



SEABORN, SQUARIFY, MISSINGNO LIBRARIES

ArcGIS Pro



ناهید نعمتی کوتنائی (تیسا)
دکتری جغرافیا و برنامه‌ریزی شهری
مدرس دانشگاه

 **Dr.nemati.K**
 **@Nemati_k**
 **09112230798**



محمدطاهر طاهرپور
دانشجوی ارشد مدیریت شهری
دانشگاه تهران

 **mtaherpoor**
 **@mtaherpoor**
 **09336144947**



فهرست مطالب:

۳seaborn, squarify, missingno Libraries
۳ Seaborn چیه؟
۳کنترل زیبایی فیگورها
۴کنترل سبک یا style فیگورها
۶کنترل زمینه یا context فیگورها
۷خلاصه کدهای این بخش
۹پالت‌های رنگی در seaborn
۹پالت‌های رنگی categorical
۱۰پالت‌های رنگی sequential
۱۰پالت‌های رنگی diverging
۱۲خلاصه کدهای این بخش
۱۳نمودارهای پایه در seaborn
۱۷خلاصه کدهای این بخش
۱۹ساخت نمودارهای چندگانه یا Multiplot در seaborn
۲۰خلاصه کدهای این بخش
۲۱seaborn.regplot(): Regression plot
۲۲خلاصه کدهای این بخش
۲۳خلاصه دستوره‌های Seaborn
۲۷squarify چیه؟
۲۸خلاصه کدهای این بخش
۲۹خلاصه دستوره‌های squarify
۳۰missingno چیه؟
۳۷خلاصه کدهای این بخش
۳۸خلاصه دستوره‌های missingno

seaborn, squarify, missingno Libraries

تو این جزوه سه تا از کتابخانه‌های دیگه پایتون به اسمی Seaborn، Squarify (واسه زیبایی بیشتر نمودارها) و missingno (واسه چک کردن داده‌های موجود در یه جدول) رو با هم یاد میگیریم. این کتابخانه‌ها هم مثل کتابخانه Numpy (واسه محاسبات ریاضی و کار کردن با آرایه‌ها)، pandas (واسه تبدیل آرایه‌ها به جدول و یا کار با جدولهای اکسل و CSV) و matplotlib.pyplot (واسه ترسیم نمودار) واسه علم داده یا Data Science ضروری هستن.

مطالب این جزوه خلاصه مطالب آموزشی Python - Data Visualization with Python Masterclass - Udemey بخش مربوط به Fundamentals of Data Science، سایت <https://coderzcolumn.com/tutorials/data-> A-Z <https://www.geeksforgeeks.org/treemaps-in-python-> science/missingno-visualize-missing-data-in-python <https://github.com/ResidentMario/missingno> و پلتفرم [using-squarify](https://github.com/ResidentMario/missingno) گیت‌هاب هست.

پیش نیاز این جزوه درک جزوه‌های شماره ۷ تا ۱۱ هست. اگه اونا رو کار نکردی لطفا این جزوه رو شروع نکن. اگه موقع کار با کدها تو نوت بوک ArcGIS Pro به مشکل خوردی از Jupyter Notebook حالت local رو باز نکن چون بعضی از کتابخانه‌ها رو نمی‌شناسه. ولی از Python Command Prompt که بیاریش کد رو برات اجرا میکنه. (جزوه شماره ۱۰ رو ببین).

Seaborn چیه؟

seaborn یک کتابخانه در زبان برنامه‌نویسی پایتون هست که واسه مصورسازی داده‌های آماری و ترسیم گرافیک‌های زیبا و اطلاعاتی به کار میره. ولی یه کتابخانه مجزا نیست و بر اساس کتابخانه matplotlib ساخته شده و به عنوان یک رابط سطح بالا برای ترسیم نمودارها و چارت‌های مختلف به کار میره. seaborn امکانات متنوعی برای ترسیم نمودارها از جمله نمودارهای ماتریسی، نمودارهای شبکه‌ای (Grid)، نمودارهای رگرسیونی و غیره داره. این کتابخانه با استفاده از تنظیمات پیش‌فرض زیبا و حرفه‌ای نمودار ایجاد میکنه. seaborn میتونه با ساختار داده‌های pandas انطباق پیدا کنه، از پلتفرم‌های مختلف پشتیبانی کنه و نمودارهای آماری با کد کمتر ایجاد کنه. seaborn پارامترهای matplotlib رو به دو گروه دسته‌بندی میکنه:

🔧 گروه اول پارامترهایی هستن که واسه aesthetic یا زیباسازی استفاده میشن.
🔧 گروه دوم عناصر مختلفی رو مقیاس بندی میکنن که تو contexts یا زمینه‌های مختلفی استفاده میشن مثل مصورسازی داده‌ها، ارائه یا پوستر ساختن و ...

کنترل زیبایی فیگورها

با اینکه matplotlib قابلیت تنظیم بالایی داره ولی نمودارهایی به زیبایی نمودارهای seaborn نمیتونه خروجی بده. یه چند تا مثال کار کنیم که بهتر درکش کنی.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

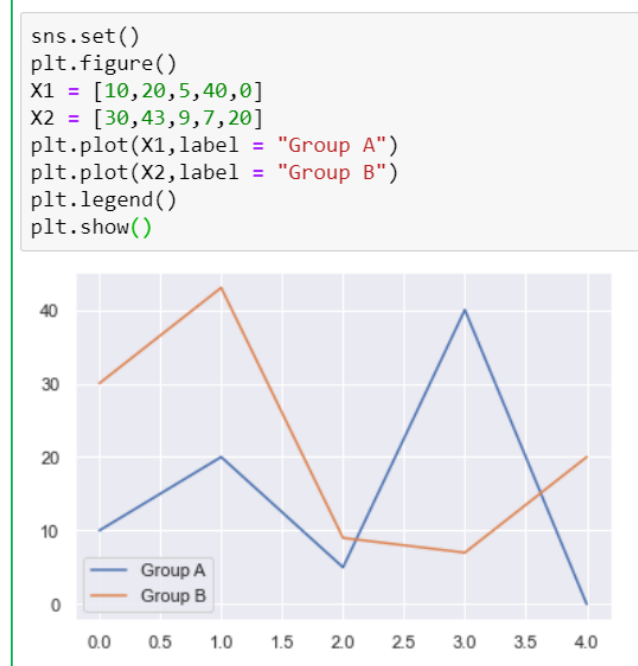
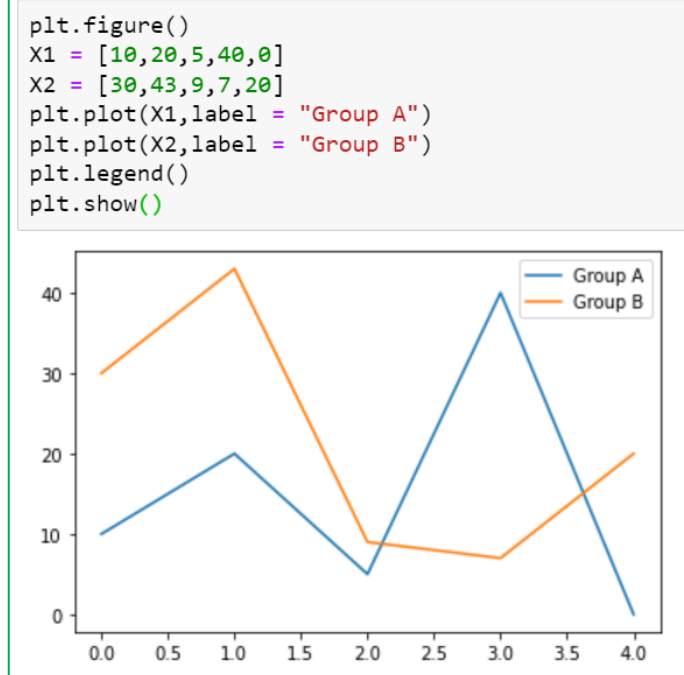
کتابخانه‌های numpy، pandas، matplotlib.pyplot، seaborn و عبارت جادویی رو وارد نوت بوک ArcGIS Pro کن. ماژول seaborn به عنوان sns وارد میشه.

تمرین اول:

میخوایم دو تا نمودار تو یه فیگور با matplotlib بسازیم.

- به فیگور با `plt.figure()` درست کن که محیط ترسیم نمودار رو برات مشخص کنه.
- دو تا متغیر به اسم `X1` و `X2` بساز و بهشون مقادیری به صورت لیست بده که عددی باشن.
- بعد با `plt.plot()` به نمودار تبدیلیشون کن. تو پرانتز به `label` هم بهشون بده.
- با `plt.legend()` براشون راهنما ترسیم کن.
- در نهایت با `plt.show()` نمایشش بده.

حالا میخوایم ببینیم **seaborn** با همین نمودار چیکار میتونه بکنه. تو خط اول همین کد بنویس `sns.set()` و اجراش کن که تغییرات رو ببینی. این دستور تنظیمات ترسیم رو به تنظیمات پیش فرض seaborn تغییر میده.



کنترل سبک یا style فیگورها

متدهای مختلفی در کتابخونه seaborn وجود داره که واسه کنترل استایل فیگورها استفاده میشه و شامل موارد زیر میشه:

- `set_style(style, rc)`: پارامترهایی رو تنظیم میکنه که سبک کلی نمودارها رو کنترل کنه.
- `axes_style(style, rc)`: پارامترهایی رو دریافت میکنه که سبک کلی نمودارها رو کنترل میکنه.
- ✓ **Style** به دیکشنری از پارامترها یا یکی از استایل‌های مقابل رو میگیره: `{darkgrid, whitegrid, dark, {white, ticks}}`
- ✓ **rc** دیکشنریهای استایل پیش فرض رو لغو میکنه. آوردنش تو پرانتز اختیاری هست.
- ✓ پارامترهای سبک یا **style parameters** ویژگی‌هایی مثل رنگ پس زمینه و فعال بودن یا نبودن شبکه یا grid به طور پیش فرض رو کنترل می‌کنن. این کار با استفاده از سیستم **matplotlib rcParams** انجام میشه.

تمرین دوم و سوم:

کد تمرین اول تو یه سلول جدید کپی کن و به جای `sns.set()` بنویس `sns.set_style()` و استایل `whitegrid` رو داخل پرانتزش توی کوتیشن بنویس.

باز هم کد بالا رو کپی کن ولی اینبار بذار `sns.set()` بمونه. خط بعد از `X2` با ¹`with` متد `sns.axes_style()` استایل `'dark'` رو بهش بده و پلاتهایی که واسه `X1` و `X2` نوشته بودی رو بذار تو دل این خط کد که رنگ و ظاهرشون تغییر کنه.

حالا اجراشون کن و با هم مقایسه‌شون کن.



اگه این تمرینها رو تو یه نوت بوک واحد انجام میدی ممکنه به دلیل محدودیتهایی که نوت بوک ArcGIS Pro داره و تو جزوه‌های قبلی بهش اشاره کردیم، بهت خطای زیر رو بده که میگه نمیتونی یه فیگور رو برای چند تا نمودار استفاده کنی.

```
-----
TypeError                                 Traceback (most recent call last)
In [3]:
Line 10: plt.show()

File E:\ArcGIS Pro\bin\Python\envs\gisenv\lib\site-packages\matplotlib\pyplot.py, in show:
Line 378: return _backend_mod.show(*args, **kwargs)

File E:\ArcGIS Pro\bin\Python\envs\gisenv\lib\site-packages\ipykernel\pylab\backend_inline.py, in show:
Line 41: display(

TypeError: 'NoneType' object is not iterable
-----
```

برای رفعش میتونی هر نمودار رو تو یه نوت بوک مجزا بنویسی. فقط یادت باشه که برای هر نوت بوک باید کتابخونه‌ها رو مجدد بیاری.

¹ With در اینجا برای اعمال یه تغییر موقت در استایل ترسیم مورد استفاده قرار میگیره. با `with` تغییرات شروع میشه و با اتمام بلاک `with` تغییرات اعمال شده به حالت پیش فرض برمیگردن. اینجا استایل محورها به `dark` تغییر میکنن و بعد از اتمام بلاک `with` به حالت پیش فرض برمیگردن.

کنترل زمینه یا context فیگورها

واسه کنترل **زمینه** یا context از تابع زیر استفاده میشه:

🔴 **seaborn.set_context(context, [font_scale], [rc])** : پارامترهایی رو تنظیم میکنه که مقیاس بندی

عناصر نمودار رو کنترل می‌کند مثل اندازه برجسبها، خطوط و

✓ **context** چهارتا پارامتر دیکشنری قبول میکنه: paper, notebook, talk, poster

✓ **font_scale** اندازه فونت عناصر رو مشخص میکنه.

✓ **rc** واسه لغو زمینه‌های پیش فرض استفاده میشه. مثل متدهای بالا آوردنش تو پرانتز اختیاری

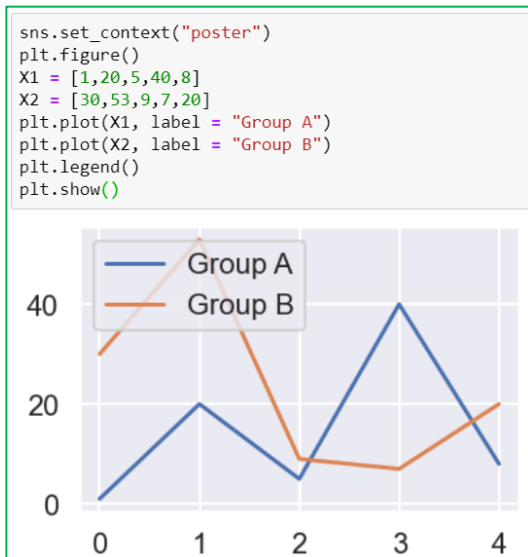
هست.

تمرین چهارم:

باز هم کد بالا رو کیی کن.

اولش بنویس **sns.set_context()** پارامتر poster رو بهش بده و

اجراش کن.



تمرین پنجم:

میخوایم به `boxplot` با کتابخانه `seaborn` بسازیم.

📁 به فایل csv. دلخواه می‌آریم و ازش خروجی می‌گیریم که جدولش رو

ببینیم. (فایل gpa_iq.scv رو تو فایل‌های تمرینی واست گذاشتیم).

