000   100.0
      mean     3.763598    3.446103    2.619771    1.100746     0.0
      std      3.971108    3.957581    3.484531    2.428087     0.0
      min      0.000000    0.000000    0.000000    0.000000     0.0
      25%      0.740316    0.367785    0.029768    0.000000     0.0
      50%      3.058512    2.628039    1.331556    0.000000     0.0
      75%      5.000578    4.556366    3.813692    1.016204     0.0
      max     17.456369   17.222736   15.391099   12.863422     0.0

      df_pcorrect_woMulti.describe()
              rdp_10398    rdp_5224    rdp_1017      rdp_92  rdp_12
      count  100.000000  100.000000  100.000000  100.000000   100.0
      mean     4.319489    3.621532    2.422105    0.771246     0.0
      std      2.882590    2.573890    1.972394    1.100488     0.0
      min      0.000000    0.000000    0.000000    0.000000     0.0
      25%      2.062500    1.659836    1.509498    0.000000     0.0
      50%      4.546498    3.478524    1.851852    0.000000     0.0
      75%      6.202686    5.000000    3.389831    1.639344     0.0
      max     12.903226   12.903226    9.090909    4.000000     0.0
```

[25]:
```
df_sen=pd.DataFrame(df_pcorrect.mean().copy())
df_sen_woMulti=pd.DataFrame(df_pcorrect_woMulti.mean().copy())
```

[26]:
```
df_sen.columns = [ "Sensitivity"]
df_sen.index.name = 'Reference'
df_sen_woMulti.columns = [ "Sensitivity_wo_multi"]
df_sen_woMulti.index.name = 'Reference'
sen_table = pd.concat([df_sen, df_sen_woMulti], axis=1)
sen_table
```

[26]:
```
           Sensitivity  Sensitivity_wo_multi
Reference
rdp_10398     3.763598              4.319489
rdp_5224      3.446103              3.621532
rdp_1017      2.619771              2.422105
rdp_92        1.100746              0.771246
rdp_12        0.000000              0.000000
```

[27]:
```
sen_plot = sen_table.plot.bar()
```



0.2 Score Analysis

[28]:

```
display(score_table.head())
display(score_table.shape)
```

```
                          CC11CM0  CC11CM1  CC11CM2  CC11CM3  CC11CM4  \
sv_id
AB542765.1.1491               NaN      NaN      NaN      NaN      NaN
AGXW01000013.688.2204         NaN      NaN      NaN      NaN      NaN
HQ457030.1.1394               NaN      NaN      NaN      NaN      NaN
JHEF01000050.37729.39243      NaN      NaN      NaN      NaN      NaN
GU326240.1.1428               NaN      NaN      NaN      NaN      NaN

                          CC11CM5  CC11CM6  CC11CM7  CC11CM8  CC11CM9  ...  \
sv_id                                                                  ...
AB542765.1.1491               2.0      NaN      NaN      NaN      2.0  ...
AGXW01000013.688.2204         NaN      NaN      NaN      NaN      NaN  ...
HQ457030.1.1394               NaN      NaN      NaN      NaN      NaN  ...
JHEF01000050.37729.39243      NaN      NaN      NaN      NaN      NaN  ...
GU326240.1.1428               NaN      NaN      0.0      NaN      NaN  ...

                          CC11CM90  CC11CM91  CC11CM92  CC11CM93  CC11CM94  \
sv_id
AB542765.1.1491                NaN       2.0       NaN       NaN       NaN
AGXW01000013.688.2204          NaN       NaN       NaN       NaN       NaN
HQ457030.1.1394                NaN       NaN       NaN       NaN       NaN
JHEF01000050.37729.39243       NaN       NaN       NaN       NaN       NaN
GU326240.1.1428                NaN       NaN       NaN       NaN       NaN

                          CC11CM95  CC11CM96  CC11CM97  CC11CM98  CC11CM99
sv_id
AB542765.1.1491                NaN       NaN       NaN       NaN       NaN
AGXW01000013.688.2204          NaN       NaN       NaN       NaN       NaN
HQ457030.1.1394                NaN       NaN       NaN       NaN       2.0
JHEF01000050.37729.39243       NaN       NaN       NaN       NaN       NaN
GU326240.1.1428                NaN       NaN       NaN       NaN       NaN

[5 rows x 100 columns]
```

```
(1830, 100)
```

[29]:

```
score=score_table.CC11CM0
```

[30]:

```
score_table.describe()
```

[30]:

```
         CC11CM0    CC11CM1    CC11CM2    CC11CM3    CC11CM4    CC11CM5  \
count  55.000000  73.000000  40.000000  59.000000  52.000000  61.000000
```

```
mean    4.618182   4.410959   3.250000   3.152542   2.230769   4.754098
std     8.910025   8.055081   5.776833   5.148865   1.352056   8.564765
min     0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
25%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
50%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
75%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
max    38.000000  38.000000  38.000000  38.000000   6.000000  38.000000

         CC11CM6    CC11CM7    CC11CM8    CC11CM9  ...   CC11CM90   CC11CM91  \
count  54.000000  67.000000  65.000000  57.000000  ...  48.000000  59.000000
mean    3.111111   4.089552   4.184615   3.473684  ...   3.583333   2.135593
std     5.265289   7.314849   7.405429   6.375470  ...   7.310247   1.332066
min     0.000000   0.000000   0.000000   0.000000  ...   0.000000   0.000000
25%     2.000000   2.000000   2.000000   2.000000  ...   2.000000   2.000000
50%     2.000000   2.000000   2.000000   2.000000  ...   2.000000   2.000000
75%     2.000000   2.000000   2.000000   2.000000  ...   2.000000   2.000000
max    38.000000  38.000000  38.000000  38.000000  ...  38.000000   6.000000

        CC11CM92   CC11CM93   CC11CM94   CC11CM95   CC11CM96   CC11CM97  \
count  53.000000  61.000000  53.000000  61.000000  50.000000  57.000000
mean    3.207547   3.475410   4.226415   4.163934   3.160000   3.824561
std     5.238021   6.130814   8.130289   7.644563   5.658153   6.636272
min     0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
25%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
50%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
75%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
max    38.000000  38.000000  38.000000  38.000000  38.000000  38.000000

        CC11CM98   CC11CM99
count  62.000000  56.000000
mean    3.806452   3.285714
std     6.216949   5.207387
min     0.000000   0.000000
25%     2.000000   2.000000
50%     2.000000   2.000000
75%     2.000000   2.000000
max    38.000000  38.000000

[8 rows x 100 columns]
```

[31]:

```python
def get_tax_data(taxid):
    """once we have the taxid, we can fetch the record"""
    search = Entrez.efetch(id = taxid, db= "taxonomy", retmode = "xml")
    record = Entrez.read(search)

    return (record)
```

[32]:

```
def get_lineage_ids_fromdata(data, uprank):
    """once you have the data from get_tax_data fetch the lineage"""
    #uprank=['kingdom','phylum','class','order','family','genus','species']
    lineage_toparse = data[0]['LineageEx']
#     print("printing data[0]:", data[0]['LineageEx'])
#     print("hahha..............")
    lineage=dict()
    ids=dict()
    for l in lineage_toparse:
#         print("printing l:", l)
        for r in uprank:
            try:
                if l['Rank']==r:
                    lineage[r]=l['ScientificName']
                    ids[r]=l['TaxId']
            except:
                pass
    return lineage, ids
```

[33]:

```
# this function outputs all the names and ids of every level given a taxid and␣
 ↪a connection bulit from SQL database.
def get_lineage_ids(taxid, conn):
    """ This function gets the names and ids if all parents of the given id """

    query="SELECT nd.tax_id, nd.parent_id, nd.rank, na.tax_id, na.tax_name, na.
 ↪name_class from nodes nd inner join names na on nd.tax_id=na.tax_id where na.
 ↪name_class=='scientific name' AND na.tax_id==" + "'"+taxid+"'"

    df = pd.read_sql_query(query, conn)
    #print(df)

    df.columns=['tax_id', 'parent_id', 'rank', 'tax_id_drop', 'tax_name',␣
 ↪'name_class']
    df.drop("tax_id_drop",axis=1, inplace=True)
    rankorder=np.
 ↪array(['no_rank','superkingdom','phylum','class','order','family','genus','species'])[:
 ↪:-1]
    #print(df['rank'])
    if not df['rank'].iloc[0].strip() in rankorder:
        rankorder=np.append(rankorder,df['rank'].iloc[0])
    rank_ind=np.where(df['rank'].iloc[0]==rankorder)[0][0]
#     if len(df.tax_name.iloc[0].split(" "))>=2:
#         lineage={rankorder[rank_ind]:" ".join(df.tax_name.iloc[0].split(" ")[1:
 ↪])}
#     else:
#         lineage={rankorder[rank_ind]:df.tax_name.iloc[0]}
```

```
    lineage={rankorder[rank_ind]:df.tax_name.iloc[0]}
    ids={rankorder[rank_ind]: taxid}
    stop=False
    temp=df.copy()
    #print(lineage, ids)
    while not stop:
        parent_id=temp['parent_id'].iloc[0]
        if parent_id is None or parent_id=="" or parent_id=='0':
            stop=True
            break
        #print(parent_id)
        query="SELECT nd.tax_id, nd.parent_id, nd.rank, na.tax_id, na.tax_name,␣
↪na.name_class from nodes nd inner join names na on nd.tax_id=na.tax_id where␣
↪na.name_class=='scientific name' AND na.tax_id==" + "'"+parent_id+"'"
        temp = pd.read_sql_query(query, conn)
        temp.columns=['tax_id', 'parent_id', 'rank', 'tax_id_drop', 'tax_name',␣
↪'name_class']
        temp.drop("tax_id_drop",axis=1, inplace=True)
        lineage.update({temp['rank'].iloc[0]:temp['tax_name'].iloc[0]})

        ids.update({temp['rank'].iloc[0]: temp.tax_id.iloc[0]})



    return lineage, ids
```

[ ]:

```
```

[ ]:

```
```

[34]:

