How to drop rows of Pandas DataFrame whose value in certain columns is NaN

COMPOSE
AN IBM COMPANY
Get a Free Compose T-Shirt
when you sign up for a hosted, production-ready database.
START FREE

I have a `DataFrame` :

```
>>> df
                 STK_ID  EPS  cash
STK_ID RPT_Date
601166 20111231  601166  NaN   NaN
600036 20111231  600036  NaN    12
600016 20111231  600016  4.3   NaN
601009 20111231  601009  NaN   NaN
601939 20111231  601939  2.5   NaN
000001 20111231  000001  NaN   NaN
```

Then I just want the records whose `EPS` is not `NaN` , that is, `df.drop(....)` will return the dataframe as below:

```
                  STK_ID  EPS  cash
STK_ID RPT_Date
600016 20111231  600016  4.3   NaN
601939 20111231  601939  2.5   NaN
```

How do I do that?

python pandas dataframe

edited Jan 5 at 17:01
Ninjakannon
2,192 3 20 38

asked Nov 16 '12 at 9:17
bigbug
6,595 19 53 75

11 dropna: pandas.pydata.org/pandas-docs/stable/generated/… – Wouter Overmeire Nov 16 '12 at 9:29

56 `df.dropna(subset = ['column1_name', 'column2_name', 'column3_name'])` – osa Sep 5 '14 at 23:53

7 Answers

Don't `drop` . Just take rows where `EPS` is **finite**:

```
df = df[np.isfinite(df['EPS'])]
```

answered Nov 16 '12 at 9:34
eumiro
95.9k 9 178 204

189 I'd recommend using `pandas.notnull` instead of `np.isfinite` – Wes McKinney Nov 21 '12 at 3:08

5 @shootingstars The docs say that pandas.notnull is a direct replacement for np.isfinite. In this case, null does not mean zero. – semi-extrinsic Mar 5 '15 at 10:52

4 Is there any advantage to indexing and copying over dropping? – Robert Muil Jul 31 '15 at 8:15

Creates Error: TypeError: ufunc 'isfinite' not supported for the input types, and the inputs could not be safely coerced to any supported types according to the casting rule "safe" – Philipp Schwarz Oct 7 '16 at 13:18

This question is already resolved, but...

...also consider the solution suggested by Wouter in his original comment. The ability to handle missing data, including `dropna()` , is built into pandas explicitly. Aside from potentially improved performance over doing it manually, these functions also come with a variety of options which may be useful.

```
In [24]: df = pd.DataFrame(np.random.randn(10,3))

In [25]: df.ix[::2,0] = np.nan; df.ix[::4,1] = np.nan; df.ix[::3,2] = np.nan;

In [26]: df
Out[26]:
          0         1         2
0       NaN       NaN       NaN
1  2.677677 -1.466923 -0.750366
2       NaN  0.798002 -0.906038
3  0.672201  0.964789       NaN
4       NaN       NaN  0.050742
5 -1.250970  0.030561 -2.678622
6       NaN  1.036043       NaN
7  0.049896 -0.308003  0.823295
8       NaN       NaN  0.637482
9 -0.310130  0.078891       NaN



In [27]: df.dropna()     #drop all rows that have any NaN values
Out[27]:
          0         1         2
1  2.677677 -1.466923 -0.750366
5 -1.250970  0.030561 -2.678622
7  0.049896 -0.308003  0.823295



In [28]: df.dropna(how='all')     #drop only if ALL columns are NaN
Out[28]:
          0         1         2
1  2.677677 -1.466923 -0.750366
2       NaN  0.798002 -0.906038
3  0.672201  0.964789       NaN
4       NaN       NaN  0.050742
5 -1.250970  0.030561 -2.678622
6       NaN  1.036043       NaN
7  0.049896 -0.308003  0.823295
8       NaN       NaN  0.637482
9 -0.310130  0.078891       NaN



In [29]: df.dropna(thresh=2)   #Drop row if it does not have at least two values that are
**not** NaN
Out[29]:
          0         1         2
1  2.677677 -1.466923 -0.750366
2       NaN  0.798002 -0.906038
3  0.672201  0.964789       NaN
5 -1.250970  0.030561 -2.678622
7  0.049896 -0.308003  0.823295
9 -0.310130  0.078891       NaN



In [30]: df.dropna(subset=[1])   #Drop only if NaN in specific column (as asked in the
question)
Out[30]:
          0         1         2
1  2.677677 -1.466923 -0.750366
2       NaN  0.798002 -0.906038
3  0.672201  0.964789       NaN
5 -1.250970  0.030561 -2.678622
6       NaN  1.036043       NaN
7  0.049896 -0.308003  0.823295
9 -0.310130  0.078891       NaN
```

There are also other options (See docs at http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.dropna.html), including dropping columns instead of rows.