🚩 ستونهای جدولش رو تبدیل میکنیم به لیست. از متد `tolist()` برای

اینکار استفاده میکنیم. ستونها رو با شماره ایندکسشون باید صدا

بزینیم. هر کدوم از لیستها رو هم خروجی میگیریم.

```
Mydata = pd.read_csv("gpa_iq.csv")
Mydata
```

	obs	gpa	iq	gender	concept
0	1	7.940	111	2	67
1	2	8.292	107	2	43
2	3	4.643	100	2	52
3	4	7.470	107	2	66
4	5	8.882	114	1	58
...
73	85	9.000	112	1	60
74	86	9.500	112	1	70
75	87	6.057	114	2	51
76	88	6.057	93	1	21
77	89	6.938	106	2	56

78 rows \times 5 columns

```
Group_A = Mydata[Mydata.columns[1]].tolist()
Group_B = Mydata[Mydata.columns[2]].tolist()
Group_C = Mydata[Mydata.columns[3]].tolist()
Group_D = Mydata[Mydata.columns[4]].tolist()
```

Group_A
[7.94, 8.292, 4.643, 7.47, 8.882, 7.585, 7.65, 2.412, 6.0, 8.833, 7.47, 5.528, 7.167, 7.571, 4.7, 8.167, 7.822, 7.598, 4.0, 6.231, 7.643, 1.76, 6.419, 9.648, 10.7, 10.58, 9.429, 8.0, 9.585, 9.571, 8.998, 8.333, 8.175, 8.0, 9.333, 9.5, 9.167, 10.14, 9.996, 10.76, 9.763, 9.41, 9.167, 9.348, 8.167, 3.647, 3.408, 3.936, 7.167, 7.647, 0.53, 6.173, 7.295, 7.295, 9.938, 7.882, 8.353, 5.062, 8.175, 8.235, 7.588, 7.647, 5.237, 7.825, 7.333, 9.167, 9.996, 8.714, 7.833, 4.885, 7.998, 3.82, 5.936, 9.0, 9.5, 6.057, 6.057, 6.938]

Group_B

Group_C
[2, 2, 2, 2, 1, 2, 2, 2, 1, 2, 1, 1, 2, 1, 1, 1, 1, 2, 1, 2, 1, 2, 2, 2, 2, 1, 1, 1, 2, 2, 1, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 1, 1, 2, 1, 2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 1, 2]

Group_D
[67, 43, 52, 66, 58, 51, 71, 51, 49, 51, 35, 54, 54, 64, 56, 69, 55, 65, 40, 66, 55, 20, 56, 68, 60, 70, 80, 53, 65, 67, 62, 39, 71, 59, 60, 64, 71, 72, 54, 64, 58, 70, 72, 70, 47, 52, 46, 66, 67, 63, 53, 67, 61, 54, 60, 60, 63, 30, 54, 66, 44, 49, 44, 67, 64, 73, 59, 37, 63, 36, 64, 42, 28, 60, 70, 51, 21, 56]

```
plt.figure(dpi = 150)
sns.set_style('whitegrid')
sns.boxplot(data = data, x = 'Groups', y = 'iq')
sns.despine(left = True, right = True, top = True)
plt.title('IQ score for different test group')
plt.show()
```

لیستها رو به دیتافریم پانداس تبدیلش میکنیم.

میخوایم یه نمودار جعبه‌ای با seaborn ازشون میسازیم.

اول باید یه figure بسازیم با dpi = 150

بعد با sns.set_style() بهش استایل 'whitegrid' بدیم.

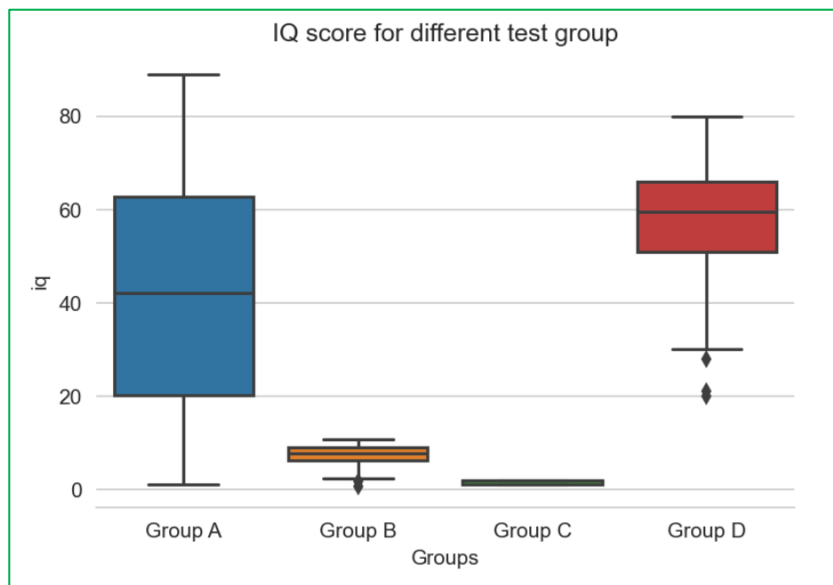
بعد با sns.boxplot() نمودار جعبه‌ای بسازیم.

حالا با sns.despine() اسپینه‌های بالا، چپ و راست نمودار رو پاک میکنیم. یعنی معادل True قرارشون میدیم.

بهش با plt.title() عنوان میدیم.

با plt.show() ازش خروجی میگیریم.

```
data = pd.DataFrame({'Groups': ['Group A'] * len(Group_A) + ['Group B'] * len(Group_B) +
                             ['Group C'] * len(Group_C) + ['Group D'] * len(Group_D),
                     'iq': Group_A + Group_B + Group_C + Group_D})
```

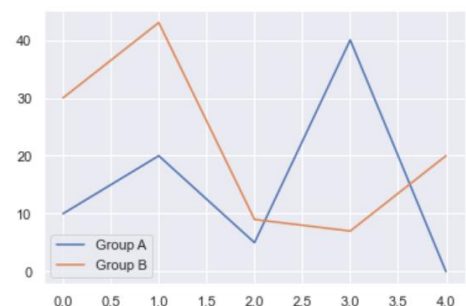


خلاصه کدهای این بخش

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

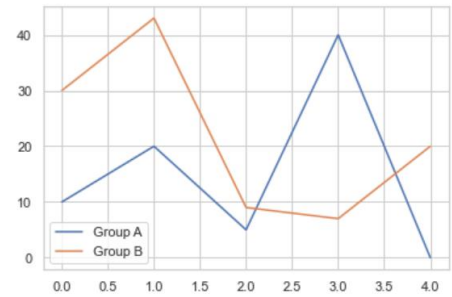
تمرین #۱

```
sns.set()
plt.figure()
X1 = [10,20,5,40,0]
X2 = [30,43,9,7,20]
plt.plot(X1,label = "Group A")
plt.plot(X2,label = "Group B")
plt.legend()
plt.show()
```



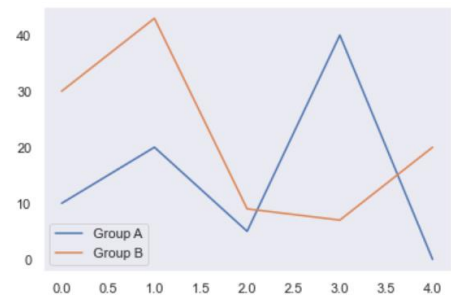
#تمرین ۲

```
sns.set_style("whitegrid")
plt.figure()
X1 = [10,20,5,40,0]
X2 = [30,43,9,7,20]
plt.plot(X1,label = "Group A")
plt.plot(X2,label = "Group B")
plt.legend()
plt.show()
```



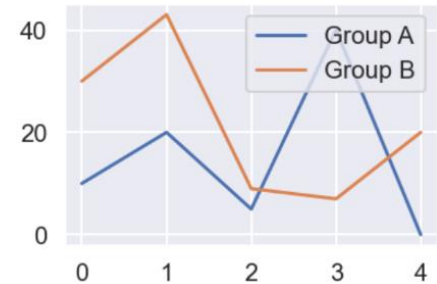
#تمرین ۳

```
sns.set()
plt.figure()
X1 = [10,20,5,40,0]
X2 = [30,43,9,7,20]
with sns.axes_style('dark'):
    plt.plot(X1,label = "Group A")
    plt.plot(X2,label = "Group B")
plt.legend()
plt.show()
```



#تمرین ۴

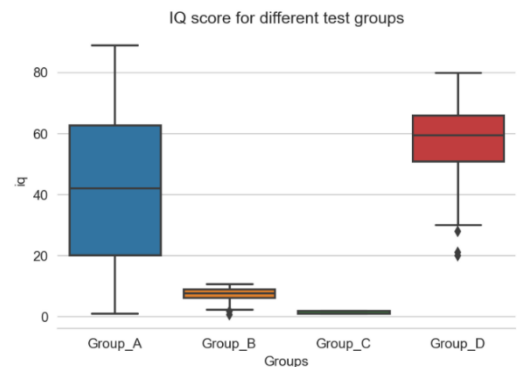
```
sns.set_context("poster")
plt.figure()
X1 = [10,20,5,40,0]
X2 = [30,43,9,7,20]
plt.plot(X1,label = "Group A")
plt.plot(X2,label = "Group B")
plt.legend()
plt.show()
```



#تمرین ۵

```
My_data = pd.read_csv('gpa_iq.csv')
Group_A = My_data[My_data.columns[0]].tolist()
Group_B = My_data[My_data.columns[1]].tolist()
Group_C = My_data[My_data.columns[3]].tolist()
Group_D = My_data[My_data.columns[4]].tolist()
data = pd.DataFrame({'Groups': ['Group_A'] * len(Group_A) +
                              ['Group_B'] * len(Group_B) +
                              ['Group_C'] * len(Group_C) +
                              ['Group_D'] * len(Group_D),
                    'iq': Group_A + Group_B + Group_C +
                          Group_D})

plt.figure(dpi=150)
sns.set_style('whitegrid')
sns.boxplot(data=data, x='Groups', y='iq')
sns.despine(left=True, right=True, top=True)
plt.title('IQ score for different test groups')
plt.show()
```



پالت‌های رنگی در seaborn

رنگ‌ها مهمترین عامل برای مصورسازی داده‌ها هستند.

از `color_palette()` واسه اختصاص دادن رنگ به عناصر استفاده میشه. در واقع واسه رنگ دادن به

عناصر با seaborn از `seaborn.color_palette([palette],[n_colors],[desat])` استفاده میشه.

✓ `palette` اسم پالت رنگی هست. پیش فرضش `none` هست یعنی هیچی انتخاب نشه. یه پارامتر اختیاری هست.

✓ `n_colors` تعداد رنگ هست. اگه تعداد رنگی که میدیم از تعداد رنگهای خودش بیشتر باشه حالت دایره عمل میکنه و برمیگرده به اولین رنگ. اینم یه پارامتر اختیاری هست.

✓ `desat` میزان ترکیبات هر رنگ رو مشخص میکنه. این هم اختیاری هست.

پالت‌های رنگی شامل `sequential`، `categorical` و `diverging` میشن.

پالت‌های رنگی `categorical`

پالت‌های رنگی `categorical` بیشتر برای داده‌هایی استفاده میشه که دستورات وراثتی ندارن. ۷ تا پیش فرض تو seaborn داره که شامل `deep`, `muted`, `pastel`, `bright`, `dark`, `colorblind` میشه.

این پالت‌ها با `sns.color_palette()` ترسیم میشن. با `sns.palplot()` هم ازشون خروجی میگیریم.

ArcGIS Pro واسه نشون دادن پالت‌های رنگی تو نوت بوکش محدودیت داره. یکبار دیگه باید اسمی که برای پالت

انتخاب کردی رو تو یه خط مجزا بنویسی که پالت رنگی دیده شه.

```
palette_deep = sns.color_palette("deep")
sns.palplot(palette_deep)
palette_deep
```



بقیه کدها رو هم مثل زیر وارد کن.

خروجی	کد	نوع پالت
	<code>palette_deep = sns.color_palette("deep")</code> <code>sns.palplot(palette_deep)</code> <code>palette_deep</code>	deep
	<code>palette_muted = sns.color_palette("muted")</code> <code>sns.palplot(palette_muted)</code> <code>palette_muted</code>	muted
	<code>palette_bright = sns.color_palette("bright")</code> <code>sns.palplot(palette_bright)</code> <code>palette_bright</code>	bright
	<code>palette_pastel = sns.color_palette("pastel")</code> <code>sns.palplot(palette_pastel)</code> <code>palette_pastel</code>	pastel
	<code>palette_dark = sns.color_palette("dark")</code> <code>sns.palplot(palette_dark)</code> <code>palette_dark</code>	dark
	<code>palette_colorblind = sns.color_palette("colorblind")</code> <code>sns.palplot(palette_colorblind)</code> <code>palette_colorblind</code>	colorblind

پالت‌های رنگی sequential

```
palette_seq1 = sns.light_palette("brown")
sns.palplot(palette_seq1)
palette_seq1
```



```
palette_seq1 = sns.light_palette("brown", reverse = True)
sns.palplot(palette_seq1)
palette_seq1
```



پالت‌های رنگی تک رنگ هستند و از تیره به روشن یا برعکس همون رنگ رو بهت خروجی میدن. این پالت‌ها رو با `sns.light_palette()` میسازیم و توی پرانتز رنگ مورد نظر رو تو کوتیشن وارد میکنیم. مثل کدهای بالا با `sns.palplot()` ازشون خروجی میگیریم. رنگها با گزینه `reverse` برعکس میشن.

کدها رو هم مثل زیر وارد کن و هر رنگی که دوست داری رو بهش بده.

خروجی	کد	نوع پالت
	<pre>palette_seq1 = sns.light_palette("brown") sns.palplot(palette_seq1) palette_seq1</pre>	brown
	<pre>palette_seq1 = sns.light_palette("brown", reverse = True) sns.palplot(palette_seq1) palette_seq1</pre>	brown / reverse
	<pre>palette_seq1 = sns.light_palette("blue") sns.palplot(palette_seq1) palette_seq1</pre>	blue
	<pre>palette_seq1 = sns.light_palette("blue", reverse = True) sns.palplot(palette_seq1) palette_seq1</pre>	blue/reverse
	<pre>palette_dark = sns.color_palette("dark") sns.palplot(palette_dark) palette_dark</pre>	purple
	<pre>palette_colorblind = sns.color_palette("colorblind") sns.palplot(palette_colorblind) palette_colorblind</pre>	Purple/ reverse

پالت‌های رنگی diverging

متداولترین پالت‌های رنگی diverging شامل موارد زیر هستن:

➡ **پالت رنگ‌های گرم و سرد یا coolwarm:** این نوع پالت شامل ترکیب رنگ‌های گرم مثل نارنجی یا قرمز و

رنگ‌های سرد مثل آبی یا سبز هست. این ترکیب‌ها تبدیل‌های جذابی از نظر رنگی ایجاد می‌کنن.

➡ **پالت رنگ‌های روشن و تاریک:** در این نوع پالت، یک رنگ روشن با یک رنگ تاریک ترکیب می‌شه. این

تضاد نوری تبدیل‌های شدیدی از نظر رنگی به وجود می‌آرن.

➡ **پالت رنگ‌های متقابل:** در این نوع پالت، دو رنگ متقابل از دو قسمت مختلف پالت برای تاکید بر تقابل

رنگی استفاده میشن.



➡ **پالت رنگ‌های تاریک و روشن در کنار یک رنگ وسطی:** در این نوع، دو رنگ تاریک و روشن در کنار یک

رنگ وسطی ترکیب می‌شن تا تبدیل‌های متعادلی از نظر رنگی ایجاد کنن.

این کدها با `sns.color_palette()` ساخته میشه و با `sns.palplot()` ازشون خروجی گرفته میشه. با

`sns.diverging_palette()` و دادن عدد بهش هم ساخته میشه.

کدها رو هم مثل زیر وارد کن که نتیجه رو ببینی.

خروجی	کد	نوع پالت
	palette_div = sns.color_palette("coolwarm",7) sns.palplot(palette_div) palette_div	coolwarm
	palette_div_custom = sns.diverging_palette(440,40,n=7) sns.palplot(palette_div_custom) palette_div_custom	با شماره

برای داشتن ترکیبات رنگی بیشتر این سایت رو ببین:

https://seaborn.pydata.org/tutorial/color_palettes.html

تمرین اول:

میخوایم با heat map اطلاعات مسافران پرواز از سال ۲۰۰۱ تا ۲۰۱۲ رو نمایش بدیم ولی با رنگهای خودمون.

✚ با تعریف یه متغیر یه فایل CSV به اسم flights رو اضافه میکنیم.

✚ فایل سه تا ستون داره با pivot(). ستونها رو بهش اختصاص میدیم.

✚ sns.set() مینویسیم.