```
def get_parent(taxid, taxaDB):
    parent_rank=False
    parent_taxid=False
    if taxid=='2':
        return parent_taxid, parent_rank
    #create connection:
    conn = create_connection(taxaDB)
    ranks=np.
↪array(['superkingdom','phylum','subphylum','class','subclass','order','suborder',␣
↪'family','genus','species', 'subspecies'])
    try:
        lineage, ids = get_lineage_ids(str(taxid), conn)
    except:
        data_t=get_tax_data(str(taxid))
        lineage, ids = get_lineage_ids_fromdata(data_t, ranks)

    if type(ids)==dict:
        allparents = ids
    elif type(temp[0])==dict:
        allparents = ids[0]
    else:
        print("Caught exception")
        sys.exit(1)

    rankorder=ranks[::-1]
    #handling when the taxid is itself among parent ids
    for r,i in allparents.items():
        if i==taxid: #and r != "superkingdom":
            taxid_rank=r
            ind_=np.where(rankorder==taxid_rank)[0][0]
            parent_rank=rankorder[ind_+1]


    for i, r in enumerate(rankorder):
        #handling when the taxid is itself among parent ids
        if parent_rank and taxid_rank==r:
            continue

        try:
            parent_taxid=allparents[r]
            parent_rank=r
            break
        except:
            pass
    return parent_taxid, parent_rank
```

[ ]:

[ ]:

[35]:
```
def isSVInsample(svid, sample_id, table):
    """ table has indexes as the unique taxa and rows as samples
        if taxid exist in sample and has abundance>0 return True otherwise␣
 ↪return False"""
    assert(table.index.name=="sv_id"),"the table has to have sv_id as its index␣
 ↪and the index should be named sv_id"
    if np.any(table.index.isin([svid])):
        if table.loc[svid,sample_id]>0.0: #abundance is not zero
            return True
        else:
            return False
    else:
        return False
```

[36]:
```
def istaxIDEqual(svid, sample_id, table1, table2):
    """this function implicitly assumes that svid exists in both tables for the␣
 ↪sample_id and it checks if abundances is>0 in both
```

```
       before comparing their tax_ids"""
    if table1.loc[svid, sample_id]>0 and table2.loc[svid, sample_id]>0:
        tax1 = table1.loc[svid, 'tax_id']
        tax2 = table2.loc[svid, 'tax_id']
    elif table1.loc[svid, sample_id]>0:
        tax1 = table1.loc[svid, 'tax_id']
        tax2=np.nan
    elif table2.loc[svid, sample_id]>0:
        tax1=np.nan
        tax2 = table2.loc[svid, 'tax_id']
    else:
        tax1=np.nan
        tax2=np.nan

    if tax1 == tax2:
        return True, tax1, tax2
    else:
        return False, tax1, tax2
```

[37]:

```
def ranks_off(table1, table2, sv_id, sample_id, taxaDB):
    """A measure that calculates how many ranks is the SV in sample_id in␣
↪table2 is off from
       that in table1
       table1 and table2 are pandas dataframes with exactly the same indices␣
↪and columns
       columns as samples and indices as SVs"""
    ranks=['superkingdom', 'phylum', 'class', 'order', 'family', 'genus',␣
↪'species']
    isInTable1=isSVInsample(sv_id, sample_id, table1)
    isInTable2=isSVInsample(sv_id, sample_id, table2)
    #SV has abundance in both
    if isInTable1 and isInTable2:
        isTaxaEq, tax1, tax2 = istaxIDEqual(sv_id, sample_id, table1, table2)
        if isTaxaEq: #SV in both and the corresponding tax_id is equal
            return 0
        else: #SV in both and the corresponding tax_ids differ
            foundParent=False
            #try parents of tax1
            dummy_id = tax1
            output1=0
            while not foundParent and output1<7:
                print(dummy_id)
                parent_id, parent_rank =get_parent(dummy_id, taxaDB)
                if parent_id:
                    foundParent = parent_id==tax2   #istaxaInsample(parent_id,␣
↪sample_id, table2)
                    dummy_id = parent_id
                    output1+=1
                else:
                    foundParent=True

            #try parents of tax1
            dummy_id = tax1
            output2=0
            while not foundParent and output2<7:
                print(dummy_id)
                parent_id, parent_rank =get_parent(dummy_id, taxaDB)
                if parent_id:
                    foundParent = parent_id==tax2   #istaxaInsample(parent_id,␣
↪sample_id, table2)
                    dummy_id = parent_id
                    output2+=1
                else:
                    foundParent=True

            if output1>=output2:
                return output1
            else:
                return output2

    elif isInTable1: #not in table2
        isTaxaEq, tax1, tax2 = istaxIDEqual(sv_id, sample_id, table1, table2)
        parent_id, parent_rank =get_parent(tax1, taxaDB)
        if parent_rank in ranks:
            return ranks.index(parent_rank)+2 #plus 2 because the index starts␣
↪from 0 and it is the parent
        elif parent_rank == "subspecies":
            return 7
        elif parent_rank == "subphylum":
            return 2



    elif isInTable2: #not in table1
        isTaxaEq, tax1, tax2 = istaxIDEqual(sv_id, sample_id, table1, table2)
        parent_id, parent_rank =get_parent(tax2, taxaDB)
        if parent_rank in ranks:
            return ranks.index(parent_rank)+2 #plus 2 because the index starts␣
↪from 0 and it is the parent
        elif parent_rank == "subspecies":
            return 7
        elif parent_rank == "subphylum":
            return 2
    else: #not in both
```

```
        return 0
```

[38]:

```
def get_ranksoff(allsvs, df_sv_list, df_sv_list_names, taxaDB):
    """ a measure that uses table1 taken as the mock and table2 taken as full,␣
 ↪setA1, setB1, setC1
        one at a time. The taxa tables table1 and table2 would have the same␣
 ↪tax_ids aligned
        in the index and the same samples aligned in columns """
    mock=df_sv_list[0]
    df_merge=pd.DataFrame(index=list(allsvs))
    df_merge.index.name="sv_id"
    mock_tab1=df_merge.merge(mock,how='left', left_index=True, right_index=True)
    mock_tab1=mock_tab1.fillna(0)
    df_ranksoff_all=pd.DataFrame(index=mock_tab1.columns.values[1:])
    for i, df in enumerate(df_sv_list[1:],1):
        ana_tab2=df_merge.merge(df, how="left", left_index=True,␣
 ↪right_index=True)
        ana_tab2=ana_tab2.fillna(0)
        dummy_df=pd.DataFrame(index=mock.index,columns=mock.columns.values)
        for sample_id in mock.columns.values[1:]: #exclude tax_id column
            for sv_id in mock.index.values:
                ranksOff =  ranks_off(mock_tab1, ana_tab2, sv_id, sample_id,␣
 ↪taxaDB)
                RAM = mock.loc[sv_id, sample_id]/mock.loc[:, sample_id].sum()␣
 ↪#mock relative abundance for taxa in this sample
                dummy_df.loc[sv_id,sample_id]=RAM*ranksOff #ranksOff times␣
 ↪relative abundance in mock

            df_ranksoff_all.loc[sample_id,df_sv_list_names[i]]=dummy_df.loc[:
 ↪,sample_id].sum()
            #df_ranksoff_all.to_csv("ranksOff_bySample.csv")
    fig, ax = pl.subplots(figsize=(12,12))
    sbs.violinplot(data=df_ranksoff_all, ax=ax)
    ax.set_ylabel("Ranks off")
    #fig.savefig("ranksOff_dist_comp.png")

    return df_ranksoff_all
```

[ ]:

[39]:

```
def get_score(table1, table2, sv_id, sample_id, taxaDB, rankoff_species_genus =␣
 ↪2, rankoff_genus_family=4, rankoff_family_order=8, rankoff_order_class=16,␣
 ↪rankoff_class_phylum=32, accuracyoff=32, species_option=True):
    max_penalty=9999
    rankoff_score=0
    accuracyoff_score=0
    isInTable1 = isSVInsample(sv_id, sample_id, table1)
    isInTable2 = isSVInsample(sv_id, sample_id, table2)
    isTaxaEq, tax1, tax2 = istaxIDEqual(sv_id, sample_id, table1, table2)

    if isInTable1 and isInTable2:
#         print("printing tax1:", tax1)
        lineage1, ids1 = get_lineage_ids(str(tax1), conn)
#         print("printing tax2:", tax2)
        lineage2, ids2 = get_lineage_ids(str(tax2), conn)
        # table 1 the highest rank is species.
        if "species" in lineage1.keys(): # in table1,the prediction level is␣
 ↪species.
            if "species" in lineage2.keys():
                rankoff_score +=0
                if lineage1["species"] == lineage1["species"]:
                    accuracyoff_score +=0
                else: # both at species level but different species.
                    if species_option:
                        accuracyoff_score += accuracyoff
                    else:
                        if lineage1["genus"] == lineage2["genus"]:
                            accuracyoff_score += 0
                        else: # genus are different
                            accuracyoff_score += accuracyoff
            elif "genus" in lineage2.keys():  # the highest rank in table1 is␣
 ↪genus.
                if species_option:
                    rankoff_score += rankoff_species_genus
                else: # genus option
                    rankoff_score +=0
                    if lineage1["genus"] == lineage2["genus"]:
                        accuracyoff_score +=0
                    else:
                        accuracyoff_score += accuracyoff
            else:
                if ("family" in lineage2.keys() and "family" in lineage1.keys()␣
 ↪):
                    if lineage1["family"] == lineage2["family"]:
                        accuracyoff_score +=0
                    else:
                        accuracyoff_score += accuracyoff
                    if species_option:
                        offscore = rankoff_species_genus + rankoff_genus_family
                        rankoff_score += offscore
                    else:
                        offscore =  rankoff_genus_family
                        rankoff_score += offscore
                elif "order" in lineage2.keys():
                    if lineage1["order"] == lineage2["order"]:
                        accuracyoff_score +=0
                    else:
                        accuracyoff_score += accuracyoff
                    if species_option:
                        offscore = rankoff_species_genus +␣
↪rankoff_genus_family+ rankoff_family_order
                        rankoff_score += offscore
                    else:
                        offscore = rankoff_genus_family + rankoff_family_order
                        rankoff_score += offscore
                elif "class" in lineage2.keys():
                    if lineage1["class"] == lineage2["class"]:
                        accuracyoff_score +=0
                    else:
                        accuracyoff_score += accuracyoff
                    if species_option:
                        offscore = rankoff_species_genus +␣
↪rankoff_genus_family+ rankoff_family_order+ rankoff_order_class
                        rankoff_score += offscore
                    else:
                        offscore = rankoff_genus_family+ rankoff_family_order+␣
↪rankoff_order_class
                        rankoff_score += offscore
                elif "phylum" in lineage2.keys():
                    if lineage1["phylum"] == lineage2["phylum"]:
                        accuracyoff_score +=0
                    else:
                        accuracyoff_score += accuracyoff
                    if species_option:
                        offscore = rankoff_species_genus +␣
↪rankoff_genus_family+ rankoff_family_order+␣
↪rankoff_order_class+rankoff_class_phylum
                        rankoff_score += offscore
                    else:
                        offscore = rankoff_genus_family+ rankoff_family_order+␣
↪rankoff_order_class+rankoff_class_phylum
                        rankoff_score += offscore
        # table 1 the highest rank is genus.
        elif "genus" in lineage1.keys(): # in table1,the prediction level is␣
↪species.
            if "species" in lineage2.keys():
                if lineage1["genus"] == lineage2["genus"]:
                    accuracyoff_score +=0
                else:
                    accuracyoff_score += accuracyoff
                if species_option:
                    rankoff_score += rankoff_species_genus
                else:
                    rankoff_score += 0

