Levenshtein

May 22, 2018

0.1 Step 2.1

```
In [1]: from itertools import groupby
        import numpy as np
        import pandas as pd
        import sklearn
        import matplotlib.pyplot as plt
        import numpy as np
        import scipy.io
```

```
In [2]: def cost_fun(c1,c2):
            if c1==c2:
                return 0
            else:
                if ((c1 in ['G','A','V','L','I']) and (c2 in ['G','A','V','L','I'])):
                    return 1
                if ((c1 in ['S','C','T','M']) and (c2 in ['S','C','T','M'])):
                    return 1
                if ((c1 in ['F','Y','W']) and (c2 in ['F','Y','W'])):
                    return 1
                if ((c1 in ['H','K','R']) and (c2 in ['H','K','R'])):
                    return 1
                if ((c1 in ['D','E','N','Q']) and (c2 in ['D','E','N','Q'])):
                    return 1
                if (c1=='X' or c2=='X'):
                    return 1
                return 2
```

```
In [3]: def levenshtein(s1, s2):
            insertions = 0
            deletions = 0
            substitutions=0

            if len(s1) < len(s2):
                return levenshtein(s2, s1)
```

```
            # len(s1) >= len(s2)
            if len(s2) == 0:
                return len(s1)

            memo=np.zeros((len(s1)+1, len(s2)+1),np.int16)

            memo[0]=range(len(s2)+1)

            for i, c1 in enumerate(s1):

                memo[i+1,0]=i+1
                for j, c2 in enumerate(s2):
                    insertions = memo[i,j + 1] + 1
                    deletions = memo[i+1,j] + 1
                    substitutions = memo[i,j] + cost_fun(c1,c2)
                    memo[i+1,j+1]=min(insertions, deletions, substitutions)

            return memo[-1,-1]
```

In [4]:
```
def fasta_iter(fasta_name):
    """
    given a fasta file. yield tuples of header, sequence
    """
    fh = open(fasta_name)
    # ditch the boolean (x[0]) and just keep the header or sequence since
    # we know they alternate.
    faiter = (x[1] for x in groupby(fh, lambda line: line[0] == ">"))
    for header in faiter:
        # drop the ">"
        header = header.__next__()[1:].strip()

        # join all sequence lines to one.
        seq = "".join(s.strip() for s in faiter.__next__())
        yield header, seq
```

In [5]:
```
sequence2  = ['P','I','D','N','Y','L','K','L','L','K','C','R','I','I','H','N','N','N','C
sequence1=['M','N','I','K','G','S','P','W','K','G','S','L','L','L','L','L','V','S','N','

lev_dist=levenshtein(sequence2,sequence1)
```

In [6]:
```
lev_dist
```

Out[6]:
```
208
```

In [7]:
```
def dist_matrix(ref):
    dist=[]
    name=[]
    index_t=[]
    file=fasta_iter('proteins.fasta')
```

```
            for i, j in file:
                dist.append(levenshtein(j, ref))
                name.append(i.split(' ', 1)[1])
                index_t.append(i.split(' ', 1)[0])

            return dist,name,index_t
```

In [ ]:
```
def output_k_homologs(ref,k):
    dist,name,index_t=dist_matrix(ref)
    dist_dataframe = pd.DataFrame({'Protein Name': name ,'levenshteinDistance': dist },

    dist_dataframe=dist_dataframe.sort('levenshteinDistance', ascending=True)
    dist_dataframe=dist_dataframe.head(10)

    return dist_dataframe
```

In [ ]:
```
reference_sequence=['M','N','I','K','G','S','P','W','K','G','S','L','L','L','L','L','V',

dist_dataframe=k_homolog_list=output_k_homologs(reference_sequence, 10)
dist_dataframe
```

In [ ]:
```
print(dist_dataframe.dtype)
```