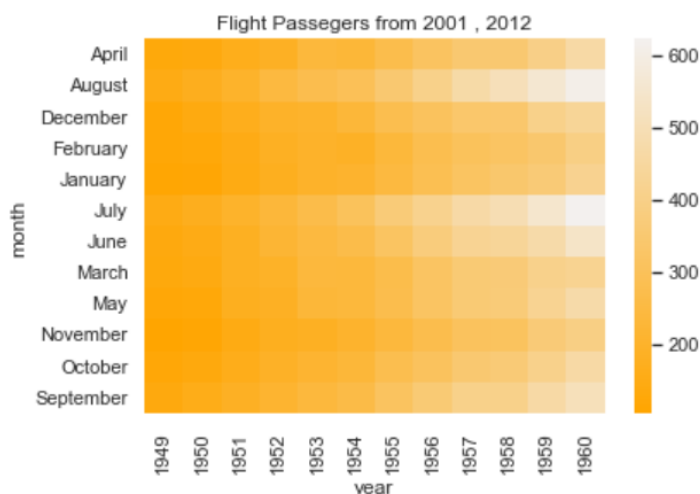
✚ یه فیگور با dpi = 150 میسازیم.

✚ Heatmap() درست میکنیم. cmap رو مساوی با sns.light_palette() که رنگ نارنجی رو به حالت

sequential و به شکل برعکس یا reverse بهمون بده.

✚ عنوان هم بهش میدیم و ارزش خروجی میگیریم.

```
Data = pd.read_csv('flights.csv')
Data_Frame = Data.pivot_table(index='month', columns='year', values='passengers')
sns.set()
plt.figure(dpi=150)
sns.heatmap(Data_Frame, cmap = sns.light_palette('orange', as_cmap = True, reverse = True))
plt.title('Flight Passengers from 2001 , 2012')
plt.show()
```

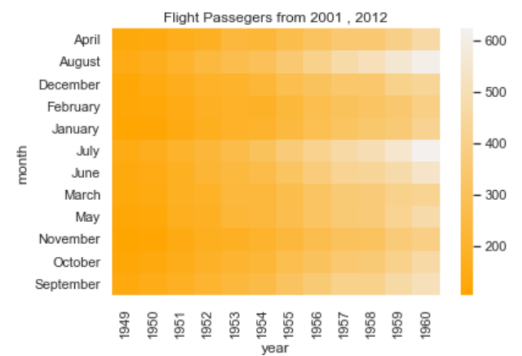


نمودار نشون میده که تعداد مسافران تو July 1960 و August 1960 بالاترین بوده.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

تمرین ۱

```
Data = pd.read_csv('flights.csv')
Data_Frame = Data.pivot_table(index='month', columns='year',
                                values='passengers')
sns.set()
plt.figure(dpi=150)
sns.heatmap(Data_Frame, cmap = sns.light_palette('orange',
as_cmap = True, reverse = True))
plt.title('Flight Passengers from 2001 , 2012')
plt.show()
```



نمودارهای پایه در seaborn

تو جزوه قبلی نمودارهای پایه رو تو matplotlib با هم ترسیم کردیم. میخوایم این نمودارها رو تو seaborn بکشیم.

برای اضافه کردن فایل‌های اکسل و CSV میتونی مستقیم از دستور **sns.load_data()** استفاده کنی و تو پرانتز اسم فایل رو داخل کوتیشن بیاری.

تمرین اول: **sns.barplot()**

sns.barplot() یک تابع در کتابخانه seaborn در پایتون هست که برای رسم نمودارهای میله‌ای (Bar Plot) استفاده می‌شه. باید بهش دو تا ستون x و y معرفی شه. اندازه هر ستونش میانگین اعداد همون ستون در جدول هستن.

Syntax:

sns.barplot(x=None, y=None, hue=None, data=None, order=None, hue_order=None, estimator=<function mean at 0x000002BC3EB5C4C8>, ci=95, n_boot=1000, units=None, orient=None, color=None, palette=None, saturation=0.75, errcolor='r', errwidth=None, capsize=None, dodge=True, ax=None, **kwargs,)

✚ فایل salaries با فرمت csv رو می‌اریم. این فایل ستونهای متعددی داره.

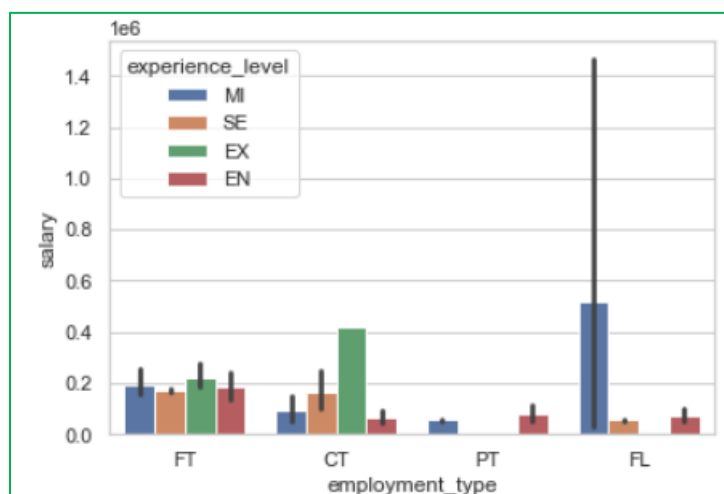
✚ با **sns.set()** بهش استایل whitetgrid میدیم.

✚ با **barplot()** ازش پلات میسازیم. یکی از ستونها رو به عنوان محور x و یه ستون دیگه به عنوان محور y

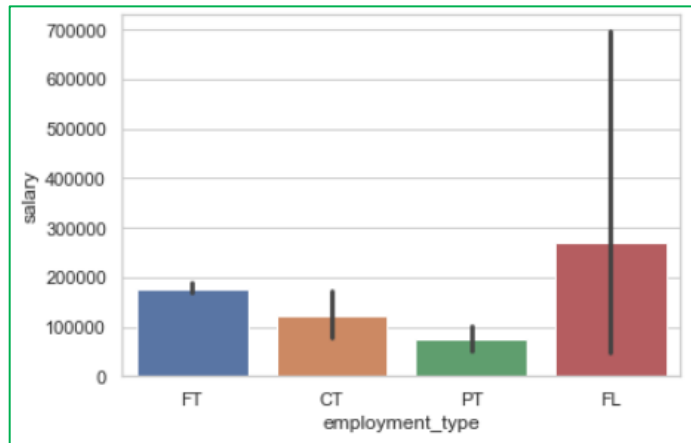
معرفی میکنیم. یه ستون هم برای **hue** باید بهش بدیم که بگیم اطلاعات رو تو چند دسته بهمون نشون بده.

✚ ازش خروجی میگیریم.

```
Data = pd.read_csv('salaries.csv')
sns.set(style='whitegrid')
sns.barplot(x='employment_type', y='salary', hue='experience_level', data=Data)
plt.show()
```



بخش hue رو بردار که ببینی نتیجه چی میشه.



تمرین دوم: `sns.kdeplot()`

میخوایم به `sns.kdeplot()` بسازیم. KDE مخفف Kernel Density Estimation یا تخمین چگالی احتمال بر اساس داده‌های مشاهده شده هست. نشون میده که چقدر احتمال وقوع مقادیر مختلف در داده‌ها وجود داره. یه نمودار اسموتینگ یا Smoothed histogram بهمون خروجی میده که همه مقادیر داده رو پوشش میده.

Syntax:

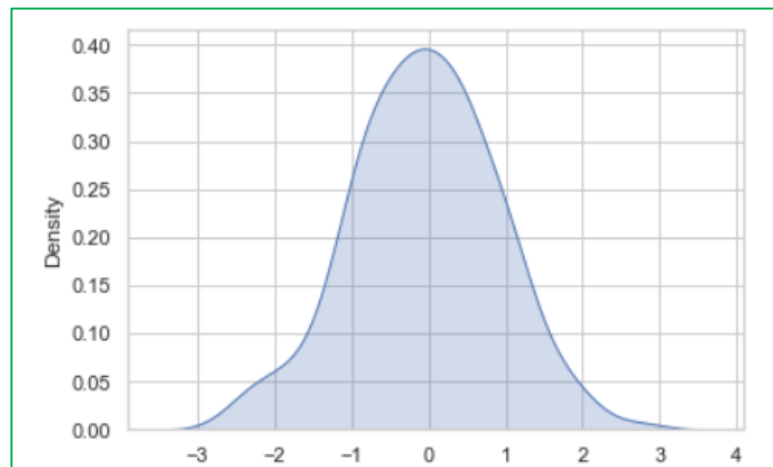
`seaborn.kdeplot(data=None, *, x=None, y=None, hue=None, weights=None, palette=None, hue_order=None, hue_norm=None, color=None, fill=None, multiple='layer', common_norm=True, common_grid=False, cumulative=False, bw_method='scott', bw_adjust=1, warn_singular=True, log_scale=None, levels=10, thresh=0.05, gridsize=200, cut=3, clip=None, legend=True, cbar=False, cbar_ax=None, cbar_kws=None, ax=None, **kwargs)`

با نامپی به آرایه رندوم درست میکنیم.

```
X = np.random.randn(200)
sns.kdeplot(X, shade=True)
plt.show()
```

با `sns.kdeplot()` ازش پلات میگیریم. واسه سایه دار کردن زیر

منحنی تو پرانتزش مینویسیم `shade = True`.



تمرین سوم: `sns.jointplot()`

میخوایم به `sns.jointplot()` بکشیم. این نمودار بهمون کمک میکنه که رابطه بین دو تا متغیر رو بررسی کنیم و توزیع هر متغیر رو جداگونه نمایش بدیم. سینتکسش به شکل زیر هست:

Syntax:

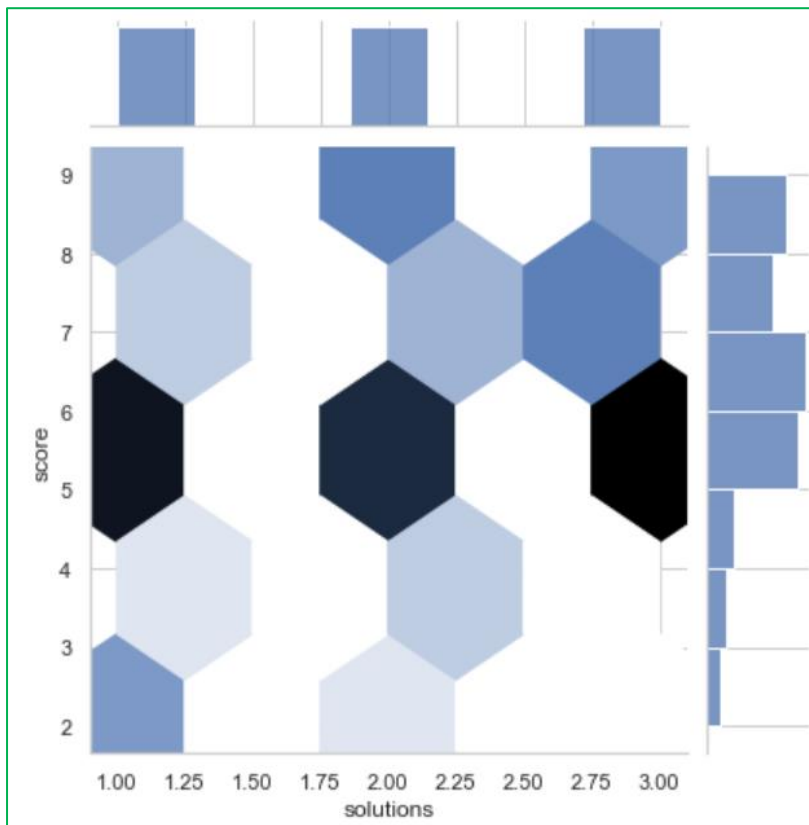
`sns.jointplot(x, y, data=None, kind='scatter', stat_func=None, color=None, height=6, ratio=5, space=0.2, dropna=True, xlim=None, ylim=None, joint_kws=None, marginal_kws=None, annot_kws=None, **kwargs)`

➤ به فایل csv به نام 'attention' رو با دستور sns.load_dataset() اضافه میکنیم.

➤ با sns.jointplot() به نمودار میسازیم. برای x و y به ستون از جدول تعریف میکنیم. برای kind کلمه hex رو تایپ میکنیم که نوع نمودار رو بهمون نشون بده. بخش data رو هم مساوی اطلاعات جدول قرار میدیم.

```
data = sns.load_dataset('attention')
sns.jointplot(x = "solutions", y = "score",
              kind = "hex", data = data)
plt.show()
```

➤ با plt.show() ازش خروجی میگیریم.



تو قسمت kind که نوع نمودار رو مشخص میکنه میتونی از kde, reg, hist و ... هم استفاده کنی که نمودارهای متعدد داشته باشی.

تمرین چهارم: sns.pairplot()

میخوایم با **pairplot()** نمودارهای جفتی بین متغیرهای مختلف به مجموعه داده بسازیم. این نمودار به صورت پیش فرض به شبکه از محورها رو ایجاد میکنه طوری که هر متغیر عددی در داده‌ها با متغیرهای دیگه در محور y به اشتراک گذاشته میشه. ازش واسه دیدن تعاملات بین متغیرهای مختلف در به مجموعه داده استفاده میشه.

Syntax:

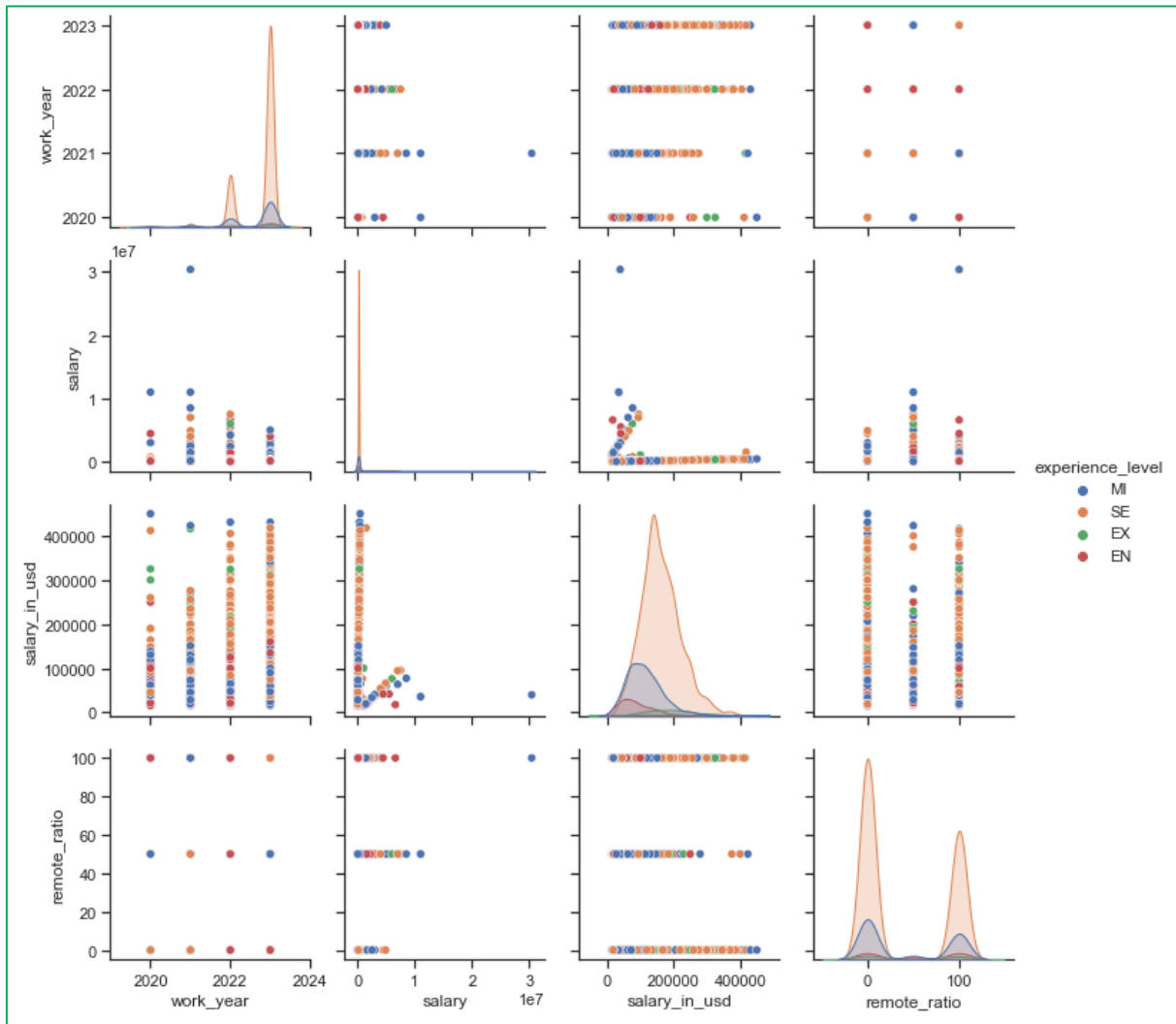
```
seaborn.pairplot(data, *, hue=None, hue_order=None, palette=None, vars=None, x_vars=None,
y_vars=None, kind='scatter', diag_kind='auto', markers=None, height=2.5, aspect=1, corner=False,
dropna=False, plot_kws=None, diag_kws=None, grid_kws=None, size=None)
```

➤ به فایل به اسم salaries.csv اضافه میکنیم. ترجیحا فایلی رو وارد کن که تعداد ستونهای زیادی داشته باشه.

