Desafio Final - Imersão Dados 3_Alura

#imersaoDados

#alura

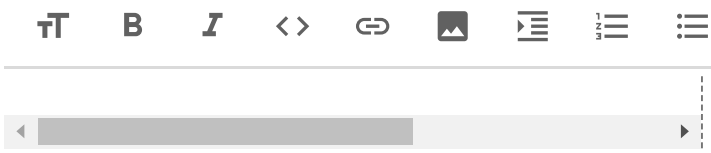
Projeto de Data Science com Análise Exploratória e proposta de um modelo de Machine Learning

Projeto ALURA inspirado em base de dados do 'Laboratory Innovation Science at Harvard', dados disponibilizados no kaggle.

Tutores da Alura: Guilherme Silveira, Thiago G. Santos, Vanessa Leiko.

Wiram Ligeiro -- https://www.linkedin.com/in/wiram-ligeiro-51565636/

Clique duas vezes (ou pressione "Enter") para editar



Clique duas vezes (ou pressione "Enter") para editar

Introdução

Rodmap dos estudos desenvolvidos durante a ImersãoDados
Análise de dados, Python, Pandas e novos fármacos; Estatísticas, Dados e distribuições; Correlações Causalidades e Relações entre genes; Merge e Análise de resultados; Machine Learning e SciKit-Learn.

***Quanto a inovação na área dos fármacos ***

Empresas farmacêuticas desenvolvem permanentemente novas pequisas e inovações (drug discovery) de forma independente assim como em parceria mútua com outras organizaçãoes, universidades e institutos de pequisas que impactam positivamente panoramas e estudos clínicos. Inovações tecnológicas surgem de testes mais eficientes, que podem diminuir o tempo de estudo, além de resultar em dados mais ricos e precisos. As técnicas de análise de dados são ferramentas fundamentais para a melhor fluidez e validação de resultados nas diferentes fases e intermediárias e finais das pesquisas.

***Pipeline para a pesquisa de novos medicamentos ***

Com a definição de um determinada patologia a ser pesquisada e tratada, identifica-se o foco atuação da droga a ser perquisada; A seleção de compostos ou agentes biológicos a serem trabalhados em estruturas celulares, em laboratório; Daí parte-se para a triagem de modo a confrontar as substâncias alinhadas ao foco da pesquisa ou para outros experimentos; Desenvolvem-se a seuir testes adicionais laboratoriais dos compostos ou elementos que atuaram em colônicas de células como candidatos aos melhores resultados para o combate da patologia.

Escopo do Projeto ** ***texto em itálico*

Este projeto objetiva tratar do modelo de ML para auxiliar novas pesquisas similares.

Numa primeira etapa foi verificado as condições dos dados da DB disponibilizado, a avaliação quanto a sua estruturação, a avaliação quanto a agregação e desagregação dos dados, a limpeza, a identificação de 'outliers', a verificação quanto a dados ausentes dos registros disponibilizados, a eventual existência de duplicação de dados, no confronto entre algumas varáveis, e a definição de métodos adotados nesta etapa dos trabalhos. Assim, o foco do projeto, como mencionado, será o de avaliar uma proposta de um modelo de ML que possa bem validar os resultados da pesquisa bioquímica de desenvolvimento de novos fármacos.

Glossário: (Contribuição do Gustavo Quadra) para favorecer o entendimento.

id : identificador para um experimento

G's : expressão gênica = Processo pelo qual a informação hereditária de um gene, forma uma proteína ou RNA

C's : viabilidade celular = Analisar células de uma cultura celular, afim de avaliar sua atividade.

Composto / droga = Placebo (talvez) para casos com controle e remédios / substâncias diferentes para casos com droga;

MOA : Mecanismo de ação do Alvo : interação bioquímica entre uma droga utilizada e um alvo

Clique duas vezes (ou pressione "Enter") para editar

Etapa 1 - Análise de dados, python, pandas e novos fármacos

Impotação dos dados

```
import pandas as pd

url_dados = 'https://github.com/alura-cursos/imersaodados3/blob/main/dados/dados_experimen

dados = pd.read_csv(url_dados, compression='zip')
dados
```

	id	tratamento	tempo	dose	droga	g-0	g-1	g-2	
0	id_000644bb2	com_droga	24	D1	b68db1d53	1.0620	0.5577	-0.2479	-0.(
1	id_000779bfc	com_droga	72	D1	df89a8e5a	0.0743	0.4087	0.2991	0.(
2	id_000a6266a	com_droga	48	D1	18bb41b2c	0.6280	0.5817	1.5540	-0.(
3	id_0015fd391	com_droga	48	D1	8c7f86626	-0.5138	-0.2491	-0.2656	0.5
4	id_001626bd3	com_droga	72	D2	7cbed3131	-0.3254	-0.4009	0.9700	0.(
...	...	...	...	...	...	...	...	...	
23809	id_fffb1ceed	com_droga	24	D2	df1d0a5a1	0.1394	-0.0636	-0.1112	-0.5
23810	id_fffb70c0c	com_droga	24	D2	ecf3b6b74	-1.3260	0.3478	-0.3743	0.9
23811	id_fffc1c3f4	com_controle	48	D2	cacb2b860	0.3942	0.3756	0.3109	-0.1
23812	id_fffcb9e7c	com_droga	24	D1	8b87a7a83	0.6660	0.2324	0.4392	0.2
23813	id_ffffdd77b	com_droga	72	D1	972f41291	-0.8598	1.0240	-0.1361	0.1

