How can I replace all the NaN values with Zero's in a column of a pandas dataframe

Ask Question

330

79

I have a dataframe as below

```
      itm Date                  Amount
67    420 2012-09-30 00:00:00   65211
68    421 2012-09-09 00:00:00   29424
69    421 2012-09-16 00:00:00   29877
70    421 2012-09-23 00:00:00   30990
71    421 2012-09-30 00:00:00   61303
72    485 2012-09-09 00:00:00   71781
73    485 2012-09-16 00:00:00     NaN
74    485 2012-09-23 00:00:00   11072
75    485 2012-09-30 00:00:00  113702
76    489 2012-09-09 00:00:00   64731
77    489 2012-09-16 00:00:00     NaN
```

when I try to .apply a function to the Amount column I get the following error.

```
ValueError: cannot convert float NaN to integer
```

I have tried applying a function using .isnan from the Math Module I have tried the pandas .replace attribute I tried the .sparse data attribute from pandas 0.9 I have also tried if NaN == NaN statement in a function. I have also looked at this article How do I replace NA values with zeros in an R dataframe? whilst looking at some other articles. All the methods I have tried have not worked or do not recognise NaN. Any Hints or solutions would be appreciated.

python pandas dataframe

edited Oct 20 '18 at 10:22
Maven Carvalho
305 3 12

asked Nov 8 '12 at 18:50
George Thompson
1,824 2 11 14

The only problem is df.fill.na() does not work if the data frame on which

8 Answers

539

I believe `DataFrame.fillna()` will do this for you.

Link to Docs for [a dataframe](#) and for [a Series](#).

Example:

```
In [7]: df
Out[7]:
          0         1
0       NaN       NaN
1 -0.494375  0.570994
2       NaN       NaN
3  1.876360 -0.229738
4       NaN       NaN

In [8]: df.fillna(0)
Out[8]:
          0         1
0  0.000000  0.000000
1 -0.494375  0.570994
2  0.000000  0.000000
3  1.876360 -0.229738
4  0.000000  0.000000
```

To fill the NaNs in only one column, select just that column. in this case I'm using inplace=True to actually change the contents of df.

```
In [12]: df[1].fillna(0, inplace='
Out[12]:
0    0.000000
1    0.570994
2    0.000000
3   -0.229738
4    0.000000
Name: 1

In [13]: df
Out[13]:
          0         1
0       NaN  0.000000
1 -0.494375  0.570994
2       NaN  0.000000
3  1.876360 -0.229738
4       NaN  0.000000
```

edited Jun 23 '16 at 17:29
jeremycg
19.1k 4 42 57

answered Nov 8 '12 at 18:54
Aman
25.5k 6 25 35

1 Is it guaranteed that `df[1]` is a view rather than a copy of the original DF? Obviously, if there's a rare situation where it's a copy, it would

@max See this, might address your question: stackoverflow.com/questions/23296282/... – Aman Feb 3 '16 at 1:23

Thanks. Is my understanding correct that in that answer an "indexer that sets" is the outermost indexing operation (executed just before the assignment. So any assignment that only uses a single indexer is guaranteed to be safe, making your code safe? – max Feb 3 '16 at 16:01

@max I do not know what you mean by "safe"... but in any case, this seems off topic here. :) Probably best to comment on that other question, or post a new question. – Aman Feb 3 '16 at 18:51

1 Why is this not working for me? see: stackoverflow.com/questions/39452095/how-to-fillna-with-value-0 – displayname Sep 12 '16 at 13:59

79

It is not guaranteed that the slicing returns a view or a copy. You can do

```
df['column'] = df['column'].fillna
```

edited Oct 7 '18 at 19:25

A-B-B
24.5k 6 64 70

answered Oct 6 '16 at 9:10

rakesh
2,045 11 12

7 Just discovered the "inplace=True" problem. This answer avoids the issue and I think is the cleanest solution presented. – TimCera Apr 28 '17 at 13:53

20

I just wanted to provide a bit of an update/special case since it looks like people still come here. If you're using a multi-index or otherwise using an index-slicer the inplace=True option may not be enough to update the slice you've chosen. For example in a 2x2 level multi-index this will not

```
idx = pd.IndexSlice
df.loc[idx[:,mask_1],idx[mask_2,:]
```

The "problem" is that the chaining breaks the fillna ability to update the original dataframe. I put "problem" in quotes because there are good reasons for the design decisions that led to not interpreting through these chains in certain situations. Also, this is a complex example (though I really ran into it), but the same may apply to fewer levels of indexes depending on how you slice.

The solution is DataFrame.update:

```
df.update(df.loc[idx[:,mask_1],idx
```

It's one line, reads reasonably well (sort of) and eliminates any unnecessary messing with intermediate variables or loops while allowing you to apply fillna to any multi-level slice you like!

If anybody can find places this doesn't work please post in the comments, I've been messing with it and looking at the source and it seems to solve at least my multi-index slice problems.

edited Dec 16 '15 at 18:29
Karalga
175 10

answered Jun 2 '15 at 5:13
Ezekiel Kruglick
3,056 27 36

19

You could use `replace` to change `NaN` to `0` :

```
import pandas as pd
import numpy as np

# for column
df['column'] = df['column'].replac

# for whole dataframe
df = df.replace(np.nan, 0)

# inplace
df.replace(np.nan, 0, inplace=True
```

answered Jun 15 '17 at 5:11
Anton Protopopov
15.5k 3 49 60

18

```
import pandas

df = pandas.read_csv('somefile.txt

df = df.fillna(0)
```

edited Sep 13 '16 at 21:13
Petter Friberg
16.4k 8 39 75

answered Sep 13 '16 at 20:59
Cornel Ciobanu
301 3 3

3

Easy way to fill the missing values:-

filling string columns: when string columns have missing values and NaN values.

```
df['string column name'].fillna(df
inplace = True)
```

filling numeric columns: when the numeric columns have missing values and NaN values.

```
df['numeric column name'].fillna(d
True)
```

filling NaN with zero:

```
df['column name'].fillna(0, inplac
```

edited Jul 7 '18 at 19:03
Martin
2,249 7 22 26

answered Jul 7 '18 at 18:31
tulsi kumar
81 1 4

0

```
      itm Date                  Amount
67    420 2012-09-30 00:00:00   65211
68    421 2012-09-09 00:00:00   29424
69    421 2012-09-16 00:00:00   29877
70    421 2012-09-23 00:00:00   30990
71    421 2012-09-30 00:00:00   61303
72    485 2012-09-09 00:00:00   71781
73    485 2012-09-16 00:00:00     NaN
74    485 2012-09-23 00:00:00   11072
75    485 2012-09-30 00:00:00  113702
76    489 2012-09-09 00:00:00   64731
77    489 2012-09-16 00:00:00     NaN
```

Considering the particular column `Amount` in the above table is of

Similarly, you can fill it with various data types like `float` , `str` and so on.

In particular, I would consider datatype to compare various values of the same column.

edited Feb 26 at 11:41
tuomastik
2,281 1 21 31

answered Feb 26 at 11:21
Bharath_Raja
25 8

0

To replace na values in pandas

```
df['column_name'].fillna(value_to_
```

if `inplace = False` , instead of updating the df (dataframe) it will return the modified values.

answered Mar 29 at 19:46
Vivek Ananthan
1,297 1 17 31

---

protected by Serenity Jul 7 '18 at 12:34

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?