Pretty handy!

edited May 23 at 12:26
Community ♦
1 1

answered Nov 17 '12 at 20:27
Aman
14.3k 5 21 31

70 you can also use `df.dropna(subset = ['column_name'])` . Hope that saves at least one person the extra 5 seconds of 'what am I doing wrong'. Great answer, +1 – James Tobin Jun 18 '14 at 14:07

6 @JamesTobin, I just spent 20 minutes to write a function for that! The official documentation was very cryptic: "Labels along other axis to consider, e.g. if you are dropping rows these would be a list of columns to include". I was unable to understand, what they meant... – osa Sep 5 '14 at 23:52

I know this has already been answered, but just for the sake of a purely pandas solution to this specific question as opposed to the general description from Aman (which was wonderful) and in case anyone else happens upon this:

```
import pandas as pd
df = df[pd.notnull(df['EPS'])]
```

answered Apr 23 '14 at 5:37
Frederick Hadley
686 5 2

6 Actually, the specific answer would be: `df.dropna(subset=['EPS'])` (based on the general description of Aman, of course this does also work) – joris Apr 23 '14 at 12:53

1 `notnull` is also what Wes (author of Pandas) suggested in his comment on another answer. – fantabolous Jul 9 '14 at 3:24

This maybe a noob question. But when I do a df[pd.notnull(...) or df.dropna the index gets dropped. So if there was a null value in row-index 10 in a df of length 200. The dataframe after running the drop function has index values from 1 to 9 and then 11 to 200. Anyway to "re-index" it – Aakash Gupta Mar 4 '16 at 6:03

You could use dataframe method notnull or inverse of isnull, or numpy.isnan:

```
In [332]: df[df.EPS.notnull()]
Out[332]:
   STK_ID  RPT_Date  STK_ID.1  EPS  cash
2  600016  20111231    600016  4.3   NaN
4  601939  20111231    601939  2.5   NaN


In [334]: df[~df.EPS.isnull()]
Out[334]:
   STK_ID  RPT_Date  STK_ID.1  EPS  cash
2  600016  20111231    600016  4.3   NaN
4  601939  20111231    601939  2.5   NaN


In [347]: df[~np.isnan(df.EPS)]
Out[347]:
   STK_ID  RPT_Date  STK_ID.1  EPS  cash
2  600016  20111231    600016  4.3   NaN
4  601939  20111231    601939  2.5   NaN
```

answered Dec 4 '15 at 7:01
Anton Protopopov
9,529 22 39

notnull is very nice! – Rustam Apr 14 '16 at 10:05

yet another solution which uses the fact that `np.nan != np.nan` :

```
In [149]: df.query("EPS == EPS")
Out[149]:
                 STK_ID  EPS  cash
STK_ID RPT_Date
600016 20111231  600016  4.3   NaN
601939 20111231  601939  2.5   NaN
```

answered Apr 20 at 21:15
MaxU
58.9k 6 27 54

For some reason none of the previously submitted answers worked for me. This basic solution did:

```
df = df[df.EPS >= 0]
```

Though of course that will drop rows with negative numbers, too. So if you want those it's probably smart to add this after, too.

```
df = df[df.EPS <= 0]
```

edited Oct 9 '15 at 18:25

answered Oct 9 '15 at 18:00
samthebrand ♦
649 11 26

It may be added at that '&' can be used to add additional conditions e.g.

```
df = df[(df.EPS > 2.0) & (df.EPS <4.0)]
```

Notice that when evaluating the statements, pandas needs parenthesis.

edited Jan 26 at 23:12
aesede
2,640 21 23

answered Mar 15 '16 at 15:33
David
1

Sorry, but OP want someting else. Btw, your code is wrong, return `ValueError: The truth value of a Series is ambiguous. Use a.empty, a.bool(), a.item(), a.any() or a.all().` . You need add parenthesis - `df = df[(df.EPS > 2.0) & (df.EPS <4.0)]` , but also it is not answer for this question. – jezrael Mar 16 '16 at 11:52
