Fill in missing values using Linear 2-dimensional interpolation for multidimensional data

OOP Project Group 2 Interpolation
Andrey Domnyshev
28.03.2021

```
interp_2D(data *frame, auto *boolean, argument_column_initial *integer,
          list_remove_column *list)
return *frame
```

The function get 'pandas.core.frame.DataFrame' object which can consist of missing values and returns the same type object but with less number of missing values or without them.

Parameters:

Data **frame*: 'pandas.core.frame.DataFrame' object that can consist of NaN entities. Each row must have at least two not empty cells (entities). All data besides first row are taken in account, so if object have first column with sequence, it must be deleted:

auto **boolean*: if auto=False, user does not allow program to change argument column. If auto=True, program still uses user's preference, but also provides interpolation using other argument columns.

argument_column_initial **integer*: integer value, that indicates which column of frame should be used as argument column – column consists of 1-D array with "x" values used to approximate some function f: y = f(x). Every time when program find empty entity it considers the column with empty entity as function column – column consists of 1-D array with "y" values.

list_remove_column **list*: list with columns numbers, that must not be considered as argument column (1-D array with "x" values). For example, 1st "No" column in "Real estate dataset that consist only of sequence, that does not have any sense for data analysation.

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores
0	1	2012.916667	NaN	84.87882	10.0
1	2	2012.916667	19.5	NaN	9.0
2	3	2013.583333	13.3	561.98450	5.0
3	4	2013.500000	NaN	561.98450	5.0
4	5	2012.833333	5.0	390.56840	5.0
..	...	...	...	...	...
409	410	2013.000000	13.7	4082.01500	0.0
410	411	2012.666667	5.6	90.45606	NaN
411	412	2013.250000	18.8	390.96960	7.0
412	413	2013.000000	8.1	104.81010	5.0
413	414	2013.500000	6.5	90.45606	9.0

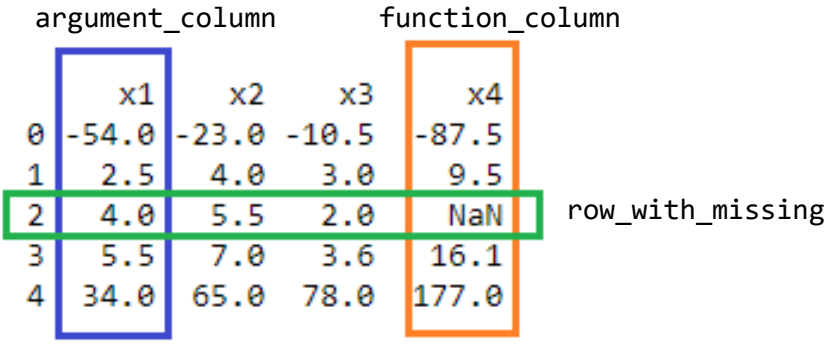
Example 1

```
import pandas as pd
old_data = pd.read_excel('Book5.xlsx')
print(old_data)
new_data = interp_2D(old_data, False, 0, [])
print()
print(new_data)
```

```
     x1    x2    x3     x4
0 -54.0 -23.0 -10.5  -87.5
1   2.5   4.0   3.0    9.5
2   4.0   5.5   2.0    NaN
3   5.5   7.0   3.6   16.1
4  34.0  65.0  78.0  177.0

      0     1     2      3
0 -54.0 -23.0 -10.5  -87.5
1   2.5   4.0   3.0    9.5
2   4.0   5.5   2.0   12.8
3   5.5   7.0   3.6   16.1
4  34.0  65.0  78.0  177.0
```