➤ با sns.set() بهش استایل بده.

```
Data = pd.read_csv('salaries.csv')
sns.set(style='ticks', color_codes = True)
g = sns.pairplot(Data, hue='experience_level')
plt.show()
```

با pairplot() از ستونهای نمودارهای جفتی بساز که گروه‌ها با رنگهای مختلفی دیده بشن.



تمرین پنجم: sns.violinplot()

میخوایم برای فایل salaries.csv به **violinplot()** بسازیم. نمودار ویولن یک ابزار مفید در تجسم توزیع داده‌های عددی و احتمالی هست. این نمودار با ترکیب ویژگی‌های نمودار جعبه‌ای (box plot) و نمودار چگالی هسته (kernel density plot)، اطلاعات زیادی از توزیع داده رو نمایش می‌ده. برای نمایش توزیع داده‌ها و تجزیه و تحلیل آماری استفاده میشه. عرض نمودار ویولنی نشون دهنده توزیع داده‌های احتمالی هست که همیشه توزیع داده‌های مختلف رو هم با هم مقایسه کرد. در قسمت میانی نمودار همیشه نمایش پیکها و توزیع چگالی داده‌ها رو دید.

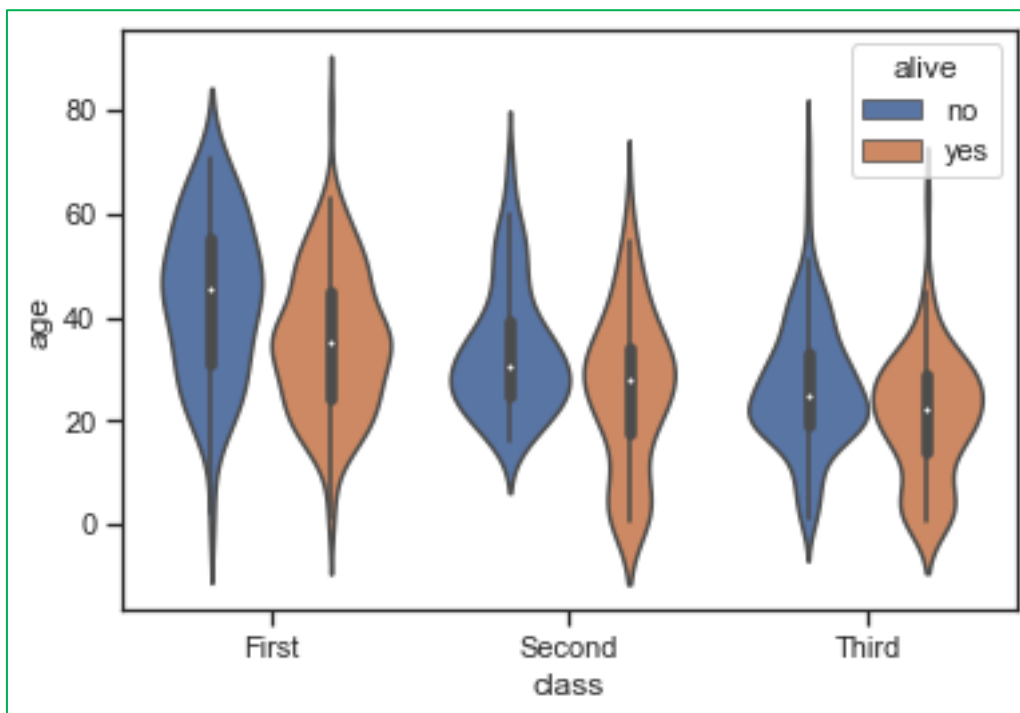
Syntax:

```
seaborn.violinplot(data=None, *, x=None, y=None, hue=None, order=None, hue_order=None, orient=None, color=None, palette=None, saturation=0.75, fill=True, inner='box', split=False, width=0.8, dodge='auto', gap=0, linewidth=None, linecolor='auto', cut=2, gridsize=100, bw_method='scott', bw_adjust=1, density_norm='area', common_norm=False, hue_norm=None, formatter=None, log_scale=None, native_scale=False, legend='auto', scale=<deprecated>, scale_hue=<deprecated>, bw=<deprecated>, inner_kws=None, ax=None, **kwargs)
```

فایل tatanic.csv رو با دستور sns.load_dataset() بیار.

ازش violinplot() با sns.violinplot() بساز. Data رو مساوی با اسمی که برای جدول گذاشتی قرار بده و دو تا محور x و y رو با ستونای جدول براش مشخص کن. اگه براش hue تعریف کنی اطلاعات رو تو چند تا دسته بهت میده. ازش خروجی بگیر.

```
df = sns.load_dataset("titanic")
sns.violinplot(data=df, x="class", y="age", hue="alive")
plt.show()
```

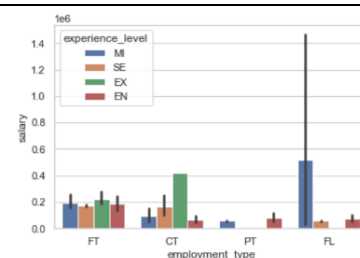


خلاصه کدهای این بخش

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

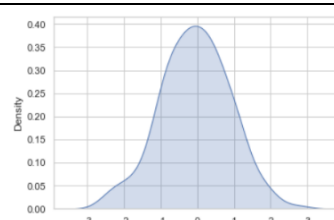
تمرین ۱

```
Data = pd.read_csv('salaries.csv')
sns.set(style='whitegrid')
sns.barplot(x='employment_type', y='salary',
            hue='experience_level', data=Data)
plt.show()
```



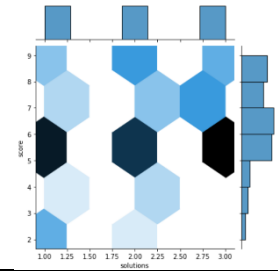
تمرین ۲

```
X = np.random.randn(200)
sns.kdeplot(X, shade=True)
plt.show()
```



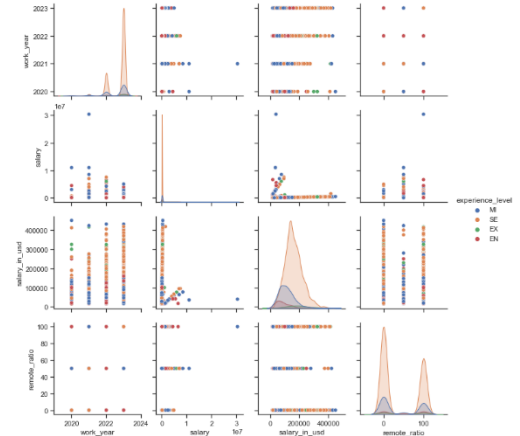
تمرین ۳

```
data = pd.read_csv('attention.csv')
sns.jointplot(x = "solutions", y = "score",
              kind = "hex", data = data)
plt.show()
```



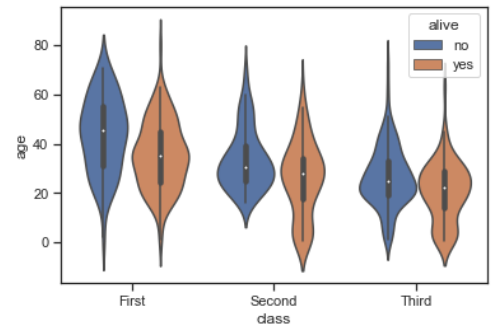
تمرین ۴

```
Data = pd.read_csv('salaries.csv')
sns.set(style='ticks', color_codes = True)
g = sns.pairplot(Data, hue='experience_level')
plt.show()
```



تمرین ۵

```
df = sns.load_dataset("titanic")
sns.violinplot(data=df, x="class", y="age", hue="alive")
plt.show()
```



ساخت نمودارهای چندگانه یا Multiplot در seaborn

میخوایم مالتی پلاتهای انعطاف پذیر بسازیم. FacetGrid واسه مصورسازی متغیرهای چندگانه به صورت مجزا عالیه. برای ترسیمشون از **sns.FacetGrid()** استفاده میشه.

Syntax:

```
seaborn.FacetGrid(data, *, row=None, col=None, hue=None, col_wrap=None, sharex=True, sharey=True, height=3, aspect=1, palette=None, row_order=None, col_order=None, hue_order=None, hue_kws=None, dropna=False, legend_out=True, despine=True, margin_titles=False, xlim=None, ylim=None, subplot_kws=None, gridspec_kws=None)
```

```
FacitGrid.map(func, *args, **kwaegs)
```

❖ از دستور **FacetGrid.map(func, *args, **kwargs)** هم میشه واسه ترسیم نمودارهای متعدد در یه شبکه تعریف شده استفاده کرد. هر تابع یا func رو واسه هر زیرمجموعه داده در هر شبکه فراخوانی میکنه. اگه در حالت hue باشه باید یه آرگومان label هم قبول کنه. بعد داده‌های هر متغیر به ترتیب مشخص شده در فراخوانی به func منتقل میشن.

❖ از **seaborn.FacetGrid.map_dataframe** که به صورت **FacetGrid.map_dataframe(func, *args, **kwargs)** نوشته میشه واسه تطبیق دیتافریمها با نمودارها استفاده میشه. این متد واسه ترسیم مالتی پلاتها بر اساس داده‌های موجود در یه دیتافریم به صورت همزمان بکار میره. با کمک **map_DataFrame** میتونی یه تابع سفارشی رو تعریف کنی و این تابع رو روی داده‌های DataFrame اعمال کنی و نمودارهایی با توجه به این تابع ایجاد کنی.

تمرین ۱:

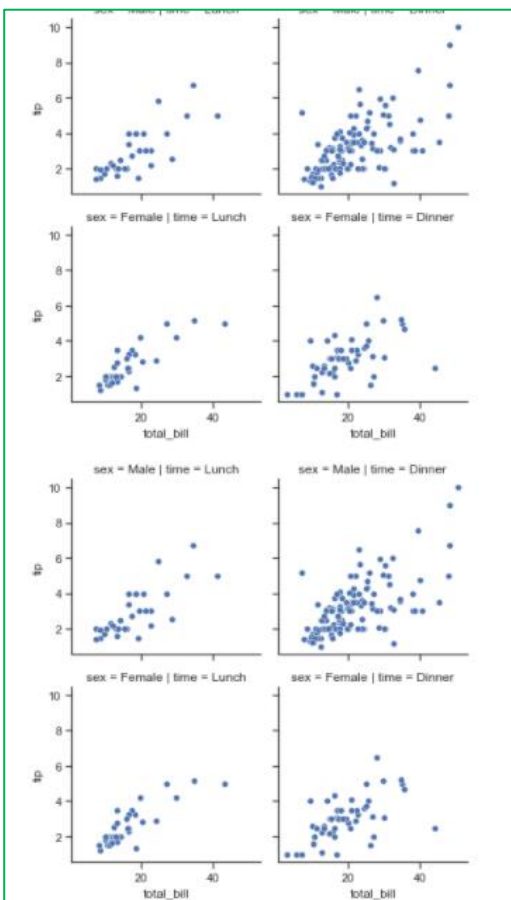
➤ یه فایل CSV به اسم tips رو با دستور **sns.load_dataset()** بارش رو بذار **tips**.

➤ با **sns.FacetGrid()** ازش مالتی پلات بساز. برای سطر و ستونش هم دو اسامی دو تا از ستونهای فایل که آوردی رو بده. اسمش رو بذار **g**. اینجا برای ستون از **time** و برای سطرها از **sex** استفاده کردیم.

➤ حالا با **g.map()** پلاتهای scatter بساز و اینبار از سر ستونهای **total_bill** و **tip** استفاده کن.

➤ با **plt.show()** نمایشش بده.

```
tips = sns.load_dataset("tips")
g = sns.FacetGrid(tips, col="time", row="sex")
g.map(sns.scatterplot, "total_bill", "tip")
plt.show()
```



تمرین ۲:

دو خط اول کد مشابه کد بالا هست. فقط برای خط سوم میخوایم از `map_DataFrame` استفاده کنیم.

اسم `FaceGrid` رو که `g` گذاشته بودی اول `map_DataFrame()` بیار و با نقطه جداشون کن. تو پرانتزش بنویس که میخوای نمودار هیستوگرام داشته باشی یعنی `sns.histplot` و فیلد `total_bill` رو هم برای ساخت ستونهای نمودار بهش بده. با `plt.show()` نمایشش بده.

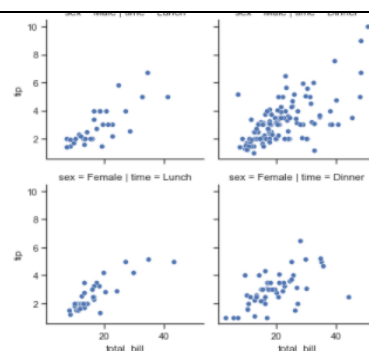
```
g = sns.FacetGrid(tips, col="time", row="sex")
g.map_dataframe(sns.histplot, x="total_bill")
plt.show()
```

خلاصه کدهای این بخش

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

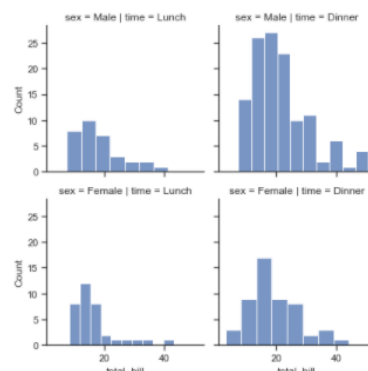
تمرین ۱

```
tips = sns.load_dataset("tips")
g = sns.FacetGrid(tips, col="time", row="sex")
g.map(sns.scatterplot, "total_bill", "tip")
plt.show()
```



تمرین ۲

```
tips = sns.load_dataset("tips")
g = sns.FacetGrid(tips, col="time", row="sex")
g.map_dataframe(sns.histplot, x="total_bill")
plt.show()
```



seaborn.regplot(): Regression plot

نمودار رگرسیون یا Regression Plot، واسه نشون دادن رابطه بین دو تا متغیر استفاده میشه. در این نمودار، محور x به یک متغیر مستقل و محور y به یک متغیر وابسته اختصاص داده میشه. این نمودار به صورت خطی یا غیرخطی رسم میشه و خط رسم شده، بهترین تقریب برای رابطه بین دو تا متغیر هست. از این نمودار در آمار و همچنین در یادگیری ماشین واسه پیدا کردن رابطه بین دو تا متغیر استفاده میشه.