            elif "genus" in lineage2.keys():  # the highest rank in table1 is␣
↪genus.
                rankoff_score +=0
                if lineage1["genus"] == lineage2["genus"]:
                    accuracyoff_score +=0
                else:
                    accuracyoff_score += accuracyoff
            else:
                if  ("family" in lineage2.keys() and "family" in lineage1.
↪keys() ):
                    if lineage1["family"] == lineage2["family"]:
                        accuracyoff_score +=0
                    else:
                        accuracyoff_score += accuracyoff
                    offscore =  rankoff_genus_family
                    rankoff_score += offscore

                elif "order" in lineage2.keys():
                    if lineage1["order"] == lineage2["order"]:
                        accuracyoff_score +=0
                    else:
                        accuracyoff_score += accuracyoff
                    offscore =  rankoff_genus_family+ rankoff_family_order
                    rankoff_score += offscore
                elif "class" in lineage2.keys():
                    if lineage1["class"] == lineage2["class"]:
                        accuracyoff_score +=0
                    else:
                        accuracyoff_score += accuracyoff
                    offscore = rankoff_genus_family+ rankoff_family_order+␣
↪rankoff_order_class
                    rankoff_score += offscore

                elif "phylum" in lineage2.keys():
                    if lineage1["phylum"] == lineage2["phylum"]:
                        accuracyoff_score +=0
                    else:
                        accuracyoff_score += accuracyoff
                    offscore = rankoff_genus_family+ rankoff_family_order+␣
↪rankoff_order_class+rankoff_class_phylum
                    rankoff_score += offscore
        # table 1 the highest rank is family.
        else: # for cases with highest rank of family, order, class, phylum.␣
↪these cases don't exist in mock data.
            accuracyoff_score=max_penalty;
            rankoff_score=max_penalty;
    else: # if not in both tables
        rankoff_score=np.nan
        accuracyoff_score=np.nan
    score=accuracyoff_score+rankoff_score
    return  rankoff_score, accuracyoff_score, score
```

[40]:

```
table1=mock_tab1
table2=ref_tabs["rdp_10398"]
sample_id="CC11CM0"
sv_id="AGXW01000015.688.2204"
```

[ ]:

```
```

[41]:

```
len(ref_tabs)
```

[41]:

```
5
```

[42]:

```
def get_community_score(table1, table2,  taxaDB):
    output_table=table2.copy()
    output_table=output_table.drop(columns="tax_id")
    community_list= table1.columns[1:]
    sv_list=table1.index
    for sample_id in community_list:
#         print("printing sample_id:", sample_id)
        for sv_id in sv_list:
#             print("printing sv_id:", sv_id)
            rankoff_score, accuracyoff_score, score =get_score(table1, table2,␣
↪sv_id, sample_id, taxaDB)
            output_table.loc[sv_id,sample_id]=score
    return output_table
```

[43]:

```
x="rdp_10398"
table2 = ref_tabs[x]
table2.index
```

[43]:

```
Index(['NR_044400.1', 'CP001071.320473.321977', 'JHYB01000010.85.1425',
       'ACIF01000047.49.1542', 'AGXH01000076.81191.82710',
       'ADLE01000001.1593.3112', 'AF050100.1.1541', 'CP007034.620467.622007',
       'NR_041508.1', 'JN600324.1.1530',
       ...
       'JN004270.1.1476', 'FJ823005.1.1445', 'CANO01000088.187.1714',
       'AJ585206.1.1357', 'NR_116458.1', 'ACFE01000007.338.1763',
       'EF406017.1.1470', 'KF052121.1.1420', 'JWIS01000029.8.1510',
       'JQJF01000002.263493.265031'],
      dtype='object', name='sv_id', length=1830)
```

[44]:
```
# table1=mock_tab1
# ref_list = {"rdp_10398",'rdp_5224','rdp_1017','rdp_92','rdp_12' }
# for ref_name in ref_list:
#     table2 = ref_tabs[ref_name]
#     score_table=  get_community_score(table1, table2,  taxaDB)
#     score_table.to_csv(str(ref_name)+"score.csv")
```

[45]:
```
community_list= table1.columns[1:]
table1.head()
```

[45]:
```
                           tax_id  CC11CM0  CC11CM1  CC11CM2  CC11CM3  \
sv_id
NR_044400.1                464322        0        0        0      184
CP001071.320473.321977     239934        0        0        0        0
JHYB01000010.85.1425      1408417        0        0        0        0
ACIF01000047.49.1542       620833        0        0        0        0
AGXH01000076.81191.82710   997875        0        0        0        0

                          CC11CM4  CC11CM5  CC11CM6  CC11CM7  CC11CM8  ...  \
sv_id                                                                  ...
NR_044400.1                     0        0        0        0        0  ...
CP001071.320473.321977          0        0        0        0        0  ...
JHYB01000010.85.1425            0        0        0        0        0  ...
ACIF01000047.49.1542            0        0        0        0        0  ...
AGXH01000076.81191.82710        0        0        0        0        0  ...

                          CC11CM90  CC11CM91  CC11CM92  CC11CM93  CC11CM94  \
sv_id
NR_044400.1                      0         0         0         0         0
CP001071.320473.321977           0         0         0         0         0
JHYB01000010.85.1425             0        15         0         0         0
ACIF01000047.49.1542             0         0         0         0         0
AGXH01000076.81191.82710         0         0         0         0         0

                          CC11CM95  CC11CM96  CC11CM97  CC11CM98  CC11CM99
sv_id
NR_044400.1                      0         0       115         0         0
CP001071.320473.321977           0         0         0         0         0
JHYB01000010.85.1425             0         0         0         0         0
ACIF01000047.49.1542             0         0         0         0         0
AGXH01000076.81191.82710         0         0         0         0         0

[5 rows x 101 columns]
```

[46]:
```
# score_table=  get_community_score(table1, table2,  taxaDB)
# score_table.to_csv("score_table.csv")
```

[47]:
```
score_rdp_10398 = pd.read_csv("score_table2.csv")
```

[48]:
```
score_rdp_10398.describe()
```

[48]:
```
         CC11CM0    CC11CM1    CC11CM2    CC11CM3    CC11CM4    CC11CM5  \
count  55.000000  73.000000  40.000000  59.000000  52.000000  61.000000
mean    4.618182   4.410959   3.250000   3.152542   2.230769   4.754098
std     8.910025   8.055081   5.776833   5.148865   1.352056   8.564765
min     0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
25%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
50%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
75%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
max    38.000000  38.000000  38.000000  38.000000   6.000000  38.000000

         CC11CM6    CC11CM7    CC11CM8    CC11CM9  ...   CC11CM90   CC11CM91  \
count  54.000000  67.000000  65.000000  57.000000  ...  48.000000  59.000000
mean    3.111111   4.089552   4.184615   3.473684  ...   3.583333   2.135593
std     5.265289   7.314849   7.405429   6.375470  ...   7.310247   1.332066
min     0.000000   0.000000   0.000000   0.000000  ...   0.000000   0.000000
25%     2.000000   2.000000   2.000000   2.000000  ...   2.000000   2.000000
50%     2.000000   2.000000   2.000000   2.000000  ...   2.000000   2.000000
75%     2.000000   2.000000   2.000000   2.000000  ...   2.000000   2.000000
max    38.000000  38.000000  38.000000  38.000000  ...  38.000000   6.000000

        CC11CM92   CC11CM93   CC11CM94   CC11CM95   CC11CM96   CC11CM97  \
count  53.000000  61.000000  53.000000  61.000000  50.000000  57.000000
mean    3.207547   3.475410   4.226415   4.163934   3.160000   3.824561
std     5.238021   6.130814   8.130289   7.644563   5.658153   6.636272
min     0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
25%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
50%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
75%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
max    38.000000  38.000000  38.000000  38.000000  38.000000  38.000000

        CC11CM98   CC11CM99
count  62.000000  56.000000
mean    3.806452   3.285714
std     6.216949   5.207387
min     0.000000   0.000000
25%     2.000000   2.000000
50%     2.000000   2.000000
75%     2.000000   2.000000
max    38.000000  38.000000

[8 rows x 100 columns]
```

[49]:

```
directory=[i+"/" for i in df_sv_list_names[1:]]
prefix='CC11CM'
```

[50]:

```
# def get_taxa_adcl(directory, prefix):
#     """ gather adcl and bestRankStats files under analysis directory and␣
 ↪return
#         a dataframe with adcl and tax_id/tax_name for each amplicon/SV that␣
 ↪has both (successfully placed SV)"""

