neighbor-joining

April 13, 2022

[118]:
```
!pip install pandas
```

```
Requirement already satisfied: pandas in /opt/conda/lib/python3.9/site-packages
(1.4.1)
Requirement already satisfied: python-dateutil>=2.8.1 in
/opt/conda/lib/python3.9/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.9/site-
packages (from pandas) (2021.3)
Requirement already satisfied: numpy>=1.18.5 in /opt/conda/lib/python3.9/site-
packages (from pandas) (1.21.5)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.9/site-
packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

[119]:
```
import pandas as pd
import numpy as np
import string

np.set_printoptions(precision=3)
```

[120]:
```
def load_distance_matrix(file_name: str) -> np.ndarray:
    sample = np.loadtxt(file_name)
    index = [chr(65 + i) for i in range(sample.shape[0])]
    sample = sample[1:, :-1]
    return pd.DataFrame(sample, index=index[1:], columns=index[:-1])

def get_argmin_value(matrix: np.ndarray) -> tuple[int, int]:
    return tuple(np.argwhere(matrix == np.min(matrix))[0])


def init_index_dict(len_: int) -> list[str]:
    index_dict = [chr(65 + i) for i in range(len_)]
    return index_dict


def get_u_mean_distances(distance_matrix: np.ndarray) -> np.array:
    u_mean_distances = []
    for i in range(distance_matrix.shape[0]):
        u_mean_distances.append(distance_matrix.values[:, i].mean())
```

```
    u_mean_distances = np.array(u_mean_distances)
    return u_mean_distances


def get_pairs_distance(distance_matrix: np.ndarray, u_mean_distances: np.array)␣
 ↪-> np.ndarray:
    pairs_distance = np.zeros(distance_matrix.shape)
    for i in range(distance_matrix.shape[0]):
        for j in range(i + 1, distance_matrix.shape[1]):
            pairs_distance[i, j] = distance_matrix.values[i, j] -␣
 ↪u_mean_distances[i] - u_mean_distances[j]
    pairs_distance = pairs_distance.transpose()
    return pairs_distance


def get_distance_from_smallest_pair_to_new_node(distance_matrix: np.ndarray,
                                                u_mean_distances: np.array,
                                                smallest_pair: tuple[int, int])␣
 ↪-> tuple[int, int]:
    d0 = distance_matrix.values[smallest_pair] / 2 + (
                u_mean_distances[smallest_pair[0]] -␣
 ↪u_mean_distances[smallest_pair[1]]) / 2

    d1 = distance_matrix.values[smallest_pair] / 2 + (
                u_mean_distances[smallest_pair[1]] -␣
 ↪u_mean_distances[smallest_pair[0]]) / 2
    return d0, d1



def get_new_values(distance_matrix: np.ndarray,
                   smallest_pair: tuple[int, int],
                   pairs_distance: np.ndarray) -> tuple[list[float],␣
 ↪list[float]]:

    new_row = []
    for i in range(distance_matrix.shape[1]):
        if i not in smallest_pair:
            new_value = (distance_matrix.values[i, smallest_pair[0]] +␣
 ↪distance_matrix.values[i, smallest_pair[1]] -
                         pairs_distance[smallest_pair]) / 2
            new_row.append(new_value)

    new_column = []
    for i in range(distance_matrix.shape[0]):
        if i not in smallest_pair:
```

```
            new_value = (distance_matrix.values[i, smallest_pair[0]] +␣
 ↪distance_matrix.values[i, smallest_pair[1]] -
                         pairs_distance[smallest_pair]) / 2
            new_column.append(new_value)

    return new_row, new_column


def get_new_distance_matrix(distance_matrix: np.ndarray,
                            smallest_pair: tuple[int, int],
                            new_row: list[int],
                            new_column: list[int],
                            iteration: int) -> np.ndarray:
    rows_columnns_to_remove = [distance_matrix.index[smallest_pair[0]],
                               distance_matrix.columns[smallest_pair[1]]]

    distance_matrix.drop(index=rows_columnns_to_remove,
                         errors="ignore",
                         inplace=True)

    distance_matrix.drop(columns=rows_columnns_to_remove,
                         errors="ignore",
                         inplace=True)

    distance_matrix[f"U{iteration}"] = new_column

    new_row.append(0)
    distance_matrix.loc[f"U{iteration}"] = new_row

    return distance_matrix
```

[121]:
```
def main():
    distance_matrix = load_distance_matrix("input.txt")
    nick_tree = []
    rows, columns = distance_matrix.shape

    for iteration in range(columns - 1):
        print(distance_matrix)
        u_mean_distances = get_u_mean_distances(distance_matrix)
        pairs_distance = get_pairs_distance(distance_matrix, u_mean_distances)
        smallest_pair = get_argmin_value(pairs_distance)
        d0, d1 = get_distance_from_smallest_pair_to_new_node(distance_matrix,
                                                             u_mean_distances,
                                                             smallest_pair)
        nick_tree.append((f"U{iteration}", distance_matrix.
 ↪index[smallest_pair[0]], d0))
```