23814 rows × 877 columns

```
dados.head()
```

	id	tratamento	tempo	dose	droga	g-0	g-1	g-2	g-3
0	id_000644bb2	com_droga	24	D1	b68db1d53	1.0620	0.5577	-0.2479	-0.6208
1	id_000779bfc	com_droga	72	D1	df89a8e5a	0.0743	0.4087	0.2991	0.0604
2	id_000a6266a	com_droga	48	D1	18bb41b2c	0.6280	0.5817	1.5540	-0.0764

```
dados.shape
```

```
(23814, 877)
```

```
dados['tratamento']
```

```
0           com_droga
1           com_droga
2           com_droga
3           com_droga
4           com_droga
             ...
23809       com_droga
23810       com_droga
23811    com_controle
23812       com_droga
23813       com_droga
Name: tratamento, Length: 23814, dtype: object
```

Tipos de entidades, atributos ou valores das colunas de dados a analisar.

```
dados['tratamento'].unique()
```

```
array(['com_droga', 'com_controle'], dtype=object)
```

```
dados['tempo'].unique()
```

```
array([24, 72, 48])
```

```
dados['dose'].unique()
```

```
array(['D1', 'D2'], dtype=object)
```

```
dados['droga'].unique()
```

```
array(['b68db1d53', 'df89a8e5a', '18bb41b2c', ..., '573c787a2',
       'b2fe3eca7', 'dd4a96d16'], dtype=object)
```

```
dados['g-0'].unique()
```

```
array([ 1.062 ,  0.0743,  0.628 , ...,  0.3942,  0.666 , -0.8598])
```

Conhecendo-se a distribuição / frequencia de resultados nos experimentos.

```
dados['tratamento'].value_counts()
```

```
com_droga       21948
com_controle     1866
Name: tratamento, dtype: int64
```

```
dados['dose'].value_counts()
```

```
D1    12147
D2    11667
Name: dose, dtype: int64
```

```
dados['tratamento'].value_counts(normalize = True)
```

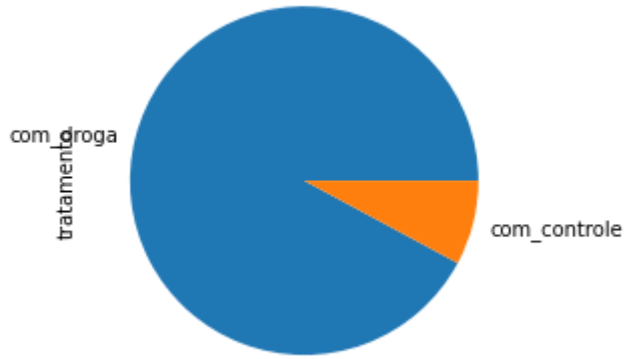
```
com_droga       0.921643
com_controle    0.078357
Name: tratamento, dtype: float64
```

```
dados['dose'].value_counts(normalize = True)
```

```
D1    0.510078
D2    0.489922
Name: dose, dtype: float64
```

```
dados['tratamento'].value_counts().plot.pie()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4721168cd0>
```

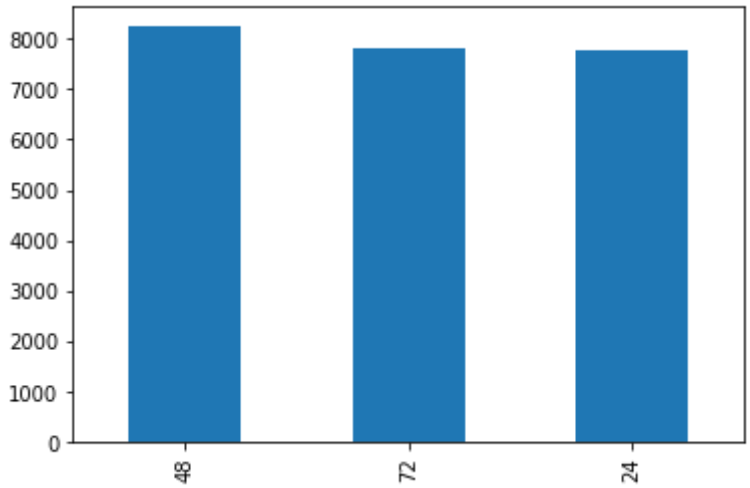



```
dados['tempo'].value_counts().plot.pie()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4721194f10>
```


```
dados['tempo'].value_counts().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f47213b3dd0>
```