#     samples = glob.glob(directory+"/analysis/"+prefix+"*.adcl.csv")

#     df = pd.DataFrame({'adcl':[],'achieved_rank':[]})
#     df_summary=pd.DataFrame(index=range(len(samples)))
#     for i,f in enumerate(samples):
#         s=os.path.basename(f).split(".adcl.csv")[0]
#         #adcl
#         adcl=pd.read_csv(directory+"analysis/"+s+".adcl.csv",header=None)
#         adcl.columns=['name','adcl','multiplicity']
#         #edpl
#         edpl=pd.read_csv(directory+"analysis/"+s+".edpl.csv",header=None)
#         edpl.columns=['name','edpl']
#         #pplacer stats: richness of placements and min_distal_length
#         pplacer_stats=pd.read_csv(directory+"analysis/"+s+"_pplaceStats.csv")
#         pplacer_stats.columns=['name','placeRichness','min_distL']
#         #best rank
#         bestRank=pd.read_csv(directory+"/analysis/"+s+"_bestRankStats.csv",␣
 ↪index_col=False)
#         #name,rank,tax_id,tax_name,likelihood,achieved_rank,ranks_off
#         bestRank.drop('index',inplace=True, axis=1)
#         print("number of reads without adcl=",bestRank.shape[0]-adcl.
 ↪shape[0])
#         df_summary.loc[i,'N_tot']=bestRank.shape[0]
#         df_summary.loc[i,'N_achieved']=bestRank[bestRank['ranks_off']==0].
 ↪shape[0]
#         df_summary.
 ↪loc[i,'N_achievedGenus']=bestRank[bestRank['ranks_off']==1].shape[0]
#         df_summary.
 ↪loc[i,'N_achievedFamily']=bestRank[bestRank['ranks_off']==2].shape[0]
#         df_summary.
 ↪loc[i,'N_achievedOrder']=bestRank[bestRank['ranks_off']==3].shape[0]
#         df_summary.loc[i,'N_off']=bestRank[bestRank['ranks_off']>0].shape[0]
#         df_summary.loc[i,'N_missed'] = bestRank['ranks_off'].isnull().sum()
#         df_summary.
 ↪loc[i,'Avrlikelihood_achieved']=bestRank[bestRank['ranks_off']==0]['likelihood'].
 ↪mean()
```

```
#         df_summary.
 ↪loc[i,'Avrlikelihood_achievedGenus']=bestRank[bestRank['ranks_off']==1]['likelihood'].
 ↪mean()
#         df_summary.
 ↪loc[i,'Avr_rankOff']=bestRank[bestRank['ranks_off']>0]['ranks_off'].mean()

#         merged=bestRank.merge(adcl, on='name', how='left')
#         merged = merged.merge(edpl, on='name', how='left')
#         merged = merged.merge(pplacer_stats, on='name', how='left')
#         merged =␣
 ↪merged[['name','tax_id','achieved_rank','adcl','edpl','placeRichness','min_distL']]
#         merged.loc[:,'runDir']=directory

#         df=df.append(merged)

#    ␣
 ↪ranks=['species','genus','family','order','class','subphylum','phylum','superkingdom']
#     values=[1,2,3,4,5,5.5,6,7]
#     fig,ax=pl.subplots(figsize=(12,12))
#     g= sbs.violinplot(x='achieved_rank',y='adcl', order=ranks, data=df,␣
 ↪ax=ax)
#     g.set_xticklabels(labels=ranks, rotation=20)
#     #pl.savefig(directory+"analysis/violinpl_adcl_verRank.png")
#     return df
# df = get_taxa_adcl(dircs, prefix)
```

0.3 Correlation between score and pplacer stats

[244]:

```
adcl = pd.read_csv("adcl_bySV_allsamples.csv", index_col=0)
edpl=pd.read_csv("edpl_bySV_allsamples.csv", index_col=0)
mindistl= pd.read_csv("mindistL_bySV_allsamples.csv", index_col=0)
prichness = pd.read_csv("prichness_bySV_allsamples.csv", index_col=0)
```

[ ]:

```
```

[52]:

```
displayFancy('adcl.describe()', 'edpl.describe()','mindistl.
 ↪describe()','prichness.describe()')
```

[52]:

```
adcl.describe()
          RDP_10398     RDP_5224     RDP_1017       RDP_92       RDP_12
count   5974.000000  5974.000000  5974.000000  5974.000000  5974.000000
mean       0.030870     0.067427     0.110561     0.140690     0.166655
std        0.076025     0.092905     0.106539     0.087717     0.087512
min        0.000001     0.000001     0.000001     0.000005     0.000008
25%        0.000006     0.000007     0.019634     0.068713     0.083255
50%        0.000008     0.024131     0.088957     0.130732     0.176666
75%        0.019636     0.099829     0.177335     0.198498     0.248669
```

```
max       0.452194     0.483433     0.435116     0.359080     0.297615

edpl.describe()
         RDP_10398     RDP_5224     RDP_1017       RDP_92       RDP_12
count  5974.000000  5974.000000  5974.000000  5974.000000  5974.000000
mean      0.019395     0.021035     0.031887     0.033674     0.022970
std       0.044893     0.038966     0.062647     0.062739     0.026978
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.000000     0.000000
50%       0.000617     0.003024     0.007771     0.016452     0.013817
75%       0.019254     0.028197     0.036618     0.035785     0.042923
max       0.361655     0.284314     0.840492     0.542972     0.126934

mindistl.describe()
          RDP_10398      RDP_5224      RDP_1017       RDP_92       RDP_12
count  5.974000e+03  5.974000e+03  5.974000e+03  5974.000000  5974.000000
mean   4.508446e-03  1.296932e-02  1.839854e-02     0.032928     0.057637
std    2.393452e-02  3.332695e-02  4.808566e-02     0.056049     0.089106
min    3.489204e-07  5.562750e-07  5.196400e-07     0.000005     0.000005
25%    5.418109e-06  5.673431e-06  5.887012e-06     0.000007     0.000005
50%    6.586377e-06  7.150903e-06  7.985791e-06     0.003853     0.000007
75%    8.450043e-06  6.398175e-03  1.254981e-02     0.031168     0.101755
max    2.984937e-01  2.462279e-01  3.345772e-01     0.311486     0.263259

prichness.describe()
         RDP_10398     RDP_5224     RDP_1017       RDP_92       RDP_12
count  5974.000000  5974.000000  5974.000000  5974.000000  5974.000000
mean      4.727653     5.068631     5.379645     5.478072     6.038165
std       5.465596     5.629558     4.954691     5.276621     4.627806
min       1.000000     1.000000     1.000000     1.000000     1.000000
25%       1.000000     1.000000     1.000000     1.000000     1.000000
50%       3.000000     3.000000     3.000000     3.000000     5.000000
75%       5.000000     7.000000     7.000000     7.000000    11.000000
max      20.000000    20.000000    20.000000    20.000000    17.000000
```

[214]:
```
score_table2 = pd.read_csv("score_table2.csv")
score_table2.describe()
```

[214]:
```
         CC11CM0    CC11CM1    CC11CM2    CC11CM3    CC11CM4    CC11CM5  \
count  55.000000  73.000000  40.000000  59.000000  52.000000  61.000000
mean    4.618182   4.410959   3.250000   3.152542   2.230769   4.754098
std     8.910025   8.055081   5.776833   5.148865   1.352056   8.564765
min     0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
25%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
50%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
75%     2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
max    38.000000  38.000000  38.000000  38.000000   6.000000  38.000000
```

```
          CC11CM6    CC11CM7    CC11CM8    CC11CM9  ...   CC11CM90   CC11CM91  \
count   54.000000  67.000000  65.000000  57.000000  ...  48.000000  59.000000
mean     3.111111   4.089552   4.184615   3.473684  ...   3.583333   2.135593
std      5.265289   7.314849   7.405429   6.375470  ...   7.310247   1.332066
min      0.000000   0.000000   0.000000   0.000000  ...   0.000000   0.000000
25%      2.000000   2.000000   2.000000   2.000000  ...   2.000000   2.000000
50%      2.000000   2.000000   2.000000   2.000000  ...   2.000000   2.000000
75%      2.000000   2.000000   2.000000   2.000000  ...   2.000000   2.000000
max     38.000000  38.000000  38.000000  38.000000  ...  38.000000   6.000000

         CC11CM92   CC11CM93   CC11CM94   CC11CM95   CC11CM96   CC11CM97  \
count   53.000000  61.000000  53.000000  61.000000  50.000000  57.000000
mean     3.207547   3.475410   4.226415   4.163934   3.160000   3.824561
std      5.238021   6.130814   8.130289   7.644563   5.658153   6.636272
min      0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
25%      2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
50%      2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
75%      2.000000   2.000000   2.000000   2.000000   2.000000   2.000000
max     38.000000  38.000000  38.000000  38.000000  38.000000  38.000000

         CC11CM98   CC11CM99
count   62.000000  56.000000
mean     3.806452   3.285714
std      6.216949   5.207387
min      0.000000   0.000000
25%      2.000000   2.000000
50%      2.000000   2.000000
75%      2.000000   2.000000
max     38.000000  38.000000

[8 rows x 100 columns]
```

[216]:

```
adcl.reset_index(inplace=True)
edpl.reset_index(inplace=True)
mindistl.reset_index(inplace=True)
prichness.reset_index(inplace=True)
```

[217]:

```
adcl.loc[:,'newIndex']=adcl['index'].apply(lambda r: r.split("_From")[0])
edpl.loc[:,'newIndex']=edpl['index'].apply(lambda r: r.split("_From")[0])
mindistl.loc[:,'newIndex']=mindistl['index'].apply(lambda r: r.
 ↪split("_From")[0])
prichness.loc[:,'newIndex']=prichness['index'].apply(lambda r: r.
 ↪split("_From")[0])
```

[ ]:

```

```

[218]:

```
adcl.head()
```

[218]:
```
   level_0                                              index  RDP_10398  \
0        0  CC11CM81SCReeb34a221dca4dc5b0ca403cc9ac9bd3_Fr...   0.000007
1        1  CC11CM99SCRce2d1fa684f340c68b6f3239c2de2521_Fr...   0.000009
2        2  CC11CM7SCR9e29ef94a3604006b11d4566a917e44b_Fro...   0.194120
3        3  CC11CM11SCR22fac7d980eb496c9258d2c043fb6ea0_Fr...   0.053160
4        4  CC11CM34SCR39303fc7db3e46a0be0bc45364d289ff_Fr...   0.000007

   RDP_5224  RDP_1017    RDP_92    RDP_12  \
0  0.137668  0.021522  0.155496  0.204297
1  0.000010  0.218740  0.198498  0.294593
2  0.097577  0.179467  0.287096  0.215959
3  0.149998  0.097881  0.112167  0.065455
4  0.000006  0.126795  0.186934  0.267457