```
        nick_tree.append((f"U{iteration}", distance_matrix.
↪columns[smallest_pair[1]], d1))
        new_row, new_column = get_new_values(distance_matrix,
                                             smallest_pair,
                                             pairs_distance)

        # break
        distance_matrix = get_new_distance_matrix(distance_matrix,
                                                  smallest_pair,
                                                  new_row,
                                                  new_column,
                                                  iteration)
        rows, columns = distance_matrix.shape
        print("*" * 80)
```

[122]:
```
main()
```

```
          A         B         C         D         E         F         G
B  2.226662  0.000000  1.156708  0.914018  1.380749  4.450111  3.885507
C  3.335384  1.156708  0.000000  1.819830  2.489471  5.558833  4.994230
D  3.092694  0.914018  1.819830  0.000000  2.246781  5.316143  4.751540
E  1.889630  1.380749  2.489471  2.246781  0.000000  3.347123  2.782520
F  4.958991  4.450111  5.558833  5.316143  3.347123  0.000000  1.196623
G  4.394388  3.885507  4.994230  4.751540  2.782520  1.196623  0.000000
H  5.412214  4.903333  6.012055  5.769365  3.800346  2.214449  1.217620
********************************************************************************
          A         B         D         F         G        U0
B  2.226662  0.000000  0.914018  4.450111  3.885507  4.095991
D  3.092694  0.914018  0.000000  5.316143  4.751540  3.970543
F  4.958991  4.450111  5.316143  0.000000  1.196623  8.498116
G  4.394388  3.885507  4.751540  1.196623  0.000000  7.933513
H  5.412214  4.903333  5.769365  2.214449  1.217620  8.951339
U0 4.095991  3.970543  8.498116  7.933513  8.951339  0.000000
********************************************************************************
          A         B         F         U0        U1
B  2.226662  0.000000  4.450111  4.095991  6.545399
F  4.958991  4.450111  0.000000  8.498116  6.521406
H  5.412214  4.903333  2.214449  8.951339  7.855242
U0 4.095991  3.970543  7.933513  0.000000 12.079150
U1 6.545399  6.521406  7.855242 12.079150  0.000000
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [122], in <cell line: 1>()
----> 1 main()

Input In [121], in main()
```

```
     16 new_row, new_column = get_new_values(distance_matrix,
     17                                       smallest_pair,
     18                                       pairs_distance)
     20 # break
---> 21 distance_matrix = get_new_distance_matrix(distance_matrix,
     22                                            smallest_pair,
     23                                            new_row,
     24                                            new_column,
     25                                            iteration)
     26 rows, columns = distance_matrix.shape
     27 print("*" * 80)

Input In [120], in get_new_distance_matrix(distance_matrix, smallest_pair,␣
 ↪new_row, new_column, iteration)
     74 distance_matrix.drop(index=rows_columnns_to_remove,
     75                       errors="ignore",
     76                       inplace=True)
     78 distance_matrix.drop(columns=rows_columnns_to_remove,
     79                       errors="ignore",
     80                       inplace=True)
---> 82 distance_matrix[f"U{iteration}"] = new_column
     84 new_row.append(0)
     85 distance_matrix.loc[f"U{iteration}"] = new_row

File /opt/conda/lib/python3.9/site-packages/pandas/core/frame.py:3655, in␣
 ↪DataFrame.__setitem__(self, key, value)
   3652     self._setitem_array([key], value)
   3653 else:
   3654     # set column
-> 3655     self._set_item(key, value)

File /opt/conda/lib/python3.9/site-packages/pandas/core/frame.py:3832, in␣
 ↪DataFrame._set_item(self, key, value)
   3822 def _set_item(self, key, value) -> None:
   3823     """
   3824     Add series to DataFrame in specified column.
   3825
   (…)
   3830     ensure homogeneity.
   3831     """
-> 3832     value = self._sanitize_column(value)
   3834     if (
   3835         key in self.columns
   3836         and value.ndim == 1
   3837         and not is_extension_array_dtype(value)
   3838     ):
   3839         # broadcast across multiple columns if necessary
```

```
   3840         if not self.columns.is_unique or isinstance(self.columns,␣
 ↪MultiIndex):

File /opt/conda/lib/python3.9/site-packages/pandas/core/frame.py:4529, in␣
 ↪DataFrame._sanitize_column(self, value)
   4526     return _reindex_for_setitem(value, self.index)
   4528 if is_list_like(value):
-> 4529     com.require_length_match(value, self.index)
   4530 return sanitize_array(value, self.index, copy=True, allow_2d=True)

File /opt/conda/lib/python3.9/site-packages/pandas/core/common.py:557, in␣
 ↪require_length_match(data, index)
    553 """
    554 Check the length of data matches the length of the index.
    555 """
    556 if len(data) != len(index):
--> 557     raise ValueError(
    558         "Length of values "
    559         f"({len(data)}) "
    560         "does not match length of index "
    561         f"({len(index)})"
    562     )

ValueError: Length of values (3) does not match length of index (4)
```

[ ]: