

# Plot correlation matrix using pandas

Asked 6 years, 6 months ago   Active 3 months ago   Viewed 572k times



263



I have a data set with huge number of features, so analysing the correlation matrix has become very difficult. I want to plot a correlation matrix which we get using `dataframe.corr()` function from pandas library. Is there any built-in function provided by the pandas library to plot this matrix?



123

[python](#) [pandas](#) [matplotlib](#) [data-visualization](#) [information-visualization](#)



Share Follow

edited Mar 27 '19 at 16:30



Fabian Rost

1,844 2 12 27

asked Apr 3 '15 at 12:57



Gaurav Singh

9,872 4 20 24

- 1 Related answers can be found here [Making heatmap from pandas DataFrame](#) – [joelostblom](#) Jul 13 '18 at 13:20

14 Answers


Active


Oldest

Votes

**Join Stack Overflow** to learn, share knowledge, and build your career.

Sign up with email

 Sign up with Google

 Sign up with GitHub

 Sign up with Facebook





You can use `pyplot.matshow()` from `matplotlib`:

372

```
import matplotlib.pyplot as plt
```



```
plt.matshow(dataframe.corr())  
plt.show()
```



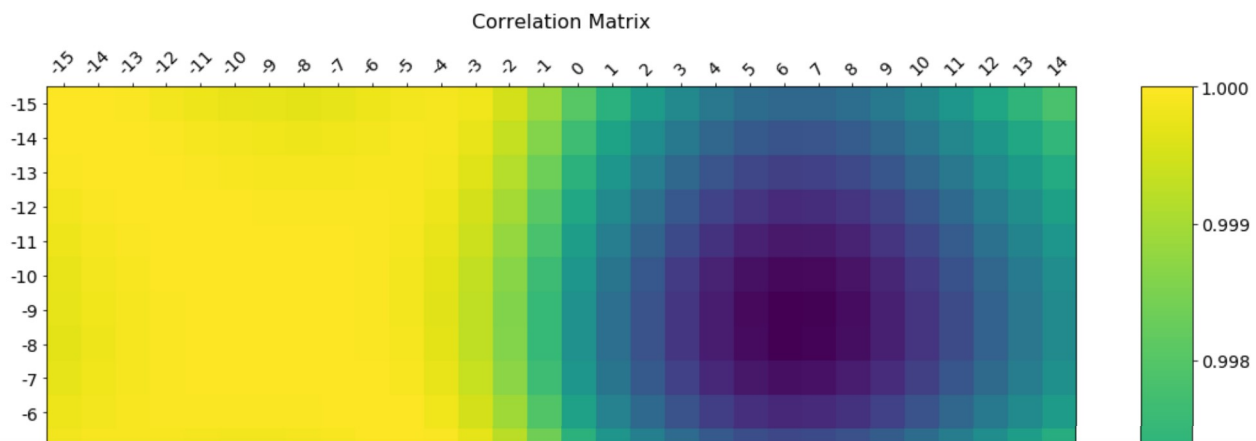
Edit:

In the comments was a request for how to change the axis tick labels. Here's a deluxe version that is drawn on a bigger figure size, has axis labels to match the dataframe, and a colorbar legend to interpret the color scale.

I'm including how to adjust the size and rotation of the labels, and I'm using a figure ratio that makes the colorbar and the main figure come out the same height.


EDIT 2: As the `df.corr()` method ignores non-numerical columns, `.select_dtypes(['number'])` should be used when defining the x and y labels to avoid an unwanted shift of the labels (included in the code below).

```
f = plt.figure(figsize=(19, 15))  
plt.matshow(df.corr(), fignum=f.number)  
plt.xticks(range(df.select_dtypes(['number']).shape[1]),  
df.select_dtypes(['number']).columns, fontsize=14, rotation=45)  
plt.yticks(range(df.select_dtypes(['number']).shape[1]),  
df.select_dtypes(['number']).columns, fontsize=14)  
cb = plt.colorbar()  
cb.ax.tick_params(labelsize=14)  
plt.title('Correlation Matrix', fontsize=16);
```



**Join Stack Overflow** to learn, share knowledge, and build your career.

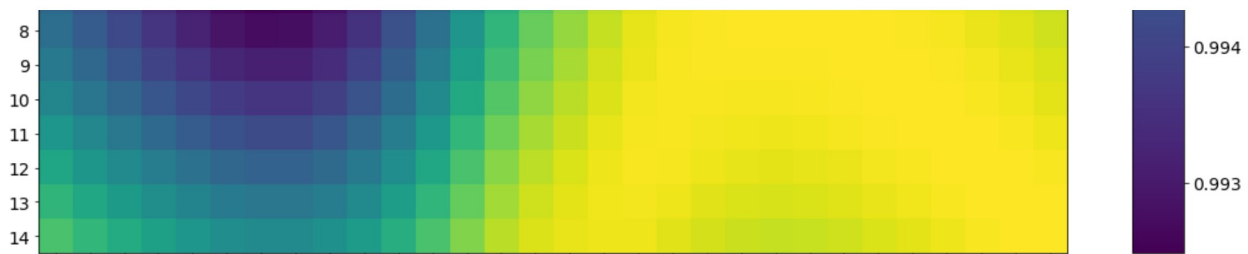
Sign up with email

 Sign up with Google

 Sign up with GitHub

 Sign up with Facebook





Share Follow

edited Jan 8 at 19:10



NicoH

725 2 6 16

answered Apr 3 '15 at 13:04




jrjc

18k 8 58 70

- 1 I must be missing something: `AttributeError: 'module' object has no attribute 'matshow'`  
– [Tom Russell](#) May 16 '18 at 22:51 ✎
- 1 @TomRussell Did you do `import matplotlib.pyplot as plt`? – [joelostblom](#) Jun 5 '18 at 15:23
- 11 do you know how to display the actual column names on the plot? – [WebQube](#) Jan 4 '19 at 12:13
- 2 @Cecilia I had resolved this matter by changing the **rotation** parameter to **90** – [lkbel benab](#) Nov 4 '19 at 16:12
- 2 With columns names longer than those, the x labels will look a bit off, in my case it was confusing as they looked shifted by one tick. Adding `ha="left"` to the `plt.xticks` call solved this problem, in case anyone has it as well :) described in [stackoverflow.com/questions/28615887/...](https://stackoverflow.com/questions/28615887/...) – [V. Déhaye](#) Apr 20 '20 at 15:44 ✎

Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email

 Sign up with Google

 Sign up with GitHub

 Sign up with Facebook

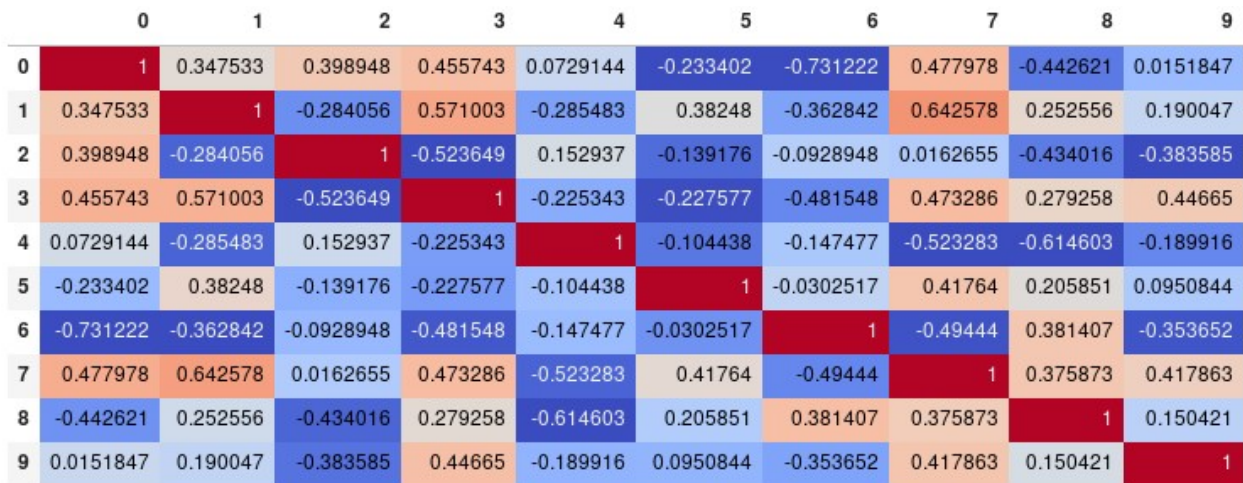


280

If your main goal is to visualize the correlation matrix, rather than creating a plot per se, the convenient pandas [styling options](#) is a viable built-in solution:

```
import pandas as pd
import numpy as np

rs = np.random.RandomState(0)
df = pd.DataFrame(rs.rand(10, 10))
corr = df.corr()
corr.style.background_gradient(cmap='coolwarm')
# 'RdBu_r', 'BrBG_r', & 'PuOr_r' are other good diverging colormaps
```

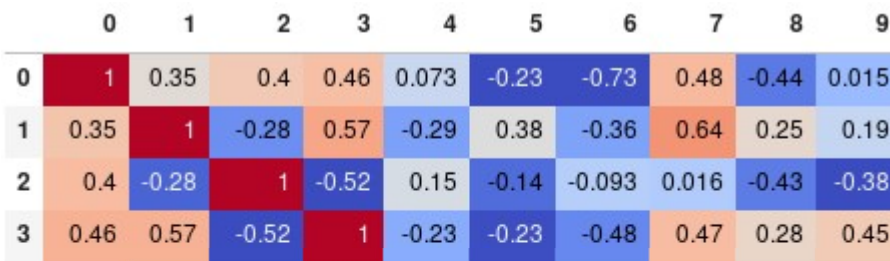


Note that this needs to be in a backend that supports rendering HTML, such as the JupyterLab Notebook.

## Styling


You can easily limit the digit precision:

```
corr.style.background_gradient(cmap='coolwarm').set_precision(2)
```




**Join Stack Overflow** to learn, share knowledge, and build your career.

Sign up with email

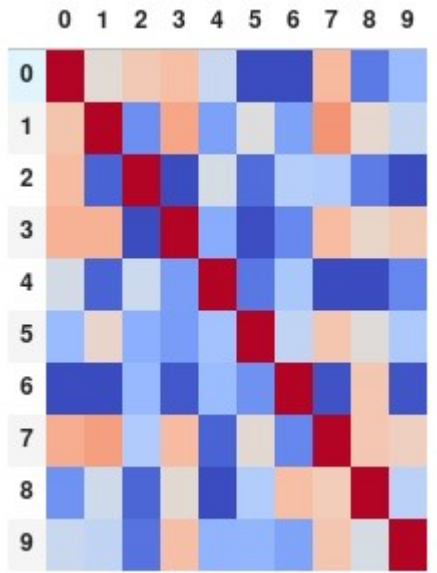
 Sign up with Google

 Sign up with GitHub

 Sign up with Facebook



```
corr.style.background_gradient(cmap='coolwarm').set_properties(**{'font-size': '0pt'})
```



The styling documentation also includes instructions of more advanced styles, such as how to change the display of the cell the mouse pointer is hovering over.

## Time comparison

In my testing, `style.background_gradient()` was 4x faster than `plt.matshow()` and 120x faster than `sns.heatmap()` with a 10x10 matrix. Unfortunately it doesn't scale as well as `plt.matshow()`: the two take about the same time for a 100x100 matrix, and `plt.matshow()` is 10x faster for a 1000x1000 matrix.

## Saving


There are a few possible ways to save the stylized dataframe:


- Return the HTML by appending the `render()` method and then write the output to a file.
- Save as an `.xlsx` file with conditional formatting by appending the `to_excel()` method.
- [Combine with imgkit to save a bitmap](#)
- Take a screenshot (like I have done here).