                                      newIndex
0  CC11CM81SCReeb34a221dca4dc5b0ca403cc9ac9bd3
1  CC11CM99SCRce2d1fa684f340c68b6f3239c2de2521
2   CC11CM7SCR9e29ef94a3604006b11d4566a917e44b
3  CC11CM11SCR22fac7d980eb496c9258d2c043fb6ea0
4  CC11CM34SCR39303fc7db3e46a0be0bc45364d289ff
```

[219]:
```
adcl.set_index("newIndex", inplace=True)
edpl.set_index("newIndex", inplace=True)
mindistl.set_index("newIndex", inplace=True)
prichness.set_index("newIndex", inplace=True)
```

[245]:
```
displayFancy('adcl.describe()', 'edpl.describe()','mindistl.
 ↪describe()','prichness.describe()')
```

[245]:
```
adcl.describe()
         RDP_10398     RDP_5224     RDP_1017       RDP_92       RDP_12
count  5974.000000  5974.000000  5974.000000  5974.000000  5974.000000
mean      0.030870     0.067427     0.110561     0.140690     0.166655
std       0.076025     0.092905     0.106539     0.087717     0.087512
min       0.000001     0.000001     0.000001     0.000005     0.000008
25%       0.000006     0.000007     0.019634     0.068713     0.083255
50%       0.000008     0.024131     0.088957     0.130732     0.176666
75%       0.019636     0.099829     0.177335     0.198498     0.248669
max       0.452194     0.483433     0.435116     0.359080     0.297615

edpl.describe()
         RDP_10398     RDP_5224     RDP_1017       RDP_92       RDP_12
count  5974.000000  5974.000000  5974.000000  5974.000000  5974.000000
mean      0.019395     0.021035     0.031887     0.033674     0.022970
std       0.044893     0.038966     0.062647     0.062739     0.026978
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.000000     0.000000
50%       0.000617     0.003024     0.007771     0.016452     0.013817
```

```
75%      0.019254    0.028197    0.036618    0.035785    0.042923
max      0.361655    0.284314    0.840492    0.542972    0.126934

mindistl.describe()
            RDP_10398      RDP_5224      RDP_1017       RDP_92       RDP_12
count  5.974000e+03  5.974000e+03  5.974000e+03  5974.000000  5974.000000
mean   4.508446e-03  1.296932e-02  1.839854e-02     0.032928     0.057637
std    2.393452e-02  3.332695e-02  4.808566e-02     0.056049     0.089106
min    3.489204e-07  5.562750e-07  5.196400e-07     0.000005     0.000005
25%    5.418109e-06  5.673431e-06  5.887012e-06     0.000007     0.000005
50%    6.586377e-06  7.150903e-06  7.985791e-06     0.003853     0.000007
75%    8.450043e-06  6.398175e-03  1.254981e-02     0.031168     0.101755
max    2.984937e-01  2.462279e-01  3.345772e-01     0.311486     0.263259

prichness.describe()
          RDP_10398     RDP_5224     RDP_1017       RDP_92       RDP_12
count  5974.000000  5974.000000  5974.000000  5974.000000  5974.000000
mean      4.727653     5.068631     5.379645     5.478072     6.038165
std       5.465596     5.629558     4.954691     5.276621     4.627806
min       1.000000     1.000000     1.000000     1.000000     1.000000
25%       1.000000     1.000000     1.000000     1.000000     1.000000
50%       3.000000     3.000000     3.000000     3.000000     5.000000
75%       5.000000     7.000000     7.000000     7.000000    11.000000
max      20.000000    20.000000    20.000000    20.000000    17.000000
```

[221]:
```
adcl.drop_duplicates( keep=False, inplace=True)
edpl.drop_duplicates( keep=False, inplace=True)
mindistl.drop_duplicates( keep=False, inplace=True)
prichness.drop_duplicates( keep=False, inplace=True)
```

[222]:
```
sum(prichness.index.duplicated())
```

[222]:
```
0
```

[223]:
```
cc11 = pd.read_csv("CC11.map.SeqTable.csv", index_col=0 )
```

[224]:
```
cc11.head()
```

[224]:
```
  community        sourceSeq                                       seqID  \
0   CC11CM0  AB036759.1.1480  CC11CM0SCR1e06de32f41c414aaa57f33949f4905c
1   CC11CM0  AB253730.1.1456  CC11CM0SCR3e39a5fc61b6420cac1c2dd465292aec
2   CC11CM0  AB253731.1.1463  CC11CM0SCR3823cad73fba408b8b4a7cda1eb5e493
3   CC11CM0  AB298910.1.1471  CC11CM0SCR6f14135fa1ba41f49b480f6973684fb2
4   CC11CM0  AB510708.1.1476  CC11CM0SCRc168f76390fb4e7c9f4809f8d0100c39

                        organism  ncbi_tax_id  multiplicity
0 Pseudoramibacter alactolyticus       113287             9
1            Bacteroides barnesiae     376804          4656
2          Bacteroides salanitronis     376805          1579
3              Eubacterium limosum       1736          1066
```

```
4          Bacteroides stercoris        46506          684
```

[225]:

```
adcl_cc11=pd.merge(adcl, cc11, left_on='newIndex', right_on='seqID')
edpl_cc11=pd.merge(edpl, cc11, left_on='newIndex', right_on='seqID')
mindistl_cc11=pd.merge(mindistl, cc11, left_on='newIndex', right_on='seqID')
prichness_cc11=pd.merge(prichness, cc11, left_on='newIndex', right_on='seqID')
```

[226]:

```
adcl_cc11.sort_values(by=['community'], inplace=True)
edpl_cc11.sort_values(by=['community'], inplace=True)
mindistl_cc11.sort_values(by=['community'], inplace=True)
prichness_cc11.sort_values(by=['community'], inplace=True)
```

[246]:

```
prichness_cc11.describe()
```

[246]:

```
           level_0    RDP_10398     RDP_5224     RDP_1017       RDP_92  \
count  5974.000000  5974.000000  5974.000000  5974.000000  5974.000000
mean   2986.500000     4.727653     5.068631     5.379645     5.478072
std    1724.689586     5.465596     5.629558     4.954691     5.276621
min       0.000000     1.000000     1.000000     1.000000     1.000000
25%    1493.250000     1.000000     1.000000     1.000000     1.000000
50%    2986.500000     3.000000     3.000000     3.000000     3.000000
75%    4479.750000     5.000000     7.000000     7.000000     7.000000
max    5973.000000    20.000000    20.000000    20.000000    20.000000

            RDP_12   ncbi_tax_id  multiplicity
count  5974.000000  5.974000e+03   5974.000000
mean      6.038165  4.893355e+05    802.088718
std       4.627806  4.425526e+05   1251.463832
min       1.000000  5.460000e+02      1.000000
25%       1.000000  8.218825e+04     56.000000
50%       5.000000  3.870900e+05    322.000000
75%      11.000000  7.453680e+05   1021.750000
max      17.000000  1.577349e+06  18403.000000
```

[247]:

```
adcl_CC11CM0 = adcl_cc11.loc[adcl_cc11.community=='CC11CM0']
edpl_CC11CM0 =edpl_cc11.loc[edpl_cc11.community=='CC11CM0']
mindistl_CC11CM0 =mindistl_cc11.loc[mindistl_cc11.community=='CC11CM0']
prichness_CC11CM0 =prichness_cc11.loc[prichness_cc11.community=='CC11CM0']
```

[248]:

```
adcl_CC11CM0_score = pd.merge(adcl_CC11CM0,score_table2, left_on = 'sourceSeq',␣
 ↪right_on = 'sv_id')
edpl_CC11CM0_score = pd.merge(edpl_CC11CM0,score_table2, left_on = 'sourceSeq',␣
 ↪right_on = 'sv_id')
mindistl_CC11CM0_score = pd.merge(mindistl_CC11CM0,score_table2, left_on =␣
 ↪'sourceSeq', right_on = 'sv_id')
prichness_CC11CM0_score = pd.merge(prichness_CC11CM0,score_table2, left_on =␣
 ↪'sourceSeq', right_on = 'sv_id')
```

[249]:

```
adcl_CC11CM0_RDP_10398 = adcl_CC11CM0_score[['seqID', 'CC11CM0', 'RDP_10398']]
edpl_CC11CM0_RDP_10398 = edpl_CC11CM0_score[['seqID', 'CC11CM0', 'RDP_10398']]
```

```
mindistl_CC11CM0_RDP_10398 = mindistl_CC11CM0_score[['seqID', 'CC11CM0',␣
 ↪'RDP_10398']]
prichess_CC11CM0_RDP_10398 = prichness_CC11CM0_score[['seqID', 'CC11CM0',␣
 ↪'RDP_10398']]
```

[250]:
```
adcl_CC11CM0_RDP_10398.describe()
```

[250]:
```
         CC11CM0  RDP_10398
count  55.000000  55.000000
mean    4.618182   0.031360
std     8.910025   0.087752
min     0.000000   0.000001
25%     2.000000   0.000006
50%     2.000000   0.000008
75%     2.000000   0.000010
max    38.000000   0.452194
```

[ ]:
```

```

[ ]:
```

```

[ ]:
```

```

[ ]:
```

```

[ ]:
```

```

[251]:
```
edpl_CC11CM0_RDP_10398=edpl_CC11CM0_RDP_10398.rename(columns={"RDP_10398":
 ↪"RDP_10398_edpl", "CC11CM0":"CC11CM0_score"})
adcl_CC11CM0_RDP_10398=adcl_CC11CM0_RDP_10398.rename(columns={"RDP_10398":
 ↪"RDP_10398_adcl", "CC11CM0":"CC11CM0_score"})
mindistl_CC11CM0_RDP_10398=mindistl_CC11CM0_RDP_10398.
 ↪rename(columns={"RDP_10398":"RDP_10398_mindistl", "CC11CM0":"CC11CM0_score"})
prichess_CC11CM0_RDP_10398=prichess_CC11CM0_RDP_10398.
 ↪rename(columns={"RDP_10398":"RDP_10398_prichness", "CC11CM0":
 ↪"CC11CM0_score"})
```

[252]:
```
mindistl_CC11CM0_RDP_10398.head()
```

[252]:
```
                                        seqID  CC11CM0_score  \
0  CC11CM0SCR00cdec97c058446e83e0ef032e61806d            2.0
1  CC11CM0SCR35529da454f0497fa16e04841e8e1639           34.0
2  CC11CM0SCR1083e70ce28e4961b8298356c0d69000            2.0
3  CC11CM0SCR8898bf89e307445282cd2fb3acb183a0            2.0
4  CC11CM0SCRfc7f700b82e44da39405162a7e5b8b77            2.0

   RDP_10398_mindistl
0        7.189148e-06
1        5.645650e-07
2        6.238599e-06
3        5.146191e-06
```

```
4        5.645650e-07
```

[253]:

```
df_list_all =␣
↪[adcl_CC11CM0_RDP_10398,edpl_CC11CM0_RDP_10398,mindistl_CC11CM0_RDP_10398,prichness_CC11CM0
↪]
```

[254]:

```
df_stats_combined=reduce(lambda x, y: pd.merge(x, y, on = ['seqID',␣
↪'CC11CM0_score']), df_list_all)

df_stats_combined.head()
```

[254]:

```
                                        seqID  CC11CM0_score  RDP_10398_adcl  \
0  CC11CM0SCR00cdec97c058446e83e0ef032e61806d            2.0        0.000007
1  CC11CM0SCR35529da454f0497fa16e04841e8e1639           34.0        0.000008
2  CC11CM0SCR1083e70ce28e4961b8298356c0d69000            2.0        0.000006
3  CC11CM0SCR8898bf89e307445282cd2fb3acb183a0            2.0        0.000005
4  CC11CM0SCRfc7f700b82e44da39405162a7e5b8b77            2.0        0.000001

   RDP_10398_edpl  RDP_10398_mindistl  RDP_10398_prichness
0        0.000000        7.189148e-06                  1.0
1        0.086949        5.645650e-07                 20.0
2        0.000000        6.238599e-06                  1.0
3        0.010897        5.146191e-06                  3.0
4        0.090313        5.645650e-07                 20.0
```

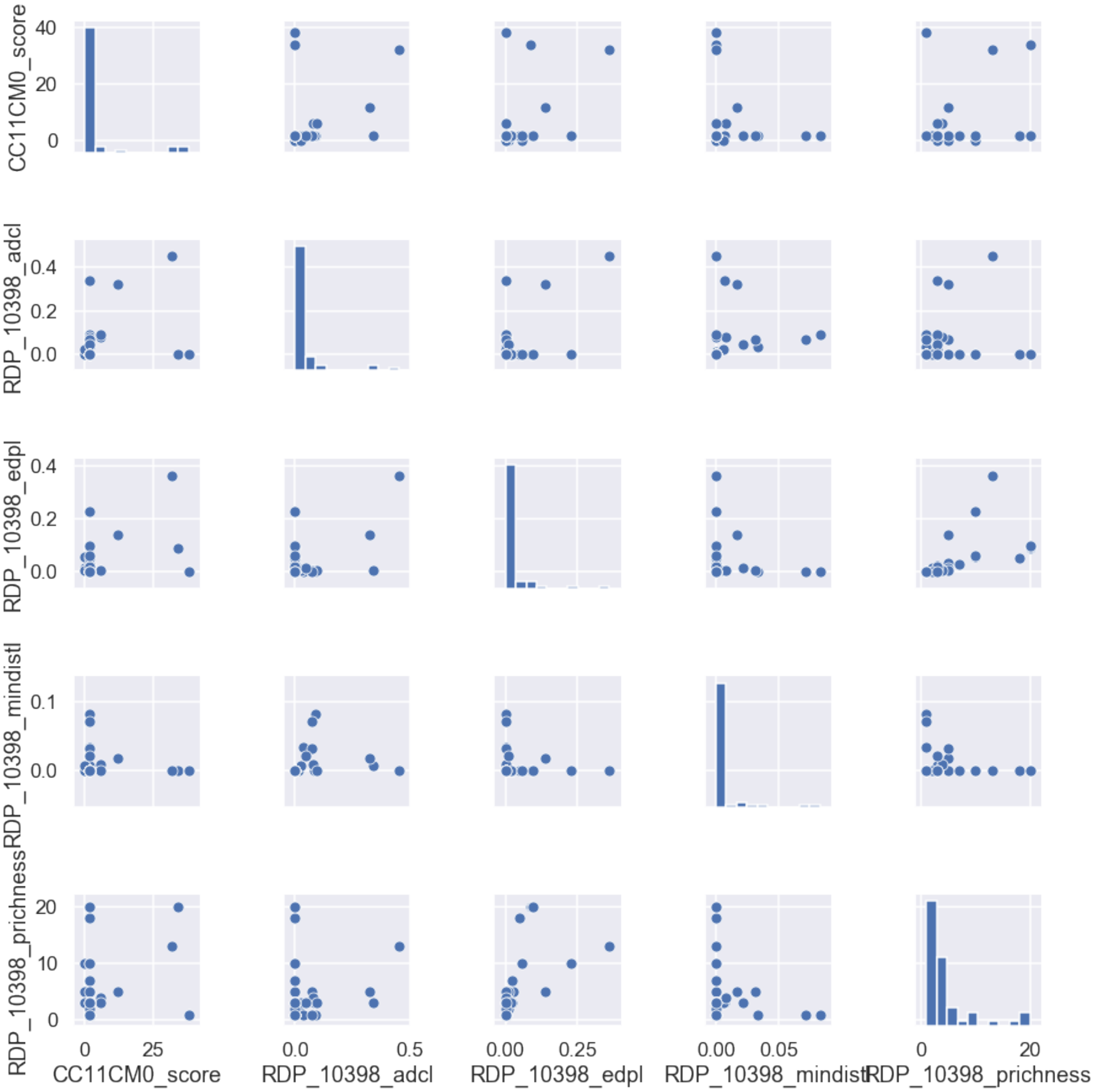
[ ]:

[255]:

```
sns.pairplot(df_stats_combined)
```

[255]:

```
<seaborn.axisgrid.PairGrid at 0x3217ecb0>
```



[261]:

```
df_stats_combined['RDP_10398_adcl_log'] = np.log10(df_stats_combined.
 ↪RDP_10398_adcl+0.0000000000000001)
df_stats_combined['RDP_10398_edpl_log'] = np.log10(df_stats_combined.
 ↪RDP_10398_edpl+0.0000000000000001)
df_stats_combined['CC11CM0_score_log'] = np.log10(df_stats_combined.
 ↪CC11CM0_score+0.0000000000000001)
df_stats_combined['RDP_10398_mindistl_log'] = np.log10(df_stats_combined.
 ↪RDP_10398_mindistl+0.0000000000000001)
df_stats_combined['RDP_10398_prichness_log'] = np.log10(df_stats_combined.
 ↪RDP_10398_prichness+0.0000000000000001)
df_stats_combined.head()
```

[261]:

```
                                         seqID  CC11CM0_score  RDP_10398_adcl  \
0  CC11CM0SCR00cdec97c058446e83e0ef032e61806d            2.0        0.000007
1  CC11CM0SCR35529da454f0497fa16e04841e8e1639           34.0        0.000008
2  CC11CM0SCR1083e70ce28e4961b8298356c0d69000            2.0        0.000006
3  CC11CM0SCR8898bf89e307445282cd2fb3acb183a0            2.0        0.000005
4  CC11CM0SCRfc7f700b82e44da39405162a7e5b8b77            2.0        0.000001

   RDP_10398_edpl  RDP_10398_mindistl  RDP_10398_prichness  \
0        0.000000        7.189148e-06                  1.0
1        0.086949        5.645650e-07                 20.0
2        0.000000        6.238599e-06                  1.0
3        0.010897        5.146191e-06                  3.0
4        0.090313        5.645650e-07                 20.0

   RDP_10398_adcl_log  RDP_10398_edpl_log  CC11CM0_score_log  \
0           -5.154902          -16.000000           0.301030
1           -5.096910           -1.060736           1.531479
2           -5.221849          -16.000000           0.301030
3           -5.301030           -1.962705           0.301030
4           -6.000000           -1.044248           0.301030

   RDP_10398_mindistl_log  RDP_10398_prichness_log
0               -5.143323                 0.000000
1               -6.248286                 1.301030
2               -5.204913                 0.000000
3               -5.288514                 0.477121
4               -6.248286                 1.301030
```

[262]:

```python
df_stats_combined_log =␣
 ↪df_stats_combined[["seqID","CC11CM0_score_log","RDP_10398_adcl_log","RDP_10398_edpl_log","R
df_stats_combined_log.head()
```

[262]:

```
                                         seqID  CC11CM0_score_log  \
0  CC11CM0SCR00cdec97c058446e83e0ef032e61806d           0.301030
1  CC11CM0SCR35529da454f0497fa16e04841e8e1639           1.531479
2  CC11CM0SCR1083e70ce28e4961b8298356c0d69000           0.301030
3  CC11CM0SCR8898bf89e307445282cd2fb3acb183a0           0.301030
4  CC11CM0SCRfc7f700b82e44da39405162a7e5b8b77           0.301030

   RDP_10398_adcl_log  RDP_10398_edpl_log  RDP_10398_mindistl_log  \
0           -5.154902          -16.000000               -5.143323
1           -5.096910           -1.060736               -6.248286
2           -5.221849          -16.000000               -5.204913
3           -5.301030           -1.962705               -5.288514
4           -6.000000           -1.044248               -6.248286