Avaliando-se quantos elementos na coluna g-0 são maiores que zero?

```
dados['g-0'] > 0
```

```
0         True
1         True
2         True
3        False
4        False
         ...
23809     True
23810    False
23811     True
23812     True
23813    False
Name: g-0, Length: 23814, dtype: bool
```

```
dados_filtrados = dados[dados['g-0'] > 0]
dados_filtrados.head()
```

Sumário da etapa 1 de exploração dos dados:

Realizado a importação do banco de dados; Exploração inicial dos dados para o estudo dada a especialidade do tema; Conhecimento inicial sobre as classes de informações das colunas do BD; Informações sobre o tamanho do BD - linhas e colunas, e categorias dos dados disponíveis; Verificou-se que experimentos são realizados com o estudo em colônias de células para o desenvolvimento de novas drogas; Foram examinadas proporções entre alguns dados examinados; Destaca-se que os experimentos podem transcorrer num tempo de 24, 48 e 72 horas; Gráficos com biblioteca do pandas; Iniciado a aplicação da função 'query' para filtragem de informações, e finalmento iniciadoo cruzamento de dados para análise; Há desdobramentos possíveis para outras pesquisas, possilvemente a serem sinalizadas na medida em que avancemos na sua análise.

Clique duas vezes (ou pressione "Enter") para editar

Etapa 2 - Estatísticas, visualização de dados e distribuições

Renomear a coluna 'droga' como 'composto'

```
mapa = {'droga' : 'composto'}
dados.rename(columns=mapa)
```

	id	tratamento	tempo	dose	composto	g-0	g-1	g-2	
0	id_000644bb2	com_droga	24	D1	b68db1d53	1.0620	0.5577	-0.2479	-0.(
1	id_000779bfc	com_droga	72	D1	df89a8e5a	0.0743	0.4087	0.2991	0.(
2	id_000a6266a	com_droga	48	D1	18bb41b2c	0.6280	0.5817	1.5540	-0.(

```
dados.head()
```

	id	tratamento	tempo	dose	droga	g-0	g-1	g-2	g-3
0	id_000644bb2	com_droga	24	D1	b68db1d53	1.0620	0.5577	-0.2479	-0.6208
1	id_000779bfc	com_droga	72	D1	df89a8e5a	0.0743	0.4087	0.2991	0.0604
2	id_000a6266a	com_droga	48	D1	18bb41b2c	0.6280	0.5817	1.5540	-0.0764
3	id_0015fd391	com_droga	48	D1	8c7f86626	-0.5138	-0.2491	-0.2656	0.5288
4	id_001626bd3	com_droga	72	D2	7cbed3131	-0.3254	-0.4009	0.9700	0.6919

5 rows × 877 columns

Observa-se que a coluna no DB inicial permanece como droga. Isto porque não foi feito a substituição do data-frame original.

Substituindo-se o nome da coluna no data-frame inicial

```
mapa = {'droga': 'composto'}
dados.rename(columns=mapa, inplace=True)
```

```
dados.head()
```

	id	tratamento	tempo	dose	composto	g-0	g-1	g-2	g-3
0	id_000644bb2	com_droga	24	D1	b68db1d53	1.0620	0.5577	-0.2479	-0.6208
1	id_000779bfc	com_droga	72	D1	df89a8e5a	0.0743	0.4087	0.2991	0.0604
2	id_000a6266a	com_droga	48	D1	18bb41b2c	0.6280	0.5817	1.5540	-0.0764
3	id_0015fd391	com_droga	48	D1	8c7f86626	-0.5138	-0.2491	-0.2656	0.5288
4	id_001626bd3	com_droga	72	D2	7cbed3131	-0.3254	-0.4009	0.9700	0.6919

5 rows × 877 columns

Quanto a modificação da substituição do nome da coluna droga para composto, registra-se um alerta! ... considerar que as funções já rodadas antes da modificação ficam prejudicadas caso tenhamos que retornar acima!!!

```
dados['composto'].value_counts()
```

```
cacb2b860    1866
87d714366     718
9f80f3f77     246
8b87a7a83     203
5628cb3ee     202
             ...
d9fcbe12c       1
dd4a96d16       1
5cc5a5a19       1
00dba5599       1
b472193a9       1
Name: composto, Length: 3289, dtype: int64
```