Syntax:

```
seaborn.regplot(data=None, *, x=None, y=None, x_estimator=None, x_bins=None, x_ci='ci', scatter=True, fit_reg=True, ci=95, n_boot=1000, units=None, seed=None, order=1, logistic=False, lowess=False, robust=False, logx=False, x_partial=None, y_partial=None, truncate=True, dropna=True, x_jitter=None, y_jitter=None, label=None, color=None, marker='o', scatter_kws=None, line_kws=None, ax=None)
```

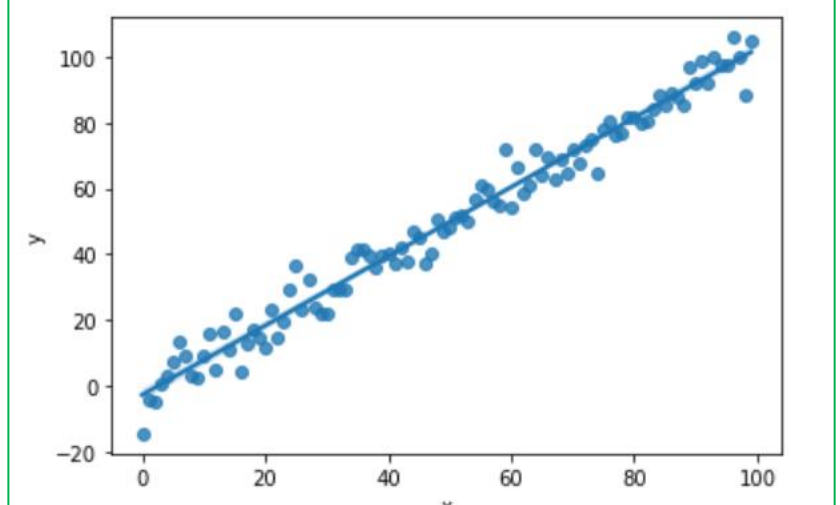
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

تمرین اول: کتابخانه‌های مورد نیاز رو اضافه کن.

```
x = np.arange(100)
y = x + np.random.normal(0, 5, size=100)

# Create a DataFrame to hold the data
data = pd.DataFrame({'x': x, 'y': y})

# Plot using seaborn
sns.regplot(data=data, x='x', y='y')
plt.show()
```



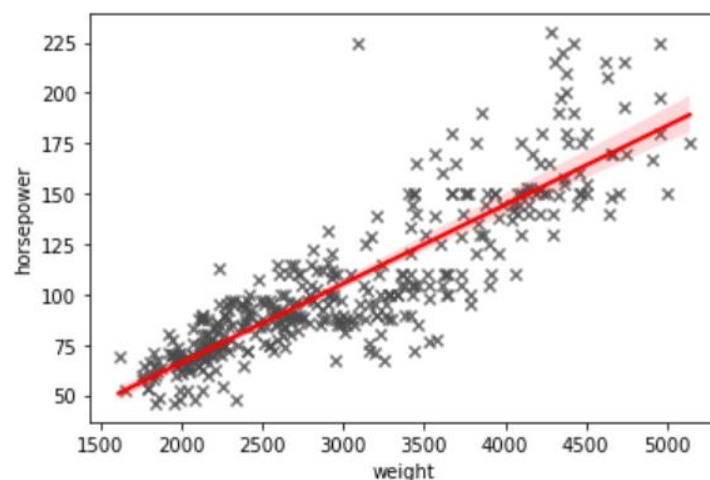
تمرین دوم:

میخوایم یه جدول اطلاعاتی بیاریم و یکی از ستونهاش رو به عنوان متغیر مستقل یا x معرفی کنیم و یه ستون دیگه رو به عنوان متغیر وابسته یا y.

فایل mpg رو با دستور sns.load_dataset() بیار. اسمش رو بذار mpg.

با sns.regplot() ازش یه نمودار رگرسیونی درست کن که data معادل mpg باشه و x که متغیر مستقل هست ستون weight رو بگیره و y که متغیر وابسته هست ستون horsepower رو. Ci اندازه فاصله اطمینان هست. نوع marker رو هر چی دوست داری بده. "3" color میزان شفافیت مارکرها رو نشون میده. واسه خود خط هم میتونی نوع و رنگ مشخص کنی.

```
mpg = sns.load_dataset('mpg')
sns.regplot(data=mpg, x="weight", y="horsepower",
            ci=99, marker="x", color=".3",
            line_kws=dict(color="r"))
plt.show()
```

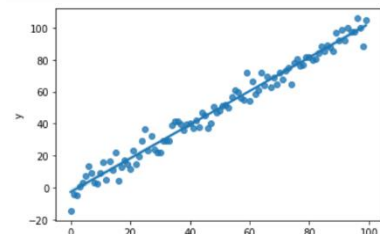


خلاصه کدهای این بخش

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

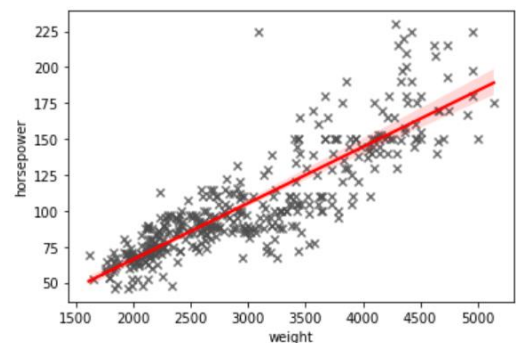
تمرین ۱

```
x = np.arange(100)
y = x + np.random.normal(0, 5, size=100)
data = pd.DataFrame({'x': x, 'y': y})
sns.regplot(data=data, x='x', y='y')
plt.show()
```



تمرین ۲

```
mpg = sns.load_dataset('mpg')
sns.regplot(data=mpg, x="weight", y="horsepower",
            ci=99, marker="x", color=".3",
            line_kws=dict(color="r"))
plt.show()
```



خلاصه دستوره‌های Seaborn

توضیحات	توابع و متدها
تنظیمات ترسیم رو به تنظیمات پیش فرض seaborn تغییر میده.	<code>sns.set()</code>
پارامترهایی رو تنظیم میکنه که سبک کلی نمودارها رو کنترل کنه. پارامترهایی رو دریافت میکنه که سبک کلی نمودارها رو کنترل میکنه. ✓ style یه دیکشنری از پارامترها یا یکی از استایل‌های مقابل رو میگیره: <code>{darkgrid, whitegrid, dark, white, ticks}</code> ✓ rc دیکشنریهای استایل پیش فرض رو لغو میکنه. آوردنش تو پرانتز اختیاری هست. پارامترهای سبک یا <code>style parameters</code> ویژگی‌هایی مثل رنگ پس زمینه و فعال بودن یا نبودن شبکه یا <code>grid</code> به طور پیش فرض رو کنترل می‌کنن. این کار با استفاده از سیستم <code>matplotlib rcParams</code> انجام میشه.	<code>set_style(style, [rc])</code> <code>axes_style(style, [rc])</code>
پارامترهایی رو تنظیم میکنه که مقیاس‌بندی عناصر نمودار رو کنترل میکنه مثل اندازه برجسبها، خطوط و ✓ context چهارتا پارامتر دیکشنری قبول میکنه: <code>paper, notebook, talk, poster</code> ✓ font_scale اندازه فونت عناصر رو مشخص میکنه. rc واسه لغو زمینه‌های پیش فرض استفاده میشه. مثل متدهای بالا آوردنش تو پرانتز اختیاری هست.	<code>Seaborn.set_context(context, [font_scale]. [rc])</code>
اطلاعات ستونهای جدول رو به لیست تبدیل میکنه.	<code>.tolist()</code>
نمودار جعبه ای	<code>sns.boxplot()</code>
برای پاک کردن اسپینه‌های اطراف نمودار	<code>sns.despine(True)</code>
واسه اختصاص دادن رنگ به عناصر ✓ palette اسم پالت رنگی هست. پیش فرضش <code>none</code> هست یعنی هیچی انتخاب نشه. یه پارامتر اختیاری هست. ✓ n_colors تعداد رنگ هست. اگه تعداد رنگی که میدیم از تعداد رنگهای خودش بیشتر باشه حالت دایره عمل میکنه و برمیگرده به اولین رنگ. اینم یه پارامتر اختیاری هست. desat میزان ترکیبات هر رنگ رو مشخص میکنه. این هم اختیاری هست.	<code>seaborn.color_palette([palette],[n_colors],[desat])</code>

<code>sns.palplot()</code>	واسه خروجی گرفتن از پلتهای رنگی
<code>sns.light_palette()</code>	پالتهای رنگی تک رنگ هستن و از تیره به روشن یا برعکس همون رنگ رو بهت خروجی میدن.
<code>sns.color_palette()</code> <code>sns.diverging_palette()</code>	<p>با دادن عدد یا اسم پالت ساخته میشه.</p> <p>متداولترین پالتهای رنگی <code>diverging</code> شامل موارد زیر هستن:</p> <p>🚩 پالت رنگهای گرم و سرد یا coolwarm: این نوع پالت شامل ترکیب رنگهای گرم مثل نارنجی یا قرمز و رنگهای سرد مثل آبی یا سبز هست. این ترکیبها تبدیل‌های جذابی از نظر رنگی ایجاد می‌کنن.</p> <p>🚩 پالت رنگهای روشن و تاریک: در این نوع پالت، یک رنگ روشن با یک رنگ تاریک ترکیب می‌شه. این تضاد نوری تبدیل‌های شدیدی از نظر رنگی به وجود می‌آرن.</p> <p>🚩 پالت رنگهای متقابل: در این نوع پالت، دو رنگ متقابل از دو قسمت مختلف پالت برای تاکید بر تقابل رنگی استفاده میشن.</p> <p>پالت رنگهای تاریک و روشن در کنار یک رنگ وسطی: در این نوع، دو رنگ تاریک و روشن در کنار یک رنگ وسطی ترکیب می‌شن تا تبدیل‌های متعادلی از نظر رنگی ایجاد کنن.</p>
<code>sns.load_data()</code>	برای اضافه کردن فایل‌های اکسل و CSV
<code>sns.barplot(x=None, y=None, hue=None, data=None, order=None, hue_order=None, estimator=<function mean at 0x000002BC3EB5C4C8>, ci=95, n_boot=1000, units=None, orient=None, color=None, palette=None, saturation=0.75, errcolor='.', errwidth=None, capsize=None, dodge=True, ax=None, **kwargs)</code>	یک تابع در کتابخانه Seaborn در پایتون هست که برای رسم نمودارهای میله‌ای (Bar Plot) استفاده می‌شه. باید بهش دو تا ستون <code>x</code> و <code>y</code> معرفی شه. اندازه هر ستونش میانگین اعداد همون ستون در جدول هستن.
<code>seaborn.kdeplot(data=None, *, x=None, y=None, hue=None, weights=None, palette=None, hue_order=None, hue_norm=None, color=None, fill=None, multiple='layer', common_norm=True,</code>	KDE مخفف Kernel Density Estimation یا تخمین چگالی احتمال بر اساس داده‌های مشاهده شده هست. نشون میده که چقدر احتمال وقوع مقادیر مختلف در داده‌ها وجود داره. یه نمودار اسموتینگ یا Smoothed

<code>common_grid=False, cumulative=False, bw_method='scott', bw_adjust=1, warn_singular=True, log_scale=None, levels=10, thresh=0.05, gridsize=200, cut=3, clip=None, legend=True, cbar=False, cbar_ax=None, cbar_kws=None, ax=None, **kwargs)</code>	histogram بهمون خروجی میدهد که همه مقادیر داده رو پوشش میدهد.
<code>sns.jointplot(x, y, data=None, kind='scatter', stat_func=None, color=None, height=6, ratio=5, space=0.2, dropna=True, xlim=None, ylim=None, joint_kws=None, marginal_kws=None, annot_kws=None, **kwargs)</code>	این نمودار بهمون کمک میکند که رابطه بین دو تا متغیر رو بررسی کنیم و توزیع هر متغیر رو جداگونه نمایش بدیم. تو قسمت kind که نوع نمودار رو مشخص میکنه میتونی از kde, reg, hist و ... هم استفاده کنی که نمودارهای متعدد داشته باشی.
<code>seaborn.pairplot(data, *, hue=None, hue_order=None, palette=None, vars=None, x_vars=None, y_vars=None, kind='scatter', diag_kind='auto', markers=None, height=2.5, aspect=1, corner=False, dropna=False, plot_kws=None, diag_kws=None, grid_kws=None, size=None)</code>	واسه ترسیم نمودارهای جفتی بین متغیرهای مختلف استفاده میشه. این نمودار به صورت پیش فرض یه شبکه از محورها رو ایجاد میکنه طوری که هر متغیر عددی در داده‌ها با متغیرهای دیگه در محور y به اشتراک گذاشته میشه. ازش واسه دیدن تعاملات بین متغیرهای مختلف در یه مجموعه داده استفاده میشه.
<code>seaborn.violinplot(data=None, *, x=None, y=None, hue=None, order=None, hue_order=None, orient=None, color=None, palette=None, saturation=0.75, fill=True, inner='box', split=False, width=0.8, dodge='auto', gap=0, linewidth=None, linecolor='auto', cut=2, gridsize=100, bw_method='scott', bw_adjust=1, density_norm='area', common_norm=False, hue_norm=None, formatter=None, log_scale=None, native_scale=False, legend='auto', scale='<deprecated>', scale_hue='<deprecated>', bw='<deprecated>', inner_kws=None, ax=None, **kwargs)</code>	. نمودار ویولن یک ابزار مفید در تجسم توزیع داده‌های عددی و احتمالی هست. این نمودار با ترکیب ویژگی‌های نمودار جعبه‌ای (box plot) و نمودار چگالی هسته (kernel density plot)، اطلاعات زیادی از توزیع داده رو نمایش میده. برای نمایش توزیع داده‌ها و تجزیه و تحلیل آماری استفاده میشه. عرض نمودار ویولنی نشون دهنده توزیع داده‌های احتمالی هست که میشه توزیع داده‌های مختلف رو هم با هم مقایسه کرد. در قسمت میانی نمودار میشه نمایش بیکها و توزیع چگالی داده‌ها رو دید.
<code>seaborn.FacetGrid(data, *, row=None, col=None, hue=None, col_wrap=None,</code>	. FacetGrid واسه مصورسازی متغیرهای چندگانه به صورت مجزا عالیه.