   RDP_10398_prichness_log
0                 0.000000
```

```
1                  1.301030
2                  0.000000
3                  0.477121
4                  1.301030
```

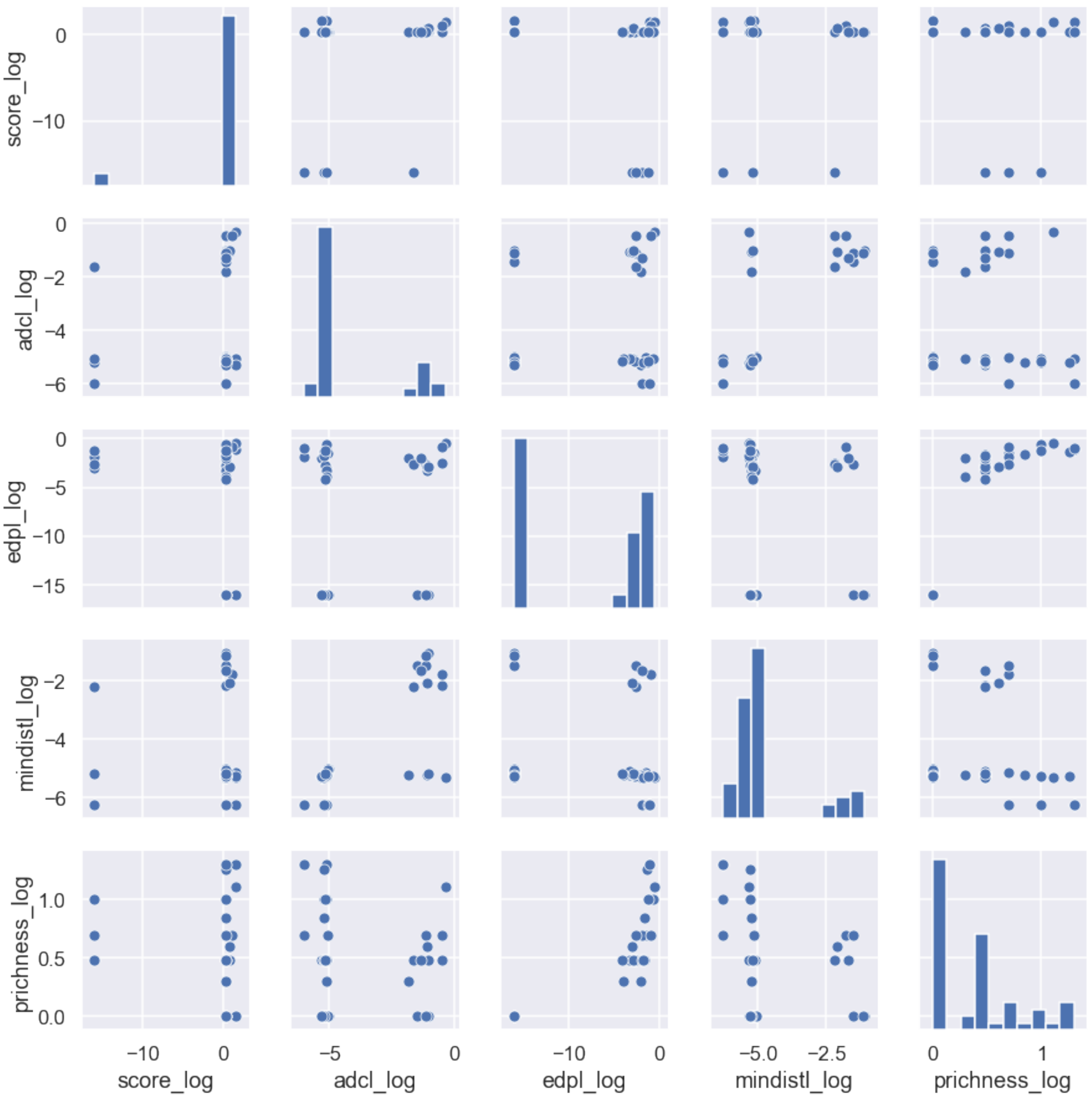
[263]:
```
df_stats_combined_log = df_stats_combined_log.rename(columns={"seqID":
 ↪"CC11CM0_RDP10398_seqID","CC11CM0_score_log":
 ↪"score_log","RDP_10398_adcl_log":"adcl_log","RDP_10398_edpl_log":
 ↪"edpl_log","RDP_10398_mindistl_log":"mindistl_log","RDP_10398_prichness_log":
 ↪"prichness_log"})
```

[264]:
```
df_stats_combined_log.head()
```

[264]:
```
                       CC11CM0_RDP10398_seqID  score_log  adcl_log   edpl_log  \
0  CC11CM0SCR00cdec97c058446e83e0ef032e61806d   0.301030 -5.154902 -16.000000
1  CC11CM0SCR35529da454f0497fa16e04841e8e1639   1.531479 -5.096910  -1.060736
2  CC11CM0SCR1083e70ce28e4961b8298356c0d69000   0.301030 -5.221849 -16.000000
3  CC11CM0SCR8898bf89e307445282cd2fb3acb183a0   0.301030 -5.301030  -1.962705
4  CC11CM0SCRfc7f700b82e44da39405162a7e5b8b77   0.301030 -6.000000  -1.044248

   mindistl_log  prichness_log
0     -5.143323       0.000000
1     -6.248286       1.301030
2     -5.204913       0.000000
3     -5.288514       0.477121
4     -6.248286       1.301030
```

[265]:
```
sns.pairplot(df_stats_combined_log)
```

[265]:
```
<seaborn.axisgrid.PairGrid at 0x352418d0>
```



[266]:

```
def histogram_intersection(a, b):
    v = np.minimum(a, b).sum().round(decimals=1)
    return v
```

[267]:

```
# df_stats_combined_log.corr(method=histogram_intersection)
```

[268]:

```
df_stats_combined_log.corr()
```

[268]:

```
               score_log  adcl_log  edpl_log  mindistl_log  prichness_log
score_log       1.000000  0.055361 -0.240960      0.049552      -0.175765
adcl_log        0.055361  1.000000  0.220745      0.783043       0.067198
edpl_log       -0.240960  0.220745  1.000000     -0.027482       0.870804
mindistl_log    0.049552  0.783043 -0.027482      1.000000      -0.158494
prichness_log  -0.175765  0.067198  0.870804     -0.158494       1.000000
```

```
[ ]:
```

```
[ ]:
```

```
[76]: adcl_cc11.columns
```

```
[76]: Index(['index', 'RDP_10398', 'RDP_5224', 'RDP_1017', 'RDP_92', 'RDP_12',
             'community', 'sourceSeq', 'seqID', 'organism', 'ncbi_tax_id',
             'multiplicity'],
            dtype='object')
```

```
[77]: df_pcorrect.index
```

```
[77]: Index(['CC11CM0', 'CC11CM1', 'CC11CM10', 'CC11CM11', 'CC11CM12', 'CC11CM13',
             'CC11CM14', 'CC11CM15', 'CC11CM16', 'CC11CM17', 'CC11CM18', 'CC11CM19',
             'CC11CM2', 'CC11CM20', 'CC11CM21', 'CC11CM22', 'CC11CM23', 'CC11CM24',
             'CC11CM25', 'CC11CM26', 'CC11CM27', 'CC11CM28', 'CC11CM29', 'CC11CM3',
             'CC11CM30', 'CC11CM31', 'CC11CM32', 'CC11CM33', 'CC11CM34', 'CC11CM35',
             'CC11CM36', 'CC11CM37', 'CC11CM38', 'CC11CM39', 'CC11CM4', 'CC11CM40',
             'CC11CM41', 'CC11CM42', 'CC11CM43', 'CC11CM44', 'CC11CM45', 'CC11CM46',
             'CC11CM47', 'CC11CM48', 'CC11CM49', 'CC11CM5', 'CC11CM50', 'CC11CM51',
             'CC11CM52', 'CC11CM53', 'CC11CM54', 'CC11CM55', 'CC11CM56', 'CC11CM57',
             'CC11CM58', 'CC11CM59', 'CC11CM6', 'CC11CM60', 'CC11CM61', 'CC11CM62',
             'CC11CM63', 'CC11CM64', 'CC11CM65', 'CC11CM66', 'CC11CM67', 'CC11CM68',
             'CC11CM69', 'CC11CM7', 'CC11CM70', 'CC11CM71', 'CC11CM72', 'CC11CM73',
             'CC11CM74', 'CC11CM75', 'CC11CM76', 'CC11CM77', 'CC11CM78', 'CC11CM79',
             'CC11CM8', 'CC11CM80', 'CC11CM81', 'CC11CM82', 'CC11CM83', 'CC11CM84',
             'CC11CM85', 'CC11CM86', 'CC11CM87', 'CC11CM88', 'CC11CM89', 'CC11CM9',
             'CC11CM90', 'CC11CM91', 'CC11CM92', 'CC11CM93', 'CC11CM94', 'CC11CM95',
             'CC11CM96', 'CC11CM97', 'CC11CM98', 'CC11CM99'],
            dtype='object')
```

```
[78]: sensitivity_10 = df_pcorrect.loc[['CC11CM0', 'CC11CM1', 'CC11CM2', 'CC11CM3',␣
       ↪'CC11CM4', 'CC11CM5',
             'CC11CM6', 'CC11CM7', 'CC11CM8', 'CC11CM9','CC11CM10', 'CC11CM11',␣
       ↪'CC11CM1', 'CC11CM13', 'CC11CM14', 'CC11CM15',
             'CC11CM16', 'CC11CM17', 'CC11CM18', 'CC11CM19']]
```

```
[79]: sensitivity_10
```

```
[79]:          rdp_10398  rdp_5224  rdp_1017    rdp_92  rdp_12
      CC11CM0   5.765636  5.765636  3.159084  2.504251     0.0
      CC11CM1   0.363896  0.378621  0.000000  0.000000     0.0
      CC11CM2   0.743100  0.743100  0.743100  0.000000     0.0
      CC11CM3   0.000000  0.000000  0.000000  0.000000     0.0
      CC11CM4   0.546669  0.435311  0.430741  0.000000     0.0
      CC11CM5   0.883439  3.843878  3.466427  3.466427     0.0
      CC11CM6   1.707271  1.707271  1.005509  0.000000     0.0
      CC11CM7   0.988916  1.269109  1.269109  1.269109     0.0
      CC11CM8   4.696282  4.696282  3.632541  2.393920     0.0
      CC11CM9   6.264562  3.415929  3.334887  0.000000     0.0
```

```
CC11CM10    7.533649   7.499137   5.743113   3.020768     0.0
CC11CM11    0.623034   0.036293   0.036293   0.000000     0.0
CC11CM1     0.363896   0.378621   0.000000   0.000000     0.0
CC11CM13    0.171222   0.171222   0.171222   0.000000     0.0
CC11CM14    6.230596   7.159869  11.374559   3.731978     0.0
CC11CM15   13.859362  11.623058  11.623058  10.569166     0.0
CC11CM16   14.508393  14.447425  14.825428   9.539408     0.0
CC11CM17    0.000000   0.000000   0.000000   0.000000     0.0
CC11CM18   10.916216   0.000000   0.000000   0.000000     0.0
CC11CM19    0.000000   0.000000   0.000000   0.000000     0.0
```

[ ]:

```

```

0.4 Read score tables

[81]:

```
rdp_10398_score = pd.read_csv("rdp_10398score.csv", index_col=0)
rdp_5224_score = pd.read_csv("rdp_5224score.csv", index_col=0)
rdp_1017_score = pd.read_csv("rdp_1017score.csv", index_col=0)
rdp_92_score = pd.read_csv("rdp_92score.csv", index_col=0)
rdp_12_score = pd.read_csv("rdp_12score.csv", index_col=0)
```

[82]:

```
adcl.head(1)
```

[82]:

```
         index  \
newIndex
CC11CM81SCReeb34a221dca4dc5b0ca403cc9ac9bd3
CC11CM81SCReeb34a221dca4dc5b0ca403cc9ac9bd3_Fr...