Para plotar novo grafico, se escolheu tratar cinco elementos com maiores frequências;

```
cod_compostos = dados['composto'].value_counts().index[0:5]
```

```
cod_compostos
```

```
Index(['cacb2b860', '87d714366', '9f80f3f77', '8b87a7a83', '5628cb3ee'], dtype='objec
```

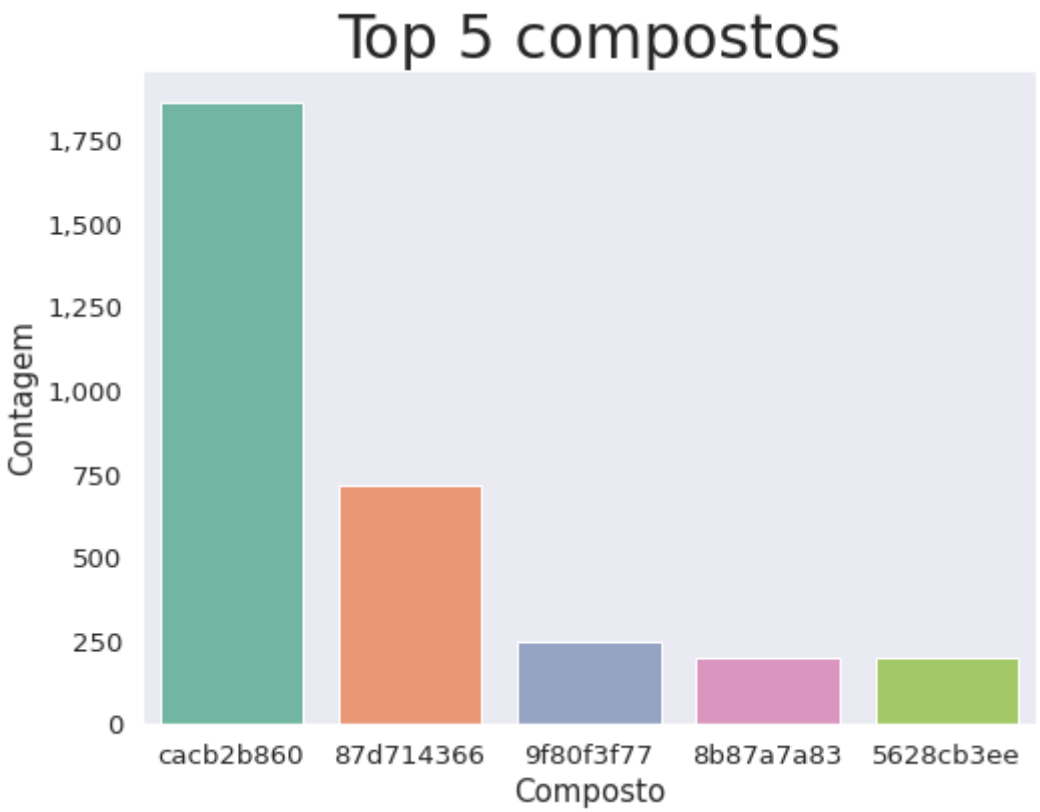
```
dados.query('composto in @cod_compostos')
```

	id	tratamento	tempo	dose	composto	g-0	g-1	g-2	
7	id_0020d0484	com_droga	48	D1	8b87a7a83	0.2711	0.5133	-0.1327	2.
16	id_002fb9c19	com_droga	48	D1	87d714366	8.7380	0.1914	2.4380	-0.
25	id_0054388ec	com_controle	48	D1	cacb2b860	-0.6696	-0.2718	-1.2230	-0.
38	id_0079af0fb	com_controle	24	D1	cacb2b860	-0.1636	-1.8230	-0.5211	0.
40	id_007bfbb91	com_controle	24	D2	cacb2b860	-1.3200	-1.7340	-0.0741	1.
...	...	...	...	...	...	...	...	...	
23793	id_ffd26f361	com_controle	48	D2	cacb2b860	0.6008	0.2781	-0.3319	-0.
23802	id_fff3976bd	com_droga	24	D1	87d714366	3.2890	2.1270	0.9770	2.
23805	id_fff6df1c5	com_droga	48	D2	5628cb3ee	1.7380	-1.2900	-0.4533	-1.
23811	id_fffc1c3f4	com_controle	48	D2	cacb2b860	0.3942	0.3756	0.3109	-0.
23812	id_fffcb9e7c	com_droga	24	D1	8b87a7a83	0.6660	0.2324	0.4392	0.

3235 rows × 877 columns

Grafico dos 'Top 5 Compostos'

```
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
sns.set_style('dark')
sns.set_context('notebook', font_scale=1.2)
plt.figure(figsize=(8, 6))
variavel_ax = sns.countplot(x = 'composto', data=dados.query('composto in @cod_compostos')
variavel_ax.set_title('Top 5 compostos', fontsize=30)
variavel_ax.set_xlabel('Composto', fontsize=15)
variavel_ax.set_ylabel('Contagem', fontsize=15)
variavel_ax.yaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.0f}'))
plt.show()
```



Explorando dados

```
dados['g-0'].min()
```

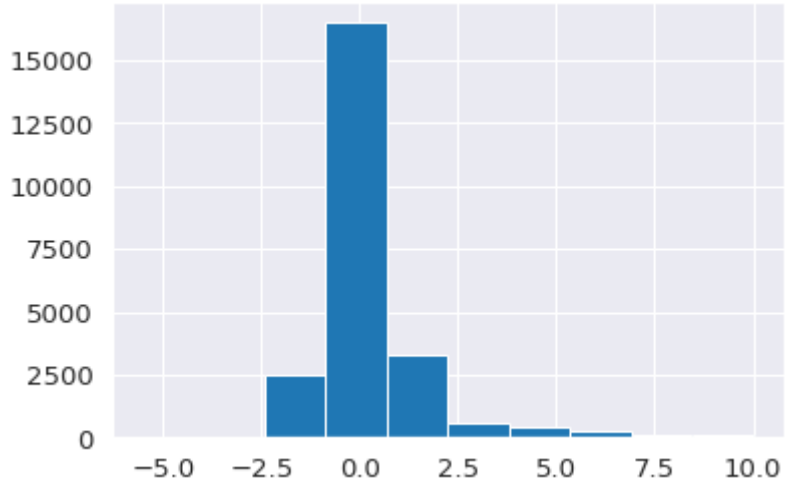
```
-5.513
```

```
dados['g-0'].max()
```

```
10.0
```

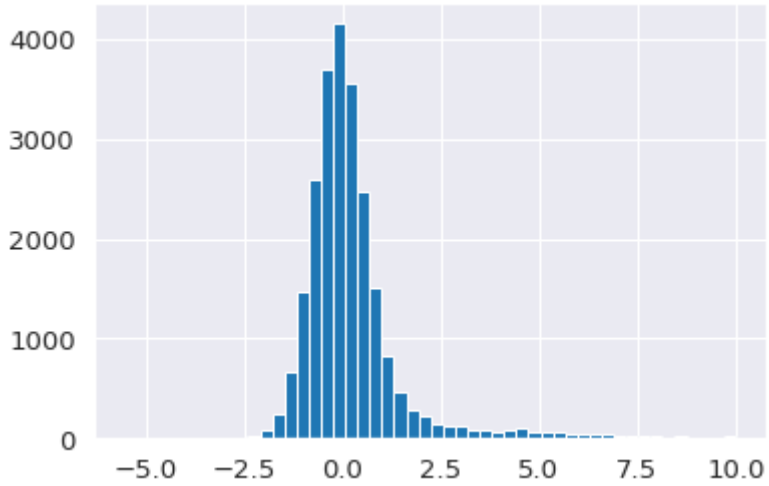
```
dados['g-0'].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4711f05f50>
```




```
dados['g-0'].hist(bins = 50)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f472151db90>
```

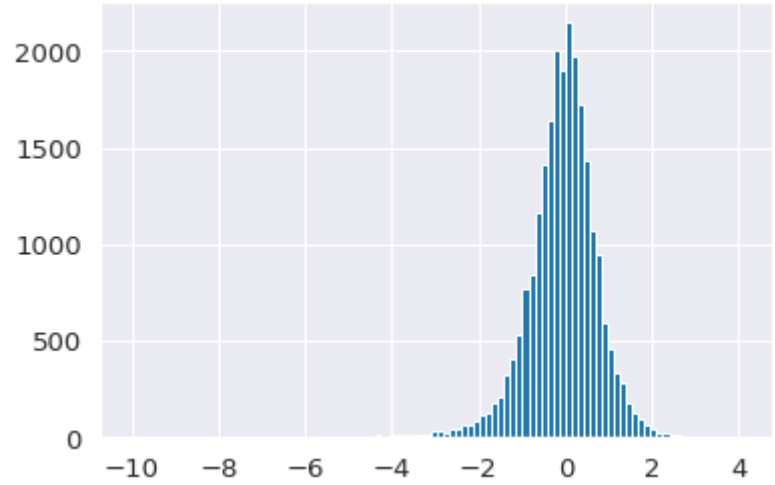



```
dados['g-0'].hist(bins = 100)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4711d8ba50>
```

```
dados['g-19'].hist(bins = 100)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4711bab350>
```




```
dados.describe()
```

	tempo	g-0	g-1	g-2	g-3	g
count	23814.000000	23814.000000	23814.000000	23814.000000	23814.000000	23814.000(
mean	48.020156	0.248366	-0.095684	0.152253	0.081971	0.057:
std	19.402807	1.393399	0.812363	1.035731	0.950012	1.032(
min	24.000000	-5.513000	-5.737000	-9.104000	-5.998000	-6.369(
25%	24.000000	-0.473075	-0.562200	-0.437750	-0.429575	-0.470(
50%	48.000000	-0.008850	-0.046600	0.075200	0.008050	-0.026(
75%	72.000000	0.525700	0.403075	0.663925	0.463400	0.465:
max	72.000000	10.000000	5.039000	8.257000	10.000000	10.000(

8 rows × 873 columns

```
dados.loc[:,'g-0':'g-771'].describe()
```

	g-0	g-1	g-2	g-3	g-4	
count	23814.000000	23814.000000	23814.000000	23814.000000	23814.000000	23814.000(
mean	0.248366	-0.095684	0.152253	0.081971	0.057347	-0.138
std	1.393399	0.812363	1.035731	0.950012	1.032091	1.179

```
dados.loc[:,'g-0':'g-771'].describe().T
```

	count	mean	std	min	25%	50%	75%	max
g-0	23814.0	0.248366	1.393399	-5.513	-0.473075	-0.00885	0.525700	10.000
g-1	23814.0	-0.095684	0.812363	-5.737	-0.562200	-0.04660	0.403075	5.039
g-2	23814.0	0.152253	1.035731	-9.104	-0.437750	0.07520	0.663925	8.257
g-3	23814.0	0.081971	0.950012	-5.998	-0.429575	0.00805	0.463400	10.000
g-4	23814.0	0.057347	1.032091	-6.369	-0.470925	-0.02690	0.465375	10.000
...	...	...	...	...	...	...	...	...
g-767	23814.0	-0.076251	1.115477	-10.000	-0.506200	0.00990	0.511175	6.317
g-768	23814.0	0.134162	0.951264	-4.269	-0.353100	0.00540	0.409075	10.000
g-769	23814.0	-0.128018	1.230636	-10.000	-0.544600	0.00060	0.498500	5.911
g-770	23814.0	-0.219210	1.326193	-10.000	-0.554400	0.02870	0.496400	10.000
g-771	23814.0	0.101524	1.417674	-10.000	-0.523800	-0.00650	0.536950	10.000

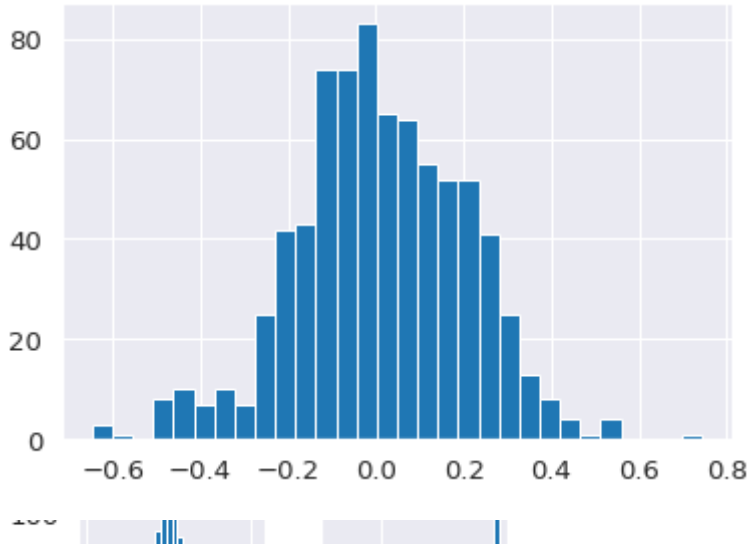
772 rows × 8 columns

```
dados.loc[:,'g-0':'g-771'].describe().T.hist(bins=30)
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f4711666b90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f4711e460d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f471162af50>],
```

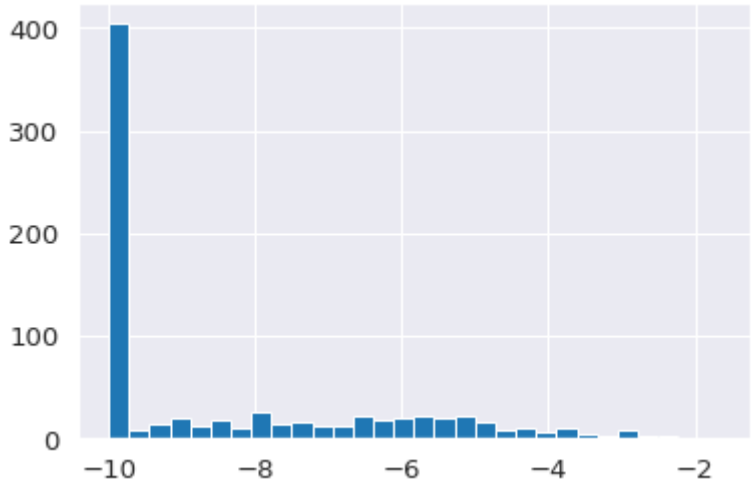
```
dados.loc[:,'g-0':'g-771'].describe().T['mean'].hist(bins=30)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4711aa6e90>
```




```
dados.loc[:,'g-0':'g-771'].describe().T['min'].hist(bins=30)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4711540810>
```