<pre>sharex=True, sharey=True, height=3, aspect=1, palette=None, row_order=None, col_order=None, hue_order=None, hue_kws=None, dropna=False, legend_out=True, despine=True, margin_titles=False, xlim=None, ylim=None, subplot_kws=None, gridspec_kws=None) FacitGrid.map(func, *args, **kwaegs)</pre>	<p>❖ از دستور FacetGrid.map(func, *args, **kwargs) هم همیشه واسه ترسیم نمودارهای متعدد در یه شبکه تعریف شده استفاده کرد. هر تابع یا func رو واسه هر زیرمجموعه داده در هر شبکه فراخوانی میکنه. اگه در حالت hue باشه باید یه آرگومان label هم قبول کنه. بعد داده‌های هر متغیر به ترتیب مشخص شده در فراخوانی به func منتقل میشن.</p> <p>❖ از seaborn.FacetGrid.map_dataframe که به صورت FacetGrid.map_dataframe(func, *args, **kwargs) نوشته میشه واسه تطبیق دیتافریمها با نمودارها استفاده میشه. این متد واسه ترسیم مالتی پلاتها بر اساس داده‌های موجود در یه دیتافریم به صورت همزمان بکار میره. با کمک map_DataFrame میتونی یه تابع سفارشی رو تعریف کنی و این تابع رو روی داده‌های DataFrame اعمال کنی و نمودارهایی با توجه به این تابع ایجاد کنی.</p>
<pre>seaborn.regplot(data=None, *, x=None, y=None, x_estimator=None, x_bins=None, x_ci='ci', scatter=True, fit_reg=True, ci=95, n_boot=1000, units=None, seed=None, order=1, logistic=False, lowess=False, robust=False, logx=False, x_partial=None, y_partial=None, truncate=True, dropna=True, x_jitter=None, y_jitter=None, label=None, color=None, marker='o', scatter_kws=None, line_kws=None, ax=None)</pre>	<p>نمودار رگرسیون یا Regression Plot، واسه نشون دادن رابطه بین دو تا متغیر استفاده میشه. در این نمودار، محور x به یک متغیر مستقل و محور y به یک متغیر وابسته اختصاص داده میشه. این نمودار به صورت خطی یا غیرخطی رسم میشه و خط رسم شده، بهترین تقریب برای رابطه بین دو تا متغیر هست. از این نمودار در آمار و همچنین در یادگیری ماشین واسه پیدا کردن رابطه بین دو تا متغیر استفاده میشه.</p>

squarify چیه؟

کتابخانه "Squarify" یک کتابخانه در زبان برنامه‌نویسی پایتون هست که برای ایجاد نمودارهای **Treemaps** استفاده میشه. Treemaps به نوع نمودار داده‌های سلسله‌مراتبی هست که به شکل یه سری مستطیل نمایش داده می‌شه و به کاربر این امکان رو میده تا داده‌های مختلف رو در سلسله‌مراتب و به شکل توزیع مساحتی بر روی مستطیل‌ها نمایش بده. Squarify برای ایجاد این نوع نمودارها از ترکیب و ترتیب مساحت‌های مستطیل‌ها بر اساس داده‌های ورودی استفاده میکنه. از Squarify میشه واسه نمایش داده‌های تاریخی، سلسله‌مراتبی و مفاهیمی که به ترتیب مرتبط با همدیگه هستن استفاده کرد. این کتابخونه برای افرادی که به دنبال نحوه نمایش داده‌ها در قالب Treemaps با انعطاف بالا هستن، مناسبه.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import squarify
%matplotlib inline
```

کتابخونه‌های pandas، seaborn، matplotlib و squarify و عبارت جادویی رو

وارد میکنیم.

تمرین اول:

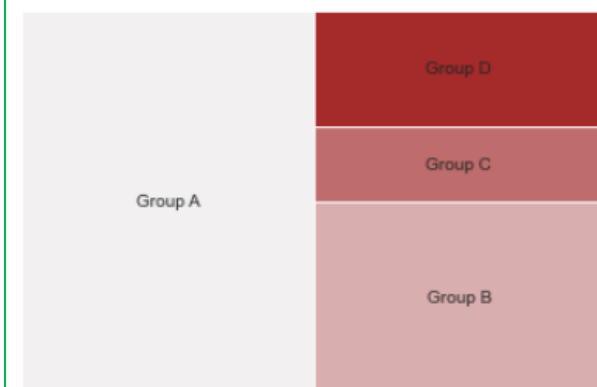
با `sns.light_palette()` پالت‌های رنگی بهش اختصاص میدیم.

ازش با `squarify.plot()` پلات میسازیم. لازم هست که سایز مستطیل‌ها و همینطور برجسبهاشون و رنگشون رو

تو پرانتز بیاریم.

محورش رو off میکنیم که خط نیفته و بعد نمایشش میدیم.

```
colors = sns.light_palette('brown', 4)
squarify.plot(sizes = [50,25,10,15], label = ["Group A", "Group B", "Group C", "Group D"],
              color = colors)
plt.axis('off')
plt.show()
```



تمرین دوم:

میخوایم ۲۰ تا کارت پوکمون pokemon برتر رو بگیریم و یه نقشه درختی بر اساس نوع اولیه ۲۰ پوکمون برتر ایجاد کنیم.

با `pd.read_csv()` وارد میکنیم و اسمش رو میذاریم `dataset`.

بعد با `pd.DataFrame()` به دیتافریم به اسم `df` تبدیلش میکنیم. اگه بخوای جدول رو ببینی باید تو یه خط

مجزا بنویسی `df` و شیفت اینتر کنی که جدول دیده شه.

تو مرحله بعد ۲۰ تا پوکمون رو از سه تا ستون انتخاب میکنیم و ارزش میخوایم که بر اساس ستون `base_total` برامون از پایین به بالا مرتب کنه و ۲۰ تا پوکمون برتر رو بهمون بده که باید بنویسیم `[20:]`.
 یه فیگور براش میسازیم و بهش اندازه میدیم.
 محور رو `off` میکنیم.
 حالا با `squarify.plot` ارزش پلات درست میکنیم. نوع پوکمونها که تو ستون `type1` اومده برامون مهم هست. برای رنگ دادن بهش هم از `seaborn` استفاده میکنیم. `sns.color_palette()` طول هر مستطیل رو هم بر اساس فیلد `type1` بهش میدیم. `Pad` هم براش در نظر میگیریم که بین مستطیلهای فاصله بیفته. با `text_kwargs` میتونی اندازه، رنگ و مکان متنها رو سفارشی سازی کنی.
 در نهایت هم میتونی برای نقشه درختی عنوان مشخص کنی و سایز هم بهش بدی.

```
dataset = pd.read_csv("pokemon.csv")
df = pd.DataFrame(dataset)
top20_pokemon = df.loc[:, ["name", "base_total", 'type1']].sort_values(by="base_total", ascending=False)[:20]
plt.figure(figsize=(12, 6))
plt.axis("off")
axis = squarify.plot(top20_pokemon['type1'].value_counts(),
                     label=top20_pokemon['type1'].value_counts().index,
                     color=sns.color_palette("tab20", len(top20_pokemon['type1'].value_counts())), pad=1,
                     text_kwargs={'fontsize': 18})
axis.set_title("Primary Data Types Of Top 20 Pokemons", fontsize=24)
plt.show()
```

Primary Data Types Of Top 20 Pokemons



خلاصه کدهای این بخش

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

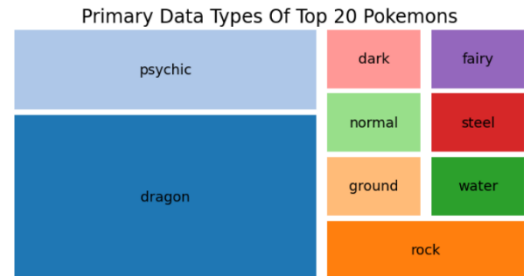
تمرین ۱

```
colors = sns.light_palette('brown', 4)
squarify.plot(sizes = [50,25,10,15], label = ["Group A", "Group B",
"Group C", "Group D"], color = colors)
plt.axis('off')
plt.show()
```



تمرین ۲

```
dataset = pd.read_csv("pokemon.csv")
df = pd.DataFrame(dataset)
top20_pokemon = df.loc[:, ["name",
"base_total", 'type1']].sort_values(by="base_total",
ascending=False)[:20]
plt.figure(figsize=(12, 6))
plt.axis("off")
axis = squarify.plot(top20_pokemon['type1'].value_counts(),
label=top20_pokemon['type1'].value_counts().index,
color=sns.color_palette("tab20", len(top20_pokemon['type1'].value_counts())), pad=1,
text_kwargs={'fontsize': 18})
axis.set_title("Primary Data Types Of Top 20 Pokemons", fontsize=24)
plt.show()
```



خلاصه دستورهایی squarify

توضیحات	توابع و متدها
برای ایجاد نمودارهای Treemaps یا نمودارهای سلسله‌مراتبی مستطیل شکل. همیشه بهش اندازه، رنگ و برچسب داد و همینطور با pad فاصله بین مستطیلها رو هم مشخص کرد.	squarify.plot()

missingno چیه؟

missingno یک کتابخونه مفید و ساده واسه زبان برنامه‌نویسی پایتون هست که ابزارها و تجسم‌های مختلفی ارائه می‌ده تا میزان و توزیع داده‌های از دست رفته در مجموعه داده‌ها رو مشخص کنه. این کتابخونه به مصورسازی اطلاعات از دست رفته مثل مقادیر **NaN (Not a Number)** و **NULLs** و **None** کمک می‌کنه. با استفاده از missingno، می‌تونی نقاط مشخصی از داده‌های خودت رو که اطلاعاتی از دست رفته دارن شناسایی و مصورسازی کنی.

از قابلیت‌های missingno مصورسازی توزیع مقادیر NaN، نمودار ماتریس از دست رفته و نمودار توزیع مقادیر NaN در داده‌ها هست. این ابزار برای تحلیل داده‌ها و تصمیم‌گیری‌های بهتر در زمینه تجزیه و تحلیل داده‌ها خیلی مفید هست.

می‌تونی از missingno واسه بهبود کیفیت داده‌ها و شناسایی الگوهای از دست رفته در داده‌ها استفاده کنی.

به داده‌هایی که اساساً اشتباه هستن و باید برطرف شن می‌گن **Noise** و به داده‌های پرت که صحیح هستن ولی از بقیه داده‌ها دود افتادن یا فاصله دارن می‌گن **outlier**. واسه اینکه دقتمون تو تجزیه و تحلیل داده‌ها بیشتر شه باید این مشکل هم برطرف شه.

کتابخونه missingno ۴ تا نمودار داره:

۱- **Bar chart**: تعداد مقادیر موجود در هر ستون رو با نادیده گرفتن مقادیر از دست رفته نمایش میده. همچنین درصدهایی رو در محور Y نشون میده که بهت این امکان رو میده که مقدار مقادیر مناسب/فقدان مقادیر در هر ستون رو درک کنی.

۲- **Matrix**: نمودار ماتریس پوچ یا nullity بهت این امکان رو میده که توزیع داده‌ها رو در کل مجموعه داده در همه ستونها به طور همزمان درک کنی. علاوه بر اون در درک بهتر توزیع مقادیر از دست رفته در داده‌ها کمک کننده است. یه sparkline رو هم نمایش میده که ردیف‌هایی رو با حداکثر و حداقل بی اعتباری یا فقدان در یک مجموعه داده برجسته می‌کنه.

۳- **Heatmap**: نمودار همبستگی بی اعتباری بین ستون‌های مجموعه داده رو نشون میده. بهت کمک میکنه که بفهمی که چطوری مقدار از دست رفته یک ستون با مقادیر از دست رفته در سایر ستونها مرتبط هست. sparkline بهت کمک میکنه تا بهتر بفهمی که در کجای مجموعه داده ما مقادیر زیادی از دست رفته وجود داره چون حرارت بالایی رو در اون بخش نشون میده.

۴- **Dendrogram**: دندروگرام مثل نقشه حرارتی ستون‌ها رو بر اساس رابطه پوچ بین اونها گروه بندی میکنه. در واقع ستون‌هایی رو گروه‌بندی میکنه که در اونها رابطه nullity بیشتری وجود داره. تقریباً مثل خوشه‌بندی سلسله مراتبی عمل می‌کنه، ولی از همبستگی nullity برای خوشه‌بندی استفاده می‌کنه که ستون‌هایی رو با همون توزیع مقادیر گمشده در یک خوشه نگه می‌داره.

تمرین اول:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import missingno as msno
%matplotlib inline
```

کتابخونه‌های pandas, numpy, matplotlib و missingno رو

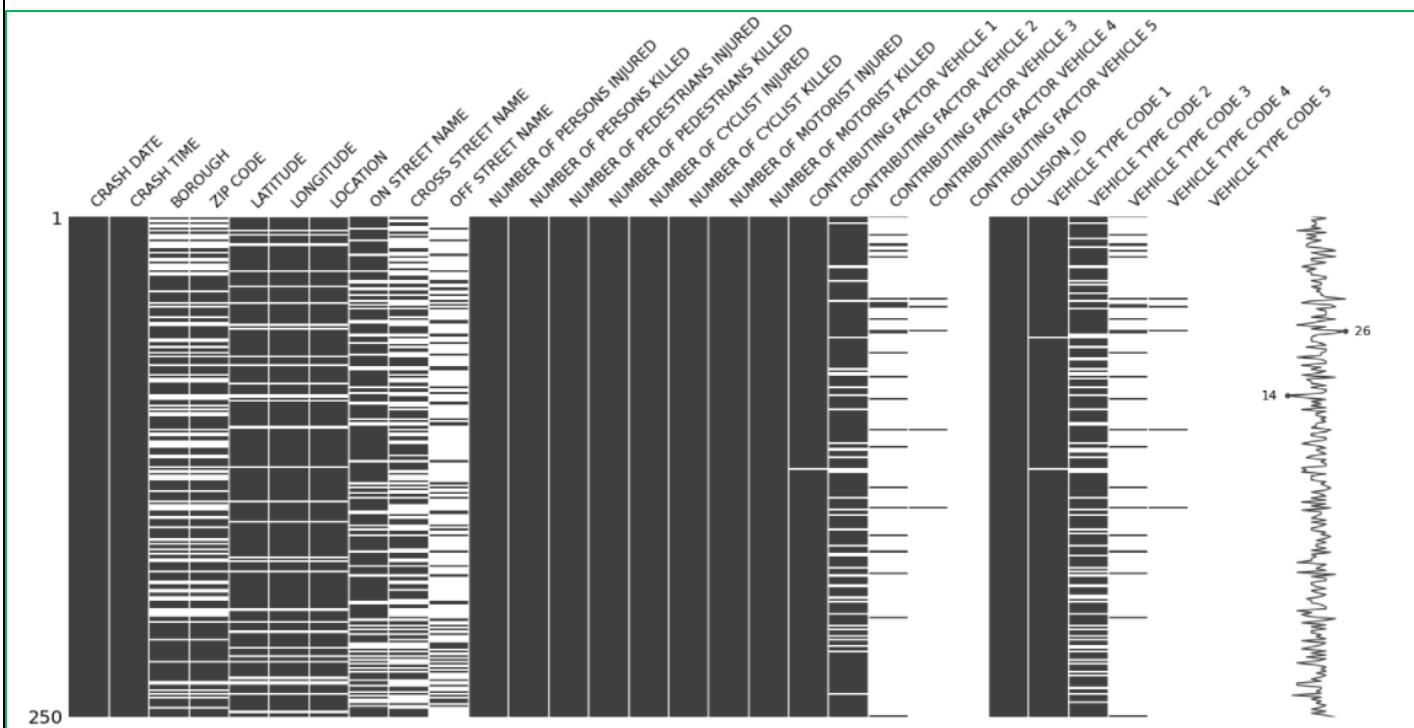
به عنوان msno بیار.

میخوایم از ماتریس `msno.matrix()` استفاده کنیم. این ماتریس کمک میکند که مقادیر NaN در یک مجموعه داده مصورسازی شوند. با استفاده از این قابلیت میشه به نمودار از داده‌ها رو ایجاد کرد که نشون میده که در کدام قسمت‌های مجموعه داده مقادیر از دست رفته داریم و یا اطلاعاتی وجود نداره.

```
collisions = pd.read_csv("Motor_Vehicle.csv")
msno.matrix(collisions.sample(250))
plt.show()
```

یه فایل CSV به اسم `Motor_Vehicle` اضافه میکنیم. اسمش میذاریم `collisions`. بعد با ماتریس `msno.matrix()` نمودارش رو ترسیم میکنیم.