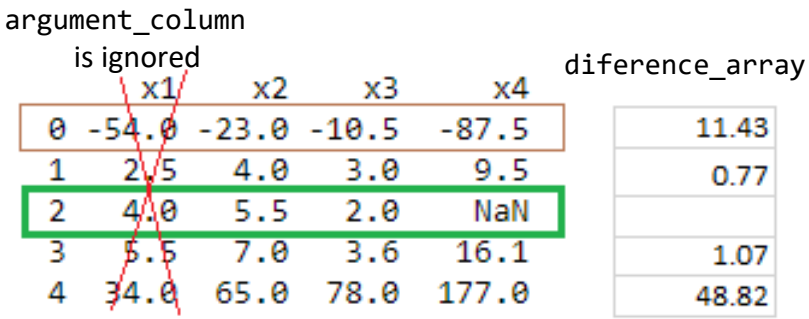
In this case we have only one missing value on 2 row and x4 column and it is not important if we use auto mode (`auto=True`) or not (`auto=False`), the argument column will be x1:



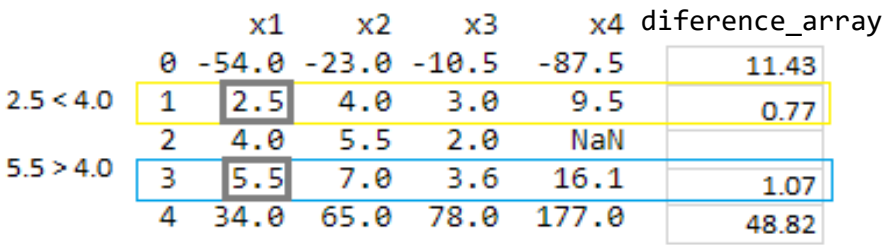
After defining `argument_column`, `function_column` and `row_with_missing` program calculate differences between row with missing and each of row.

For the first row :

$$difference.\_array[0] = \frac{|-23.0 - 5.5|}{|5.5|} + \frac{|-10.5 - 2|}{|2|} = 11.43$$



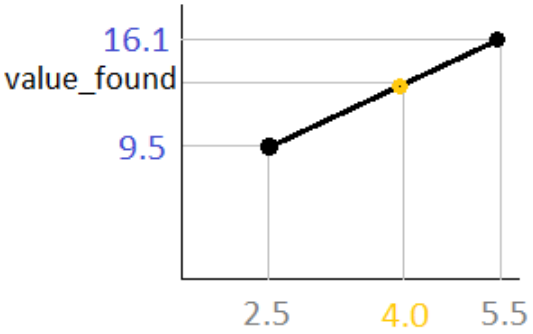
Then program goes throw `difference_array` and find two the smallest values: one that matches to the row where value in `argument_column` lower than in `row_with_missing` and one where higher:



Now we have all data to provide linear interpolation:

```
X_array = np.array ([2.5, 5.5], dtype = 'float64')
Y_array = np.array ([9.5, 16.1], dtype = 'float64')
fun = interpolate.interp1d(X_array, Y_array, kind='linear')
value_found = fun(4.0)
```

	x1	x2	x3	x4
0	-54.0	-23.0	-10.5	-87.5
1	2.5	4.0	3.0	9.5
2	4.0	5.5	2.0	NaN
3	5.5	7.0	3.6	16.1
4	34.0	65.0	78.0	177.0



This value is rounded and put in matrix. When all missing values are found, matrix is converted to frame. User can also called message from previous interpolation (`message_from_interp_2D` assigned as global!):

```
print(message_from_interp_2D)
```

```
This is the report from last Fill in missing values using 2D interpolation
---START---
28-03-2021 22:12:14.574867
Maximum decimal digits is 1
You have chosen auto mode: auto==True, columns in order[0, 1, 2, 3] will b
e considered as argument
-> Consider argument column no 0
-----> Put value 12.8 in row 2 column 3
-> Consider argument column no 1
-> Consider argument column no 2
-> Consider argument column no 3
Total number of added values: 1
28-03-2021 22:12:14.582867
---FINISH---
```

Example 2

Consider the following input frame:

```
old_data = pd.read_excel('Book1.xlsx')
print(old_data)
new_data = interp_2D(old_data, False, 2, [])
```

```
       x1     x2     x3     x4
0     1.0    1.0    0.0  455.0
1     0.5    0.5  654.0   34.0
2     2.0    2.0    2.0    6.0
3     1.0    1.0    1.5   34.0
4    23.0  543.0    1.0    2.0
5     2.0    3.0    1.0    NaN
6     1.0    1.0  567.0    NaN
7     NaN   -3.4    NaN    5.0
8     0.5    NaN    4.0   -5.0
9   234.0    4.0    5.0 -456.0
10    0.5    0.5    1.0    2.0
11    2.0   34.0 -765.0    6.0
```

```
12    4.0    4.0   33.0    4.0
13    2.0    4.0    NaN    7.0
14   34.0    4.0   46.0  -76.0
```

If Auto=False the output will consist of NaN elements:

```
new_data = interp_2D(old_data, False, 2, [])
```

```
       x1     x2     x3     x4
0     1.0    1.0    0.0  455.0
1     0.5    0.5  654.0   34.0
2     2.0    2.0    2.0    6.0
3     1.0    1.0    1.5   34.0
4    23.0  543.0    1.0    2.0
5     2.0    3.0    1.0  230.5
6     1.0    1.0  567.0   90.0
7     NaN   -3.4    NaN    5.0
8     0.5    0.5    4.0   -5.0
9   234.0    4.0    5.0 -456.0
10    0.5    0.5    1.0    2.0
11    2.0   34.0 -765.0    6.0
12    4.0    4.0   33.0    4.0
13    2.0    4.0    NaN    7.0
14   34.0    4.0   46.0  -76.0
```

The program considered only one argument column and skipped others. It added only 4 values, one of them is NaN:

```
print(message_from_interp_2D)
```

```
---START---
28-03-2021 22:12:14.706826
Maximum decimal digits is 1
You have chosen hand mode: auto==False, only column 2 will be considered a
s argument
-> Consider argument column no 2
-----> Put value 230.5 in row 5 column 3
-----> Put value 90.0 in row 6 column 3
-----> Put value nan in row 7 column 0
Argument column consists of NaN element(s) and you use hand mode (Auto==Fa
lse).
Your output dataframe will still consist of empty entities
-----> Put value 0.5 in row 8 column 1
Argument column consists of NaN element(s) and you use hand mode (Auto==Fa
lse).
Your output dataframe will still consist of empty entities
-> Skip non-argument column no 3
-> Skip non-argument column no 0
-> Skip non-argument column no 1
Total number of added values: 4
28-03-2021 22:12:14.715822
---FINISH---
```

If Auto=True the output frame will not have NaN elements:

```
new_data = interp_2D(old_data, True, 2, [])
```

```
       x1     x2     x3     x4
0     1.0    1.0    0.0  455.0
1     0.5    0.5  654.0   34.0
2     2.0    2.0    2.0    6.0
3     1.0    1.0    1.5   34.0
4    23.0  543.0    1.0    2.0
5     2.0    3.0    1.0  230.5
6     1.0    1.0  567.0   90.0
7     2.8   -3.4    2.2    5.0
8     0.5    0.5    4.0   -5.0
9   234.0    4.0    5.0 -456.0
10    0.5    0.5    1.0    2.0
11    2.0   34.0 -765.0    6.0
12    4.0    4.0   33.0    4.0
13    2.0    4.0    2.0    7.0
14   34.0    4.0   46.0  -76.0
```

Because now program tries to do interpolation using other column as arguments array too. In this case program added 6 values (value in row 7 column 0 first was replaced with NaN and then with another value):

```
---START---
28-03-2021 22:29:52.472589
Maximum decimal digits is 1
You have chosen auto mode: auto==True, columns in order[2, 3, 0, 1] will b
e considered as argument
-> Consider argument column no 2
-----> Put value 230.5 in row 5 column 3
-----> Put value 90.0 in row 6 column 3
-----> Put value nan in row 7 column 0
-----> Put value 0.5 in row 8 column 1
-> Consider argument column no 3
-----> Put value 2.8 in row 7 column 0
-----> Put value 2.2 in row 7 column 2
-----> Put value 2.0 in row 13 column 2
-> Consider argument column no 0
-> Consider argument column no 1
Total number of added values: 7
28-03-2021 22:29:52.482698
---FINISH---
```

Despite the obvious advantage of Auto mode, the data which obtained this method can be less accurate and less predictable.

Further information and tests results will come later