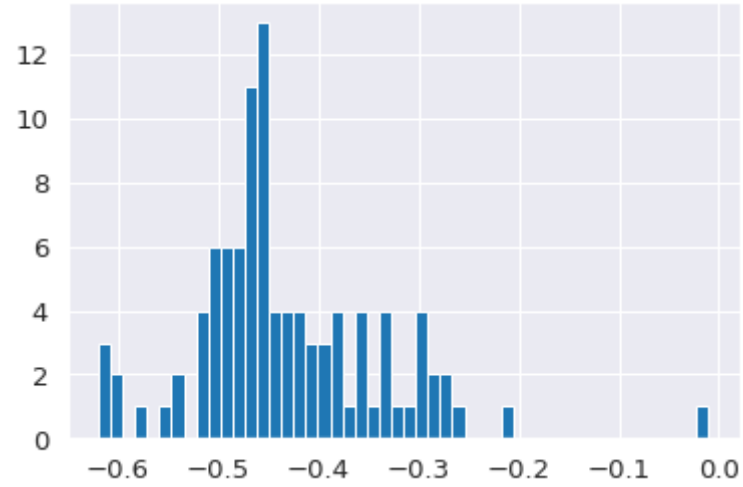



```
dados.loc[:,'g-0':'g-771'].describe().T['max'].hist(bins=30)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4711122e90>
```

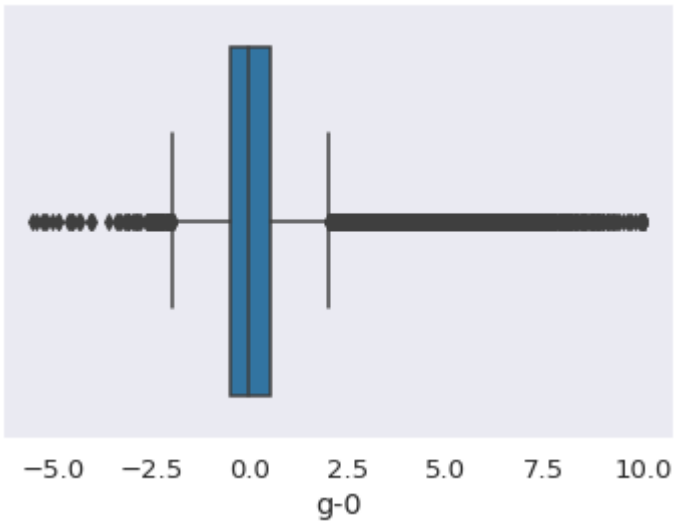
```
dados.loc[:,'c-0':'c-99'].describe().T['mean'].hist(bins=50)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4711e7fb10>
```



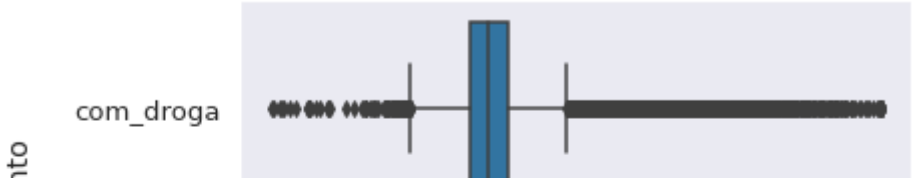

```
sns.boxplot(x='g-0', data=dados)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f471156ca10>
```



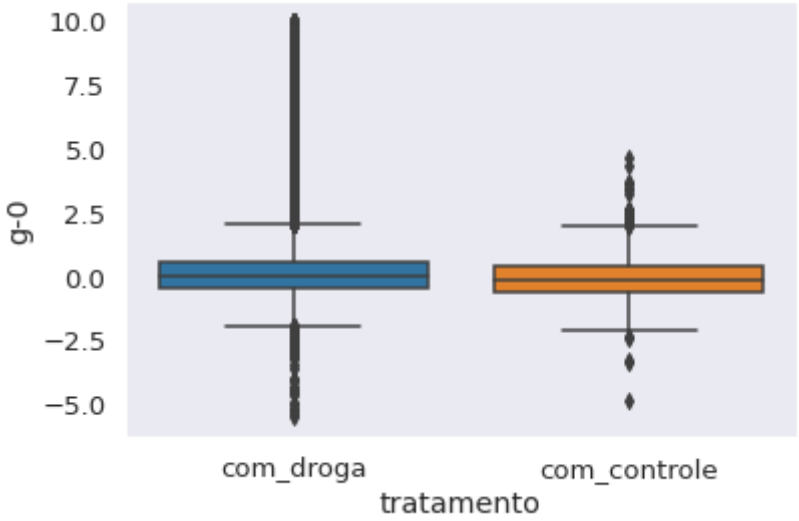

```
sns.boxplot(x='g-0',y='tratamento', data=dados)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f47112cad50>
```