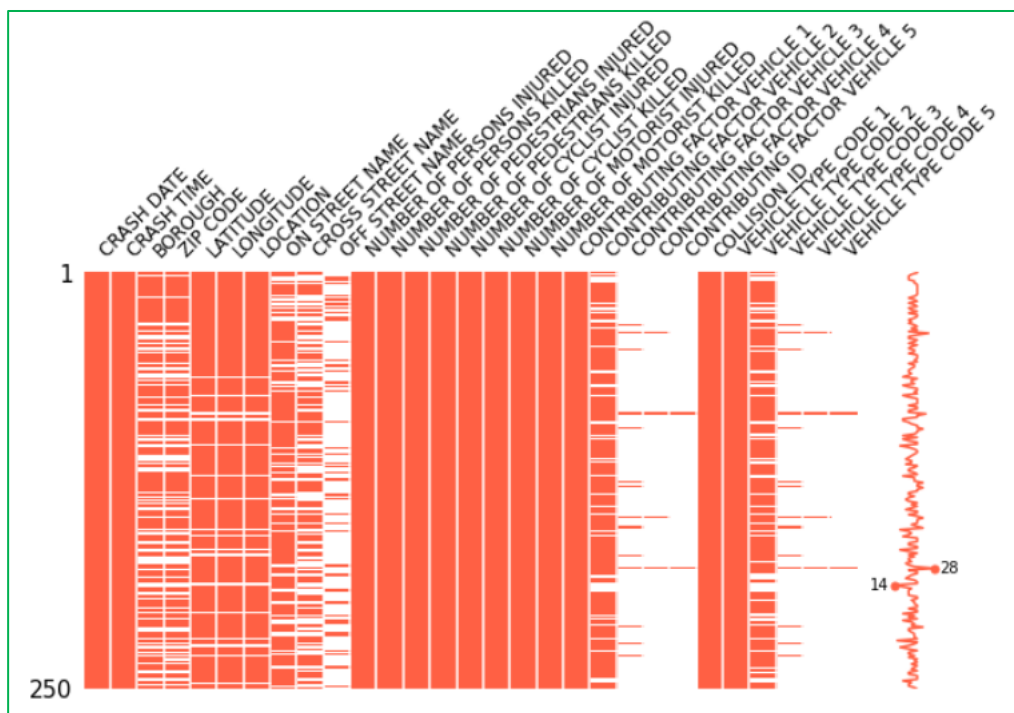
بعضی از ستونها پر هستن و بعضی دیگه لکه‌دار به نظر می‌آن. `Sparkline` که سمت راست نمودار هست به ردیفهایی با حداکثر و حداقل بی‌اعتباری داده‌ها اشاره میکنه. این مصورسازی تا ۵۰ تا متغیر رو پوشش میده ولی بیشتر از اون متغیر داشته باشیم داده‌ها ناخوانا میشن.



میتونی به نمودار رنگ و اندازه شکل و اندازه فونت هم بدی.

اگه بخوای `sparklingline` رو بهت نشون نده تو پرانتز بنویس `sparkling = False`.

```
collisions = pd.read_csv("Motor_Vehicle.csv")
msno.matrix(collisions.sample(250), figsize=(10,5), fontsize=12,
            color=(1,0.38,0.27))
plt.show()
```



تمرین دوم:

اگر با داده‌های سری زمانی کار میکنی، میتونی یک دوره تناوب رو با استفاده از پارامتر کلمه کلیدی freq مشخص کنی.

```
null_pattern = (np.random.random(1000).reshape((50, 20)) > 0.5).astype(bool)
null_pattern = pd.DataFrame(null_pattern).replace({False: None})
null_pattern
```

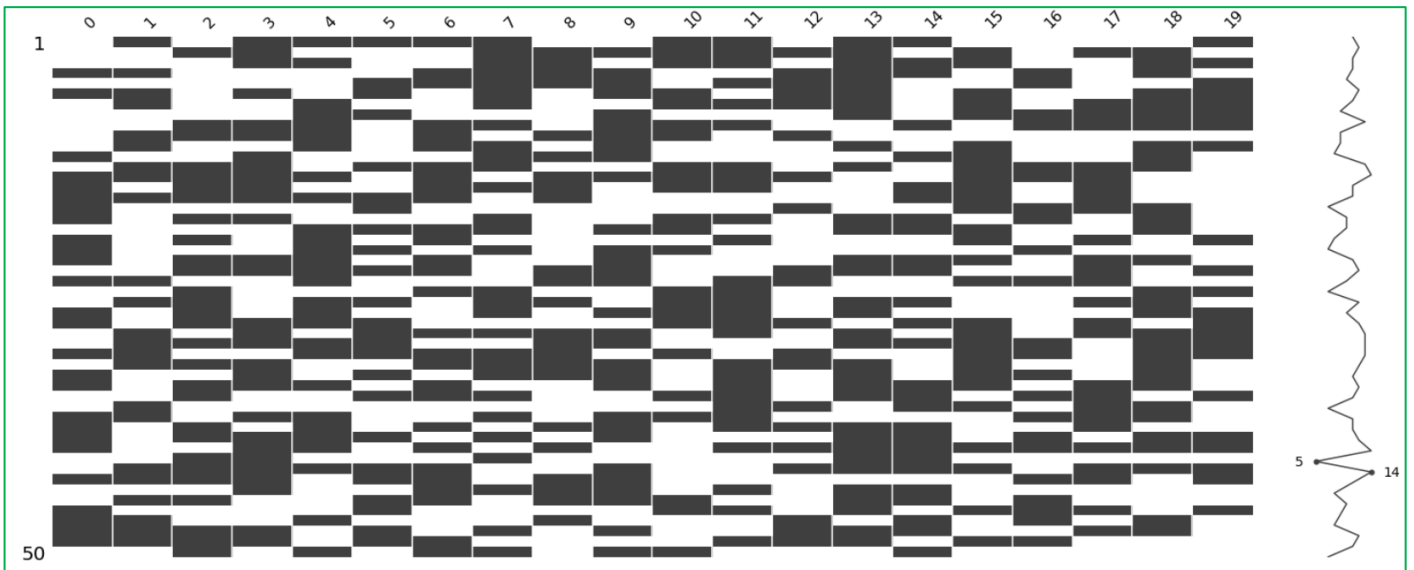
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	True	True	True	True	True	True	True	None	None	None	None	True	None	True	None	None	True	None	None	None
1	None	True	True	True	True	None	None	None	True	None	None	None	None	None	True	True	True	None	True	None
2	True	True	None	True	None	None	True	True	None	True	None	True	None	None	None	None	True	None	None	True
3	True	True	True	None	True	None	True	True	None	None	True	True	True	None	True	None	True	None	None	None
4	None	None	True	True	True	None	None	True	None	True	None	True	True	True	True	None	True	True	None	True
5	True	None	None	True	True	True	None	True	True	True	None	True	None	None	None	True	None	None	None	True
6	None	True	True	True	True	True	True	True	None	True	None	True	True	None	True	True	None	None	True	None
7	None	None	None	None	True	None	True	True	True	True	True	None	None	None	True	True	None	True	None	True
8	None	None	None	None	None	True	True	True	True	None	None	None	True	True	True	None	None	True	True	True
9	None	None	True	True	None	True	None	None	None	True	None	None	None	None	True	True	None	None	True	None
10	None	True	None	None	True	None	True	True	True	True	None	None	True	True	None	True	True	None	True	True
11	None	None	True	None	None	None	True	True	True	True	None	True	True	True	None	True	True	None	True	None
12	None	None	True	True	True	None	True	None	True	True	True	None	True	True	None	True	None	None	True	None
13	True	None	True	None	None	None	None	None	True	None	None	True	None	None	None	True	True	True	True	None

اول به سری داده با np به یه متغیر اختصاص میدیم. و بعد با DataFrame میخوایم که داده‌ها رو برامون نگه داره و اطلاعات رو تبدیل کنه به جدولی از داده‌های True و False.

بعد با msno.matrix() تبدیلیش میکنیم به نمودار. از frag هم واسه اینکه نمودار رو به صورت سری زمانی بهمون بده استفاده میکنیم.

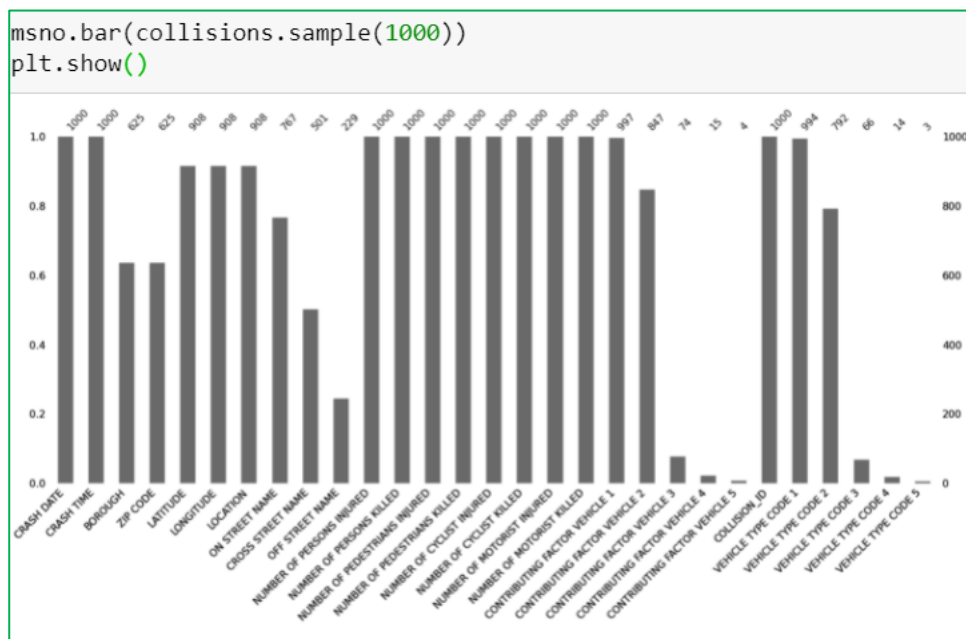
```
msno.matrix(null_pattern.set_index(pd.period_range('1/1/2011', '2/1/2015', freq='M')))
plt.show()
```


با plt.show() نمایشش میدیم.



تمرین سوم:

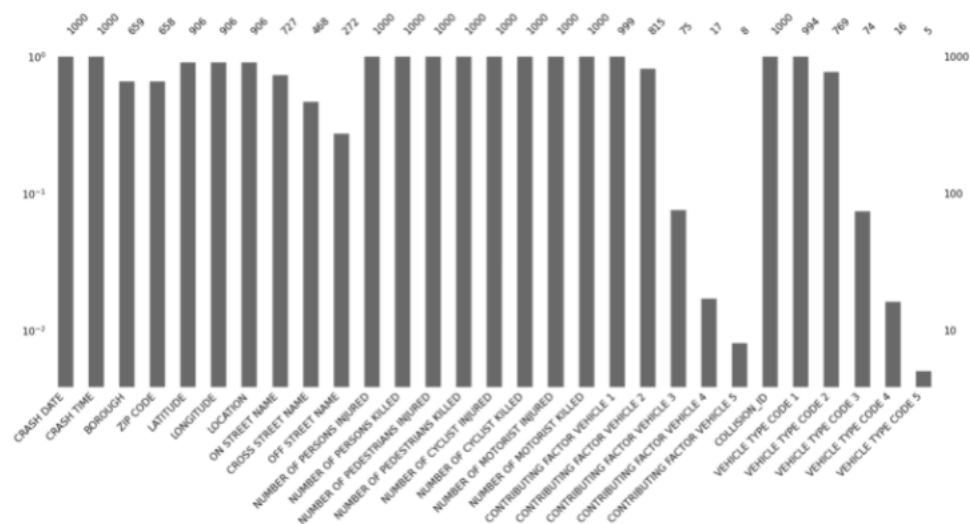
میخوایم با کمک msno.bar() داده‌های گم شده رو به صورت ستونی نشون بدیم.



اگه توی پرانتزش بنویسیم log = True نمودار اطلاعات مشابهی رو برامون فراهم میکنه ولی تو یه فرمت ساده‌تر.

```
msno.bar(collisions.sample(1000), log=True)
plt.show()
```

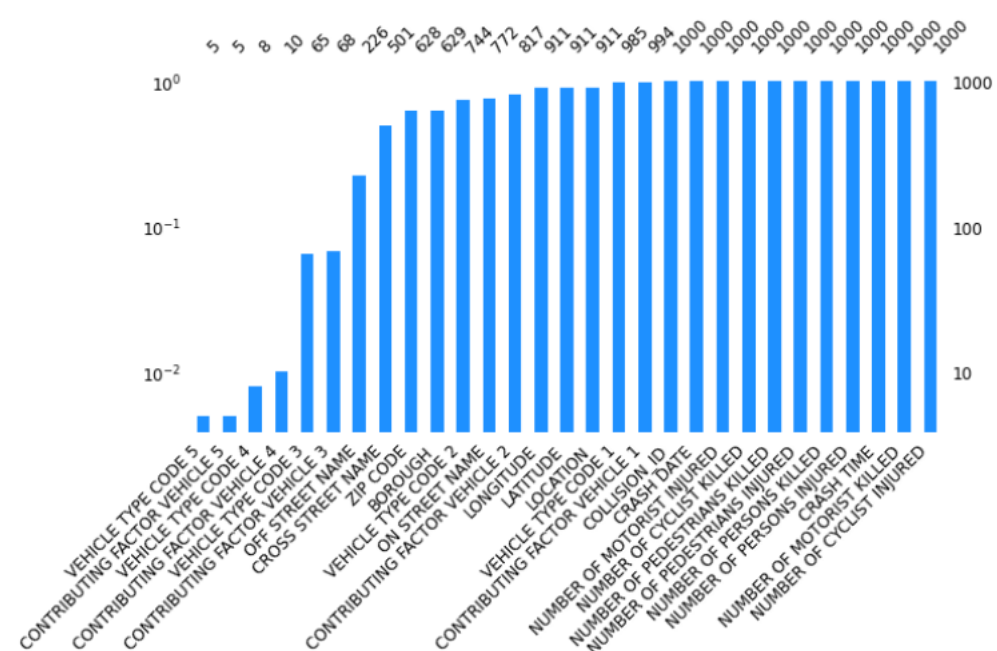
E:\ArcGIS Pro\bin\Python\envs\gisenv\lib\site-packages\missingno\missingno.py:271: UserWarning: FixedFormatter should only be used together with FixedLocator
 ax2.set_yticklabels([int(n * len(df)) for n in ax1.get_yticks()], fontsize=fontsize)



میتونی به نمودارت رنگ بدی یا بگی که از ستون کوچیک به بزرگ مرتب کنه یا اندازه شکلیش چقدر باشه.

```
msno.bar(collisions.sample(1000), log=True, color="dodgerblue",
          sort="ascending", figsize=(10,5), fontsize=12)
plt.show()
```

E:\ArcGIS Pro\bin\Python\envs\gisenv\lib\site-packages\missingno\missingno.py:271: UserWarning: FixedFormatter should only be used together with FixedLocator
 ax2.set_yticklabels([int(n * len(df)) for n in ax1.get_yticks()], fontsize=fontsize)



با fig میتونی چند تا نمودار میله‌ای با هم داشته باشی.