---

**Join Stack Overflow** to learn, share knowledge, and build your career.

Sign up with email

 Sign up with Google

 Sign up with GitHub

 Sign up with Facebook



|   |           |           |            |           |           |            |            |           |           |           |
|---|-----------|-----------|------------|-----------|-----------|------------|------------|-----------|-----------|-----------|
| 0 | 1         | 0.347533  | 0.398948   | 0.455743  | 0.0729144 | -0.233402  | -0.731222  | 0.477978  | -0.442621 | 0.0151847 |
| 1 | 0.347533  | 1         | -0.284056  | 0.571003  | -0.285483 | 0.38248    | -0.362842  | 0.642578  | 0.252556  | 0.190047  |
| 2 | 0.398948  | -0.284056 | 1          | -0.523649 | 0.152937  | -0.139176  | -0.0928948 | 0.0162655 | -0.434016 | -0.383585 |
| 3 | 0.455743  | 0.571003  | -0.523649  | 1         | -0.225343 | -0.227577  | -0.481548  | 0.473286  | 0.279258  | 0.44665   |
| 4 | 0.0729144 | -0.285483 | 0.152937   | -0.225343 | 1         | -0.104438  | -0.147477  | -0.523283 | -0.614603 | -0.189916 |
| 5 | -0.233402 | 0.38248   | -0.139176  | -0.227577 | -0.104438 | 1          | -0.0302517 | 0.41764   | 0.205851  | 0.0950844 |
| 6 | -0.731222 | -0.362842 | -0.0928948 | -0.481548 | -0.147477 | -0.0302517 | 1          | -0.49444  | 0.381407  | -0.353652 |
| 7 | 0.477978  | 0.642578  | 0.0162655  | 0.473286  | -0.523283 | 0.41764    | -0.49444   | 1         | 0.375873  | 0.417863  |
| 8 | -0.442621 | 0.252556  | -0.434016  | 0.279258  | -0.614603 | 0.205851   | 0.381407   | 0.375873  | 1         | 0.150421  |
| 9 | 0.0151847 | 0.190047  | -0.383585  | 0.44665   | -0.189916 | 0.0950844  | -0.353652  | 0.417863  | 0.150421  | 1         |

## Single corner heatmap


Since many people are reading this answer I thought I would add a tip for how to only show one corner of the correlation matrix. I find this easier to read myself, since it removes the redundant information.


```
# Fill diagonal and upper half with NaNs
mask = np.zeros_like(corr, dtype=bool)
mask[np.triu_indices_from(mask)] = True
corr[mask] = np.nan
(corr
 .style
 .background_gradient(cmap='coolwarm', axis=None, vmin=-1, vmax=1)
 .highlight_null(null_color='#f1f1f1') # Color NaNs grey
 .set_precision(2))
```



Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email



 Sign up with Google

 Sign up with GitHub

 Sign up with Facebook





- 2 If there was a way to export is as an image, that would have been great! – [Kristada673](#) Jun 27 '18 at 4:43
- 1 Thanks! You definitely need a diverging palette `import seaborn as sns corr = df.corr() cm = sns.light_palette("green", as_cmap=True) cm = sns.diverging_palette(220, 20, sep=20, as_cmap=True) corr.style.background_gradient(cmap=cm).set_precision(2)` – [stallingOne](#) Jul 5 '18 at 9:00 
- 1 @stallingOne Good point, I shouldn't have included negative values in the example, I might change that later. Just for reference for people reading this, you don't need to create a custom divergent cmap with seaborn (although the one in the comment above looks pretty slick), you can also use the built-in divergent cmaps from matplotlib, e.g. `corr.style.background_gradient(cmap='coolwarm')`. There is currently no way to center the cmap on a specific value, which can be a good idea with divergent cmaps. – [joelostblom](#) Jul 5 '18 at 13:54 
- 1 @rovyko Are you on pandas >=0.24.0? – [joelostblom](#) Mar 6 '19 at 19:17
- 2 These plots are visually great, but @Kristada673 question is quite relevant, how would you export them? – [Erfan](#) May 15 '19 at 16:31

Seaborn's heatmap version:

107

```
import seaborn as sns
corr = dataframe.corr()
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)
```

Share Follow


answered Oct 24 '16 at 22:45




5,647 3 30 34

**Join Stack Overflow** to learn, share knowledge, and build your career.

Sign up with email

 Sign up with Google

 Sign up with GitHub

 Sign up with Facebook



12 Seaborn heatmap is fancy but it performs poor on large matrices. matshow method of matplotlib is much faster. – [anilbey](#) Aug 22 '17 at 22:28

4 Seaborn can automatically infer the ticklabels from the column names. – [Tulio Casagrande](#) Oct 2 '18 at 21:32

It seems that not all ticklabels are shown always if seaborn is left to automatically infer [stackoverflow.com/questions/50754471/...](https://stackoverflow.com/questions/50754471/...) – [janto](#) Sep 14 at 15:21 

Try this function, which also displays variable names for the correlation matrix:

98

```
def plot_corr(df,size=10):  
    """Function plots a graphical correlation matrix for each pair of columns in the  
    dataframe.
```

Input:

df: pandas DataFrame

size: vertical and horizontal size of the plot

```
"""
```

```
corr = df.corr()  
fig, ax = plt.subplots(figsize=(size, size))  
ax.matshow(corr)  
plt.xticks(range(len(corr.columns)), corr.columns)  
plt.yticks(range(len(corr.columns)), corr.columns)
```

Share Follow

edited Jun 22 at 20:43



Community Bot


1 1

answered Jul 13 '15 at 13:10



Apogentus

5,713 2 28 31


7 plt.xticks(range(len(corr.columns)), corr.columns, rotation='vertical') if you want vertical orientation of column names on x-axis – [nishant](#) Feb 18 '19 at 8:38 

Another graphical thing, but adding a plt.tight\_layout() might also be useful for long column names. – [user3017048](#) May 28 '19 at 6:18

Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email

 Sign up with Google

 Sign up with GitHub

 Sign up with Facebook







You can observe the relation between features either by drawing a heat map from seaborn or scatter matrix from pandas.

88



Scatter Matrix:



```
pd.scatter_matrix(dataframe, alpha = 0.3, figsize = (14,8), diagonal = 'kde');
```

If you want to visualize each feature's skewness as well - use seaborn pairplots.

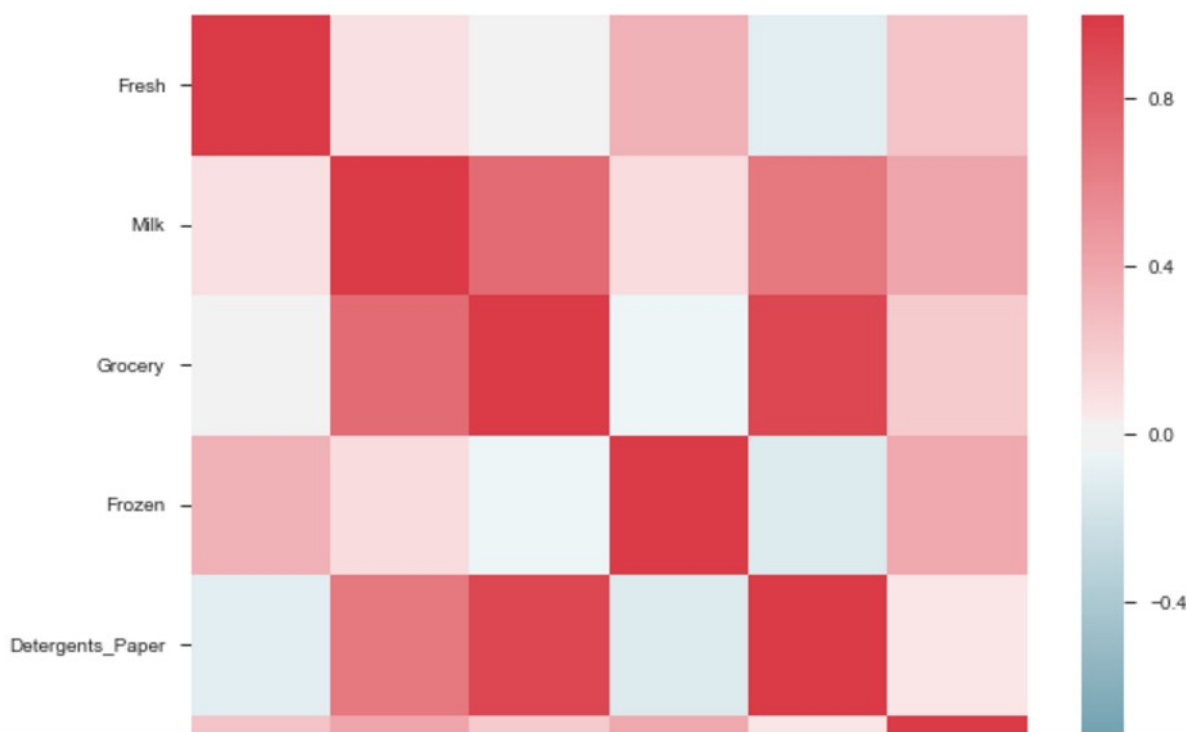
```
sns.pairplot(dataframe)
```

Sns Heatmap:

```
import seaborn as sns


f, ax = pl.subplots(figsize=(10, 8))
corr = dataframe.corr()
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool),
            cmap=sns.diverging_palette(220, 10, as_cmap=True),
            square=True, ax=ax)
```


The output will be a correlation map of the features. i.e. see the below example.




**Join Stack Overflow** to learn, share knowledge, and build your career.

Sign up with email

 Sign up with Google

 Sign up with GitHub

 Sign up with Facebook



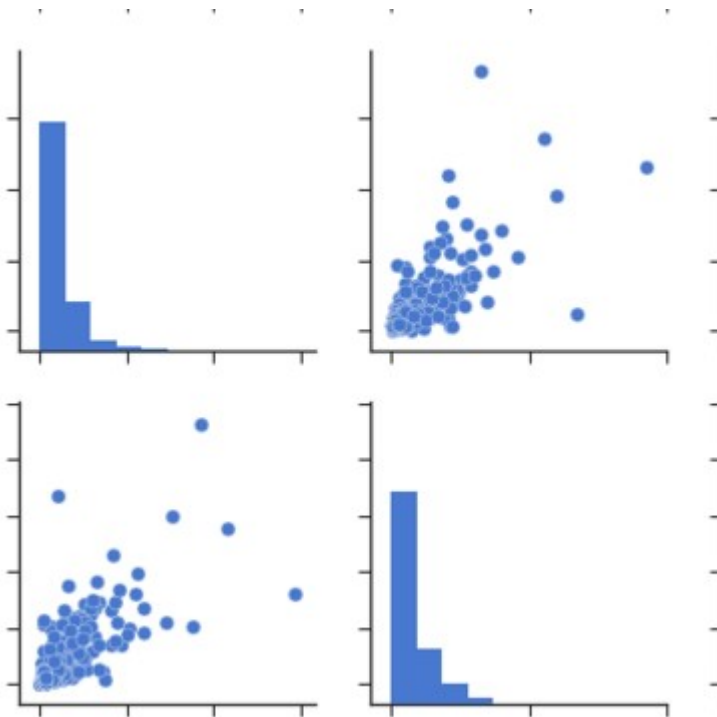
Products With Medium Correlation:

1. Milk and Grocery
2. Milk and Detergents\_Paper

Products With Low Correlation:

1. Milk and Deli
2. Frozen and Fresh.
3. Frozen and Deli.

From Pairplots: You can observe same set of relations from pairplots or scatter matrix. But from these we can say that whether the data is normally distributed or not.



Note: The above is same graph taken from the data, which is used to draw heatmap.

Share Follow

edited Mar 23 '17 at 13:54

answered Mar 23 '17 at 13:48





phanindravarma

1,117 1 8 8

**Join Stack Overflow** to learn, share knowledge, and build your career.

Sign up with email

 Sign up with Google

 Sign up with GitHub

 Sign up with Facebook



For completeness, the simplest solution i know with [seaborn](#) as of late 2019, if one is using [Jupyter](#):

17

```
import seaborn as sns
sns.heatmap(dataframe.corr())
```



Share Follow

answered Nov 8 '19 at 8:01

[Marcin](#)**3,474** 1 23 47

If you dataframe is `df` you can simply use:

10

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(15, 10))
sns.heatmap(df.corr(), annot=True)
```



Share Follow

answered Aug 15 '19 at 21:06

[Harvey](#)**8,136** 4 55 61

You can use `imshow()` method from `matplotlib`

9

```
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('ggplot')

plt.imshow(X.corr(), cmap=plt.cm.Reds, interpolation='nearest')
plt.colorbar()
tick_marks = [i for i in range(len(X.columns))]
plt.xticks(tick_marks, X.columns, rotation='vertical')
plt.yticks(tick_marks, X.columns)
plt.show()
```



Share Follow

edited Jun 19 '19 at 5:20

[Ralph Deint](#)**140** 1 2 14

answered Jun 28 '18 at 16:02

[Khandelwal-manik](#)**323** 3 9

**Join Stack Overflow** to learn, share knowledge, and build your career.

Sign up with email

Sign up with Google

Sign up with GitHub

Sign up with Facebook



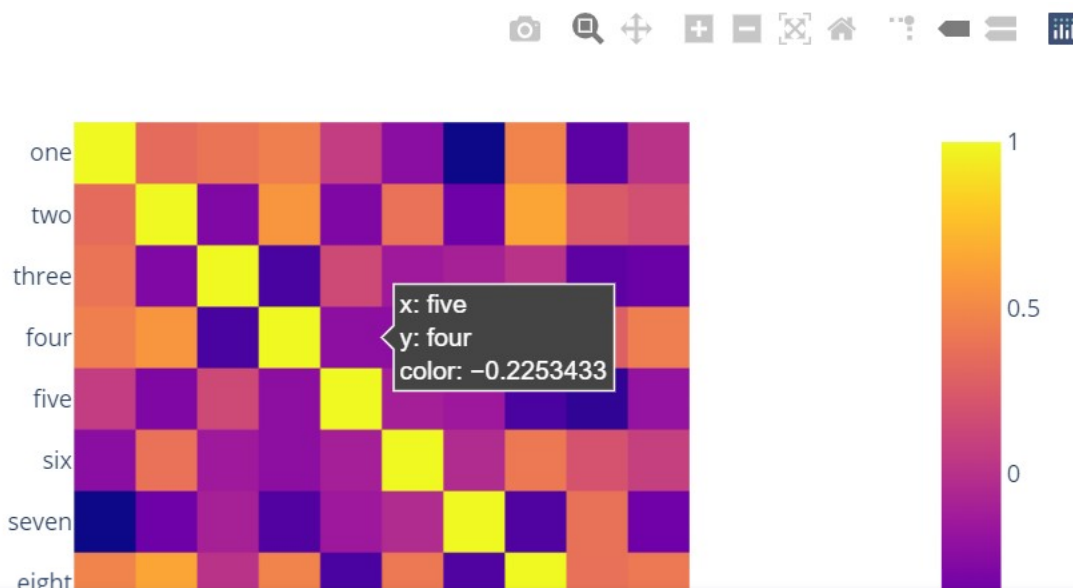
Surprised to see no one mentioned more capable, interactive and easier to use alternatives.

6

## A) You can use plotly:


1. Just two lines and you get:
2. interactivity,
3. smooth scale,
4. colors based on whole dataframe instead of individual columns,
5. column names & row indices on axes,
6. zooming in,
7. panning,
8. built-in one-click ability to save it as a PNG format,
9. auto-scaling,
10. comparison on hovering,
11. bubbles showing values so heatmap still looks good and you can see values wherever you want:


```
import plotly.express as px
fig = px.imshow(df.corr())
fig.show()
```




**Join Stack Overflow** to learn, share knowledge, and build your career.

Sign up with email

 Sign up with Google

 Sign up with GitHub

 Sign up with Facebook




All the same functionality with a tad much hassle. But still worth it if you do not want to opt-in for plotly and still want all these things:

```
from bokeh.plotting import figure, show, output_notebook
from bokeh.models import ColumnDataSource, LinearColorMapper
from bokeh.transform import transform
output_notebook()
colors = ['#d7191c', '#fdae61', '#ffffbf', '#a6d96a', '#1a9641']
TOOLS = "hover,save,pan,box_zoom,reset,wheel_zoom"
data = df.corr().stack().rename("value").reset_index()
p = figure(x_range=list(df.columns), y_range=list(df.index), tools=TOOLS,
           toolbar_location='below',
           tooltips=[('Row, Column', '@level_0 x @level_1'), ('value', '@value')],
           height = 500, width = 500)

p.rect(x="level_1", y="level_0", width=1, height=1,
       source=data,
       fill_color={'field': 'value', 'transform': LinearColorMapper(palette=colors,
                                                                    low=data.value.min(), high=data.value.max())},
       line_color=None)
color_bar = ColorBar(color_mapper=LinearColorMapper(palette=colors,
                                                    low=data.value.min(), high=data.value.max()), major_label_text_font_size="7px",
                    ticker=BasicTicker(desired_num_ticks=len(colors)),
                    formatter=PrintfTickFormatter(format="%f"),
                    label_standoff=6, border_line_color=None, location=(0, 0))
p.add_layout(color_bar, 'right')


show(p)
```


 BokehJS 2.1.1 successfully loaded.




**Join Stack Overflow** to learn, share knowledge, and build your career.

Sign up with email

 Sign up with Google

 Sign up with GitHub

 Sign up with Facebook



[one](#)[two](#)[three](#)[four](#)[five](#)[six](#)[seven](#)[eight](#)[nine](#)[ten](#)[Share](#) [Follow](#)

answered Nov 26 '20 at 0:22

[Hamza](#)**3,342**

2

18

30

---

1    Good answer. It's lot easier. – [Gaurav Koradiya](#) Mar 23 at 18:07

---

statmodels graphics also gives a nice view of correlation matrix

3

```
import statsmodels.api as sm
import matplotlib.pyplot as plt
```

```
corr = dataframe.corr()
sm.graphics.plot_corr(corr, xnames=list(corr.columns))
plt.show()
```

[Share](#) [Follow](#)

answered Oct 18 '19 at 5:07

[Shahriar Miraj](#)**168**

1

11

---

**Join Stack Overflow** to learn, share knowledge, and build your career.

[Sign up with email](#)[Sign up with Google](#)[Sign up with GitHub](#)[Sign up with Facebook](#)



Along with other methods it is also good to have pairplot which will give scatter plot for all the cases-

3

```
import pandas as pd
import numpy as np
import seaborn as sns
rs = np.random.RandomState(0)
df = pd.DataFrame(rs.rand(10, 10))
sns.pairplot(df)
```

Share Follow

answered Jan 24 '20 at 7:11



Nishant Tyagi

31 2

Form correlation matrix, in my case zdf is the dataframe which i need perform correlation matrix.

1

```
corrMatrix =zdf.corr()
corrMatrix.to_csv('sm_zscaled_correlation_matrix.csv');
html = corrMatrix.style.background_gradient(cmap='RdBu').set_precision(2).render()

# Writing the output to a html file.
with open('test.html', 'w') as f:
    print('<!DOCTYPE html><html lang="en"><head><meta charset="UTF-8"><meta
name="viewport" content="width=device-widthinitial-scale=1.0"><title>Document</title>
</head><style>table{word-break: break-all;}</style><body>' + html+'</body></html>',
file=f)
```

Then we can take screenshot. or convert html to an image file.

Share Follow

answered Mar 5 '20 at 4:56



smsivaprakaash

1,138 10 10

**Join Stack Overflow** to learn, share knowledge, and build your career.

Sign up with email

Sign up with Google

Sign up with GitHub

Sign up with Facebook



You can use `heatmap()` from `seaborn` to see the correlation b/w different features:

-1

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
co_matrices=dataframe.corr()
plot.figure(figsize=(15,20))
sns.heatmap(co_matrix, square=True, cbar_kws={"shrink": .5})
```

Share Follow

edited Apr 24 at 19:37



10 Rep

2,107 7 15 29

answered Apr 24 at 17:58



Reyan Ishtiaq

9 2

Please check below readable code

-1

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(36, 26))
heatmap = sns.heatmap(df.corr(), vmin=-1, vmax=1, annot=True)
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':12}, pad=12)``
```

[1]: <https://i.stack.imgur.com/I5SeR.png>

Share Follow

answered May 5 at 11:57





chetan wankhede


1 1

Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email

 Sign up with Google

 Sign up with GitHub

 Sign up with Facebook