                                             RDP_10398  RDP_5224  RDP_1017  \
newIndex
CC11CM81SCReeb34a221dca4dc5b0ca403cc9ac9bd3   0.000007  0.137668  0.021522

                                               RDP_92    RDP_12
newIndex
CC11CM81SCReeb34a221dca4dc5b0ca403cc9ac9bd3  0.155496  0.204297
```

[83]:

```
rdp_10398_score.head(1)
```

[83]:

```
                            CC11CM0  CC11CM1  CC11CM2  CC11CM3  CC11CM4  \
sv_id
AGEL01000014.278829.280347      NaN      NaN      NaN      NaN      NaN

                            CC11CM5  CC11CM6  CC11CM7  CC11CM8  CC11CM9  ...  \
sv_id                                                                    ...
AGEL01000014.278829.280347      NaN      NaN      NaN      NaN      6.0  ...

                            CC11CM90  CC11CM91  CC11CM92  CC11CM93  CC11CM94  \
sv_id
AGEL01000014.278829.280347       NaN       NaN       6.0       6.0       NaN
```

```
                           CC11CM95  CC11CM96  CC11CM97  CC11CM98  CC11CM99
sv_id
AGEL01000014.278829.280347       NaN       NaN       6.0       NaN       NaN

[1 rows x 100 columns]
```

[84]:

```
rdp_10398_score.shape
```

[84]:

```
(1830, 100)
```

[85]:

```
combined =pd.DataFrame(columns=["unique_index", "rdp10398"])
```

[86]:

```
combined=combined.append({'unique_index': 'abd', 'rdp10398': 12},␣
 ↪ignore_index=True)
combined.columns
```

[86]:

```
Index(['unique_index', 'rdp10398'], dtype='object')
```

[87]:

```
combined
```

[87]:

```
  unique_index rdp10398
0          abd       12
```

[88]:

```
str("abc") + str("xyz")
rdp_10398_score.loc["AGEL01000014.278829.280347","CC11CM0"]
```

[88]:

```
nan
```

[89]:

```
indices=rdp_10398_score.index
communities =rdp_10398_score.columns
```

[90]:

```
for mock_id in indices:
    for community in communities:
        combined=combined.append({'unique_index': str(community)+str(mock_id),␣
 ↪'rdp10398': rdp_10398_score.loc[mock_id,community]}, ignore_index=True)
```

[91]:

```
combined.columns
```

[91]:

```
Index(['unique_index', 'rdp10398'], dtype='object')
```

[92]:

```
# This function converts table with the index of community name and the sv_id␣
 ↪name
def convert_score_table(score_table, prefix):
    new_table=pd.DataFrame(columns=["unique_index", prefix])
    indices=score_table.index
    communities =score_table.columns
    for mock_id in indices:
        for community in communities:
           new_table=new_table.append({'unique_index':␣
 ↪str(community)+str(mock_id), prefix: score_table.loc[mock_id,community]},␣
 ↪ignore_index=True)
    new_table.set_index('unique_index')
    return new_table
```

[93]:
```
rdp_10398_converted = convert_score_table(rdp_10398_score, "rdp_10398")
rdp_5224_converted = convert_score_table(rdp_5224_score, "rdp_5224")
rdp_1017_converted = convert_score_table(rdp_1017_score, "rdp_1017")
rdp_92_converted = convert_score_table(rdp_92_score, "rdp_92")
rdp_12_converted = convert_score_table(rdp_12_score, "rdp_12")
```

[94]:
```
rdp_10398_converted.to_csv("rdp_10398_converted.csv")
rdp_5224_converted.to_csv("rdp_5224_converted.csv")
rdp_1017_converted.to_csv("rdp_1017_converted.csv")
rdp_92_converted.to_csv("rdp_92_converted.csv")
rdp_12_converted.to_csv("rdp_12_converted.csv")
```

[95]:
```
data_frames=[rdp_10398_converted,rdp_5224_converted,rdp_1017_converted,rdp_92_converted,rdp_12
```

[96]:
```
rdp_12_converted.columns
```

[96]:
```
Index(['unique_index', 'rdp_12'], dtype='object')
```

[97]:
```
df_merged = reduce(lambda  left,right: pd.merge(left,right,on=['unique_index'],
                                            how='outer'), data_frames)
df_merged.rename(columns={'rdp_10398':'RDP_10398_score','rdp_5224':
 ↪'RDP_5224_score','rdp_1017':'RDP_1017_score','rdp_92':
 ↪'RDP_92_score','rdp_12':'RDP_12_score' }, inplace=True)
df_merged.columns
```

[97]:
```
Index(['unique_index', 'RDP_10398_score', 'RDP_5224_score', 'RDP_1017_score',
       'RDP_92_score', 'RDP_12_score'],
      dtype='object')
```

[98]:
```
df_merged.to_csv("score_merged.csv")
```

[99]:
```
adcl_cc11.columns
```

[99]:
```
Index(['index', 'RDP_10398', 'RDP_5224', 'RDP_1017', 'RDP_92', 'RDP_12',
       'community', 'sourceSeq', 'seqID', 'organism', 'ncbi_tax_id',
       'multiplicity'],
      dtype='object')
```

[ ]:
```

```

[100]:
```
adcl_pplacer = adcl_cc11.rename(columns={'RDP_10398':
 ↪'RDP_10398_adcl','RDP_5224':'RDP_5224_adcl','RDP_1017':
 ↪'RDP_1017_adcl','RDP_92':'RDP_92_adcl','RDP_12':'RDP_12_adcl'})
edpl_pplacer = edpl_cc11.rename(columns={'RDP_10398':
 ↪'RDP_10398_edpl','RDP_5224':'RDP_5224_edpl','RDP_1017':
 ↪'RDP_1017_edpl','RDP_92':'RDP_92_edpl','RDP_12':'RDP_12_edpl'})
```

```
mindistl_pplacer = mindistl_cc11.rename(columns={'RDP_10398':
 ↪'RDP_10398_mindistl','RDP_5224':'RDP_5224_mindistl','RDP_1017':
 ↪'RDP_1017_mindistl','RDP_92':'RDP_92_mindistl','RDP_12':'RDP_12_mindistl'})
# prichness_pplacer=prichness.rename(columns={'newIndex':'seqID','RDP_10398':
 ↪'RDP_10398_prichness','RDP_5224':'RDP_5224_prichness','RDP_1017':
 ↪'RDP_1017_prichness','RDP_92':'RDP_92_prichness','RDP_12':
 ↪'RDP_12_prichness'})
prichness_pplacer = prichness_cc11.rename(columns={'RDP_10398':
 ↪'RDP_10398_prichness','RDP_5224':'RDP_5224_prichness','RDP_1017':
 ↪'RDP_1017_prichness','RDP_92':'RDP_92_prichness','RDP_12':
 ↪'RDP_12_prichness'})
```

[ ]:

```

```

[101]:

```
pplacer_stats=[adcl_pplacer,edpl_pplacer,mindistl_pplacer,prichness_pplacer]
pplacer_stats_merged = reduce(lambda  left,right: pd.
 ↪merge(left,right,on=['seqID'],
                                            how='outer'), pplacer_stats)
```

[102]:

```
pplacer_stats_merged.rename(columns={'seqID':'unique_index'}, inplace=True)
pplacer_stats_merged.to_csv("pplacer_stats_merged.csv")
```

[103]:

```
# merge pplacer stats with scores
merged_dataframe_list = [pplacer_stats_merged,df_merged]
pplacer_stats_score_merged = reduce(lambda  left,right: pd.
 ↪merge(left,right,on=['unique_index'],
                                            how='outer'), merged_dataframe_list)
```

[104]:

```
pplacer_stats_score_merged.drop_duplicates()
pplacer_stats_score_merged.to_csv("pplacer_stats_score_merged.csv")
```

```
      ␣
↪---------------------------------------------------------------------------

      PermissionError                           Traceback (most recent call␣
↪last)

      <ipython-input-104-228a9a5ec095> in <module>
        1 pplacer_stats_score_merged.drop_duplicates()
  ----> 2 pplacer_stats_score_merged.to_csv("pplacer_stats_score_merged.csv")


      d:\program files (x86)\python37-32\lib\site-packages\pandas\core\generic.
↪py in to_csv(self, path_or_buf, sep, na_rep, float_format, columns, header,␣
↪index, index_label, mode, encoding, compression, quoting, quotechar,␣
↪line_terminator, chunksize, tupleize_cols, date_format, doublequote,␣
↪escapechar, decimal)
     3018                                    doublequote=doublequote,
      3019                                  escapechar=escapechar,␣
↪decimal=decimal)
  -> 3020         formatter.save()
      3021
      3022         if path_or_buf is None:


      d:\program files␣
↪(x86)\python37-32\lib\site-packages\pandas\io\formats\csvs.py in save(self)
      155             f, handles = _get_handle(self.path_or_buf, self.mode,
      156                                      encoding=self.encoding,
  --> 157                                      compression=self.compression)
      158             close = True
      159


      d:\program files (x86)\python37-32\lib\site-packages\pandas\io\common.py␣
↪in _get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text)
      422         elif encoding:
      423             # Python 3 and encoding
  --> 424             f = open(path_or_buf, mode, encoding=encoding,␣
↪newline="")
      425         elif is_text:
      426             # Python 3 and no explicit encoding


      PermissionError: [Errno 13] Permission denied:␣
↪'pplacer_stats_score_merged.csv'
```

[ ]:

```
# score_table_list= list()
# for f in␣
 ↪{rdp_10398_score,rdp_5224_score,rdp_1017_score,rdp_92_score,rdp_12_score}:
#     score_table_list.append(f)
# # prefix_list = {"rdp_10398","rdp_5224","rdp_1017","rdp_92","rdp_12"}
```

[ ]:

```
```

[ ]:

```
# df_combined_list={}
# for i in range(len(score_table_list)):
#    df_combined_list[i]=convert_score_table(score_table_list[i],␣
 ↪prefix_list[i])
```

[ ]:

```
# for i in range(len(score_table_list)):
#     print (i)
```

[ ]:

```
# rdp_10398_score = pd.read_csv("rdp_10398score.csv", index_col=0)
# rdp_5224_score = pd.read_csv("rdp_5224score.csv", index_col=0)
# rdp_1017_score = pd.read_csv("rdp_1017score.csv", index_col=0)
# rdp_92_score = pd.read_csv("rdp_92score.csv", index_col=0)
# rdp_12_score = pd.read_csv("rdp_12score.csv", index_col=0)
```

[ ]:

```

```

[ ]:

```

```