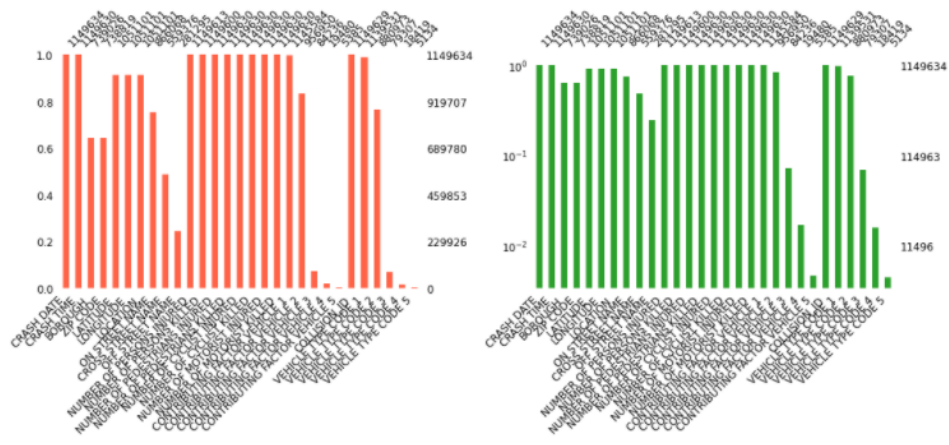
```
fig = plt.figure(figsize=(15,7))

ax1 = fig.add_subplot(1,2,1)
msno.bar(collisions, color="tomato", fontsize=12, ax=ax1);

ax2 = fig.add_subplot(1,2,2)
msno.bar(collisions, log=True, color="tab:green", fontsize=12, ax=ax2);

plt.tight_layout()
plt.show()
```

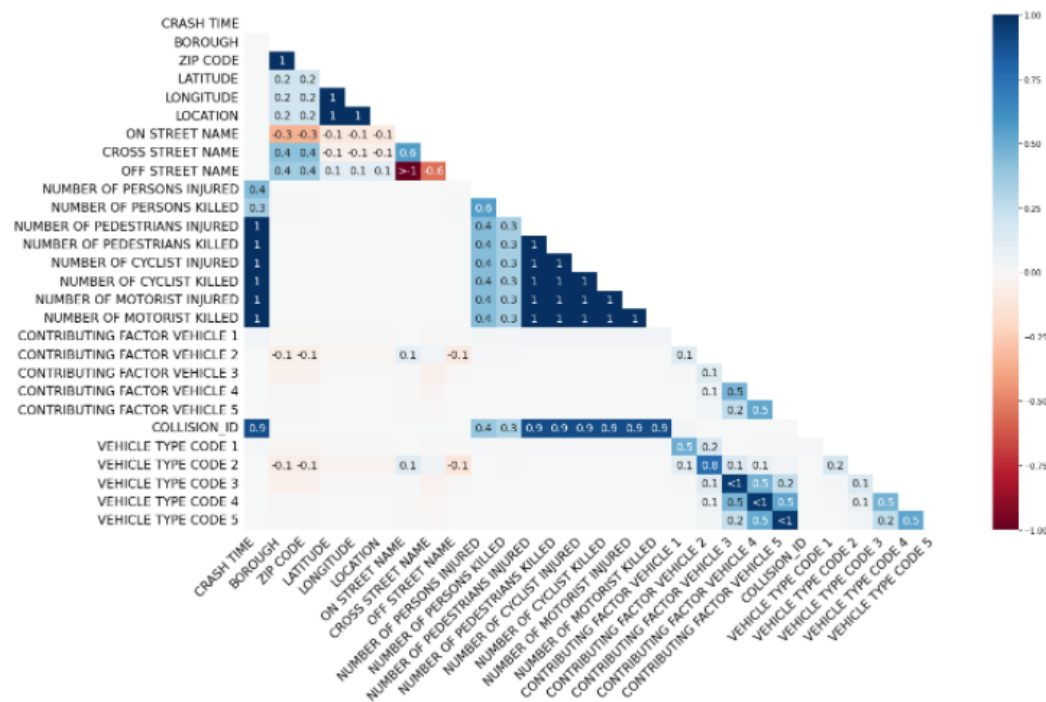
E:\ArcGIS Pro\bin\Python\envs\gisenv\lib\site-packages\missingno\missingno.py:271: UserWarning: FixedFormatter should only be used together with FixedLocator
ax2.set_yticklabels([int(n * len(df)) for n in ax1.get_yticks()], fontsize=fontsize)



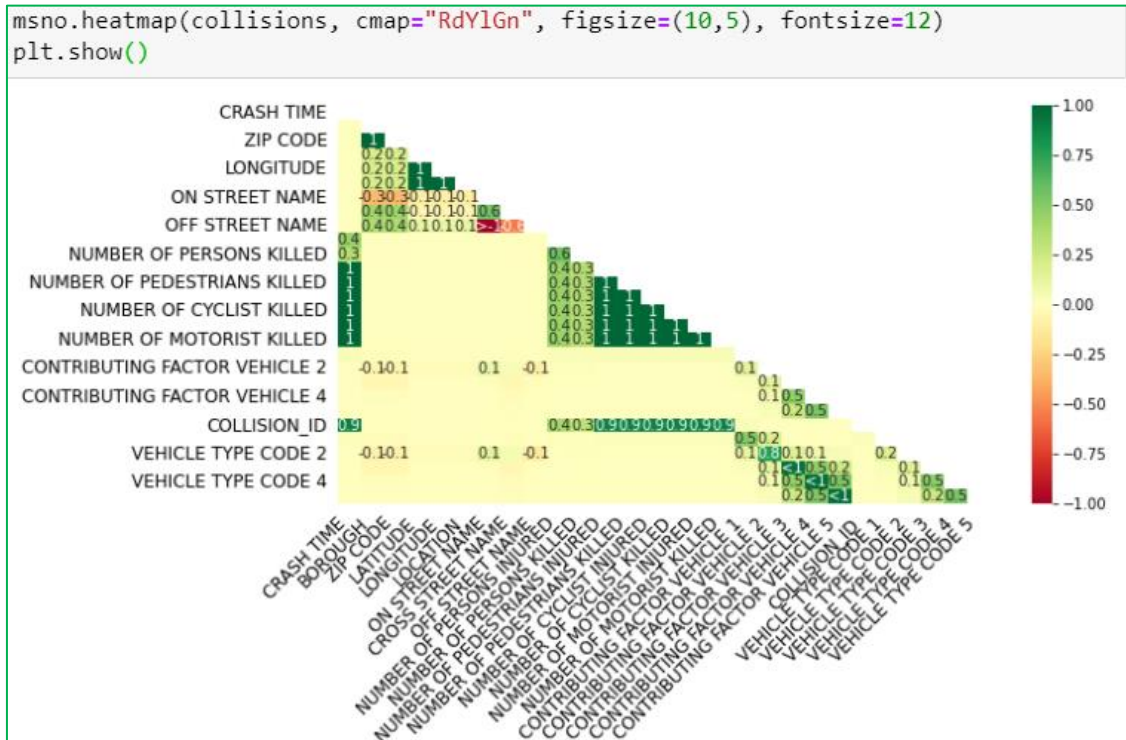
تمرین چهارم:

میخوایم با `msno.heatmap()` همبستگی nullity (نداشتن داده) یا به عبارت دیگر تأثیر نداشتن یا داشتن مقادیر یک متغیر بر عدم حضور یا حضور مقادیر دیگر را مصورسازی کنیم.

```
msno.heatmap(collisions)
plt.show()
```



میتونی به نمودار cmap و اندازه فونت و اندازه تصویر هم بدی:



این نمودار نشون میده که متغیرهای با فیلدهای Off street name احتمالش کم هست که داده‌های کاملی داشته باشن.

همبستگی nullity یا عدم وجود داده:

🚦 ۱- (اگه یک متغیر ظاهر بشه مطمئناً متغیر دیگه ظاهر نمیشه)

🚦 • (اگه یه متغیر ظاهر بشه یا نشه تاثیری روی بقیه نداره).

🚦 ۱ (اگر یک متغیر ظاهر بشه، متغیر دیگر قطعاً ظاهر میشه یعنی روش تاثیر میذاره).

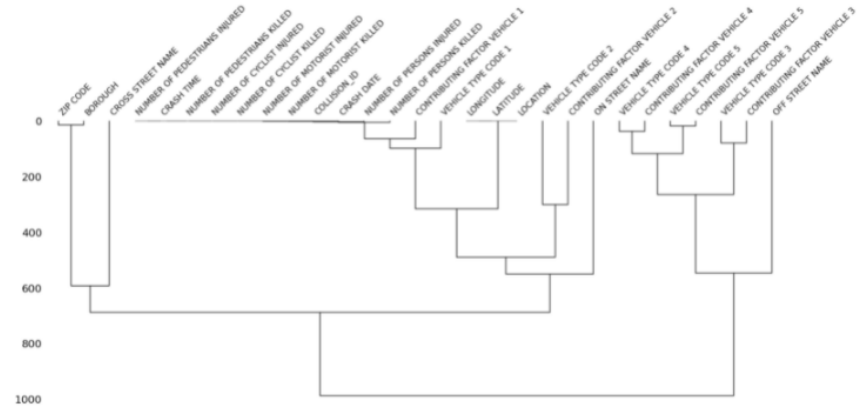
Heatmap برای انتخاب روابط کامل بودن داده‌ها بین جفت‌های متغیر عالی عمل میکنه، ولی قدرت توضیحیش در رابطه با روابط بزرگتر محدود هست و پشتیبانی خاصی برای مجموعه داده‌های بسیار بزرگ نداره.

تمرین پنجم:

دندروگرام Dendrogram بهت یه نمودار سلسله‌مراتبی میده که با استفاده از الگوریتم خوشه‌بندی یه نمودار سلسله‌مراتبی ایجاد کنی. این الگوریتم بهت کمک میکنه که از تشابه بین متغیرها استفاده بشه که اساس توزیع مقادیر از دست رفته این متغیرها محاسبه بشه. در واقع بهتر از Heatmap عمل میکنه. در واقع این نمودار نشون میده که چه ستون‌هایی با هم مشابه‌ترین توزیع از داده‌های از دست رفته رو دارن.

کدش به صورت زیر نوشته میشه:

```
msno.dendrogram(collisions)
plt.show()
```

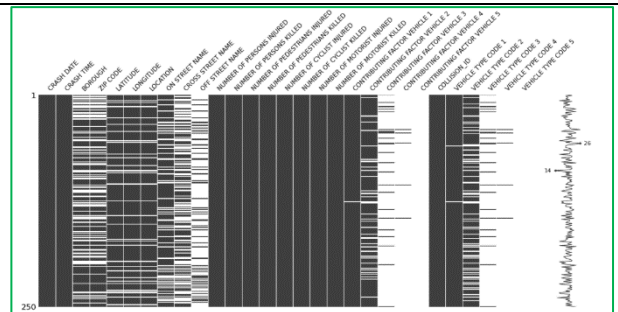


خلاصه کدهای این بخش

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
%matplotlib inline
```

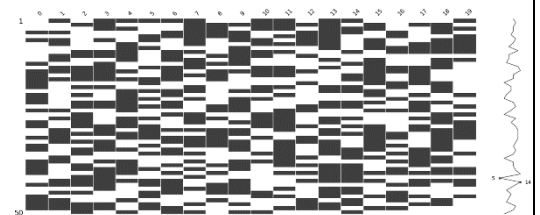
تمرین ۱

```
collisions = pd.read_csv("Motor_Vehicle.csv")
msno.matrix(collisions.sample(250))
plt.show()
```



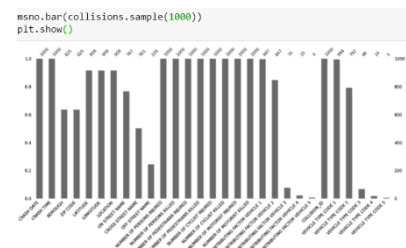
تمرین ۲

```
null_pattern = (np.random.random(1000).reshape((50, 20)) > 0.5).astype(bool)
null_pattern = pd.DataFrame(null_pattern).replace({False: None})
msno.matrix(null_pattern.set_index(pd.period_range('1/1/2011', '2/1/2015', freq='M')))
plt.show()
```



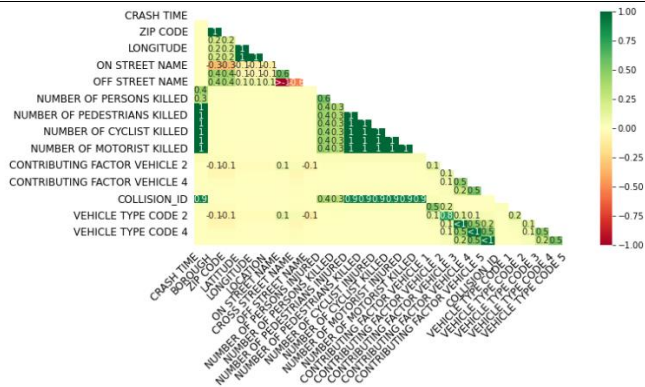
تمرین ۳

```
msno.bar(collisions.sample(1000))
plt.show()
```



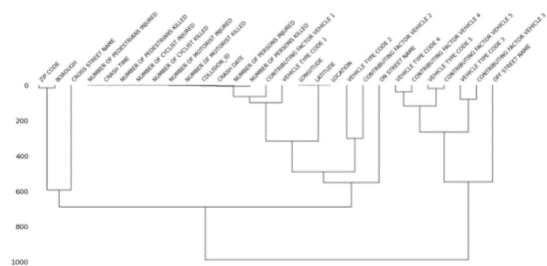
تمرین ۴

```
msno.heatmap(collisions, cmap="RdYlGn",
figsize=(10,5), fontsize=12)
plt.show()
```



تمرین ۵

```
msno.dendrogram(collisions)
plt.show()
```



خلاصه دستورهایی missingno

توابع و متدها	توضیحات
1- <code>msno.bar()</code> 2- <code>msno.matrix()</code> 3- <code>msno.heatmap()</code> 4- <code>msno.dendrogram()</code>	<p>تابخونه missingno ۴ تا نمودار داره:</p> <p>۵- Bar chart: تعداد مقادیر موجود در هر ستون رو با نادیده گرفتن مقادیر از دست رفته نمایش میده. همچنین درصدهایی رو در محور Y نشون میده که بهت این امکان رو میده که مقدار مقادیر مناسب/فقدان مقادیر در هر ستون رو درک کنی.</p> <p>۶- Matrix: نمودار ماتریس پوچ یا nullity بهت این امکان رو میده که توزیع داده‌ها رو در کل مجموعه داده در همه ستونها به طور همزمان درک کنی. علاوه بر اون در درک بهتر توزیع مقادیر از دست رفته در داده‌ها کمک کننده است. یه sparkline رو هم نمایش میده که ردیف‌هایی رو با حداکثر و حداقل بی اعتباری یا فقدان در یک مجموعه داده برجسته می‌کنه.</p> <p>۷- Heatmap: نمودار همبستگی بی اعتباری بین ستون‌های مجموعه داده رو نشون میده. بهت کمک میکنه که بفهمی که چطوری مقدار از دست رفته یک ستون با مقادیر از دست رفته در سایر ستونها مرتبط هست. sparkline بهت کمک میکنه تا بهتر بفهمی که در کجای مجموعه داده ما مقادیر زیادی از دست رفته وجود داره چون حرارت بالایی رو در اون بخش نشون میده.</p> <p>۸- Dendrogram: دندروگرام مثل نقشه حرارتی ستون‌ها رو بر اساس رابطه پوچ بین اونها گروه بندی میکنه. در واقع ستون‌هایی رو گروه‌بندی میکنه که در اونها رابطه nullity بیشتری وجود داره.</p>

	تقریباً مثل خوشه‌بندی سلسله مراتبی عمل می‌کنه، ولی از همبستگی nullity برای خوشه‌بندی استفاده می‌کن که ستون‌هایی رو با همون توزیع مقادیر گمشده در یک خوشه نگه می‌داره.
--	---

تا اینجا ۶ تا از مهمترین کتابخانه‌های کاربردی پایتون در **data Science** رو با هم یاد گرفتیم. تو جزوه بعدی می‌خوایم بریم سراغ خود Data Science یا علم داده که ببینیم چی هست و با این کتابخانه‌ها چطوری میشه پروژه‌های کاربردی انجام داد.