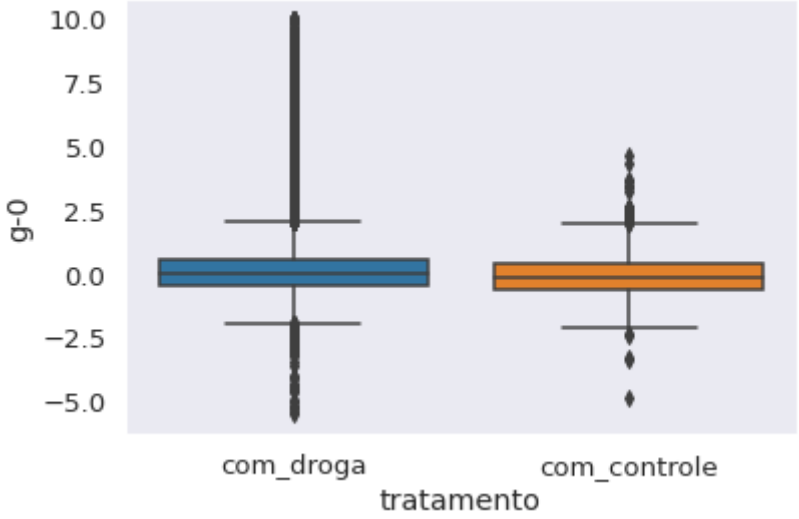

```
sns.boxplot(y='g-0', x='tratamento', data=dados)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4710f8e490>
```




```
import matplotlib.pyplot as plt
sns.boxplot(y='g-0', x='tratamento', data=dados)
plt.figure(figsize=(10, 8))
```

```
<Figure size 720x576 with 0 Axes>
```




```
<Figure size 720x576 with 0 Axes>
```

Sumário da etapa 2:

Estudou--se alterar nome de colunnas; com biblioteca 'Seaborn' criação de novos gráficos tipo histograma; aplicação da função 'loc'; comparação das informações plotadas em gráficos

'boxplot' versus 'histograma', e reflexões sobre a utilização de compostos (drogas) na pesquisa,

Clique duas vezes (ou pressione "Enter") para editar

Etapa 3 - Correlações, Causalidades e Relações entre genes

```
dados.head()
```

	id	tratamento	tempo	dose	composto	g-0	g-1	g-2	g-3
0	id_000644bb2	com_droga	24	D1	b68db1d53	1.0620	0.5577	-0.2479	-0.6208
1	id_000779bfc	com_droga	72	D1	df89a8e5a	0.0743	0.4087	0.2991	0.0604
2	id_000a6266a	com_droga	48	D1	18bb41b2c	0.6280	0.5817	1.5540	-0.0764
3	id_0015fd391	com_droga	48	D1	8c7f86626	-0.5138	-0.2491	-0.2656	0.5288
4	id_001626bd3	com_droga	72	D2	7cbed3131	-0.3254	-0.4009	0.9700	0.6919

5 rows × 877 columns

Correlacionando algumas categorias de dados, como por exemplo através das frequências, proporcionalidade, ...iniciar construindo tabela de frequência - pandas

```
pd.crosstab(dados['dose'], dados['tempo'])
```

tempo	24	48	72
dose			
D1	3886	4354	3907
D2	3886	3896	3885

```
pd.crosstab([dados['dose'], dados['tempo']], dados['tratamento'])
```

	tratamento	com_controle	com_droga
dose	tempo		
D1	24	301	3585
	48	343	4011

```
pd.crosstab([dados['dose'], dados['tempo']], dados['tratamento'],normalize=True)
```

	tratamento	com_controle	com_droga
dose	tempo		
D1	24	0.012640	0.150542
	48	0.014403	0.168430
	72	0.012892	0.151172
D2	24	0.012808	0.150374
	48	0.012808	0.150794
	72	0.012808	0.150332

```
pd.crosstab([dados['dose'], dados['tempo']], dados['tratamento'],normalize='index')
```

	tratamento	com_controle	com_droga
dose	tempo		
D1	24	0.077458	0.922542
	48	0.078778	0.921222
	72	0.078577	0.921423
D2	24	0.078487	0.921513
	48	0.078285	0.921715
	72	0.078507	0.921493

```
pd.crosstab([dados['dose'], dados['tempo']], dados['tratamento'], values=dados['g-0'], agg
```

	tratamento	com_controle	com_droga

Agora, explorar uma possivel relação entre os experimentos g-0 e g-3

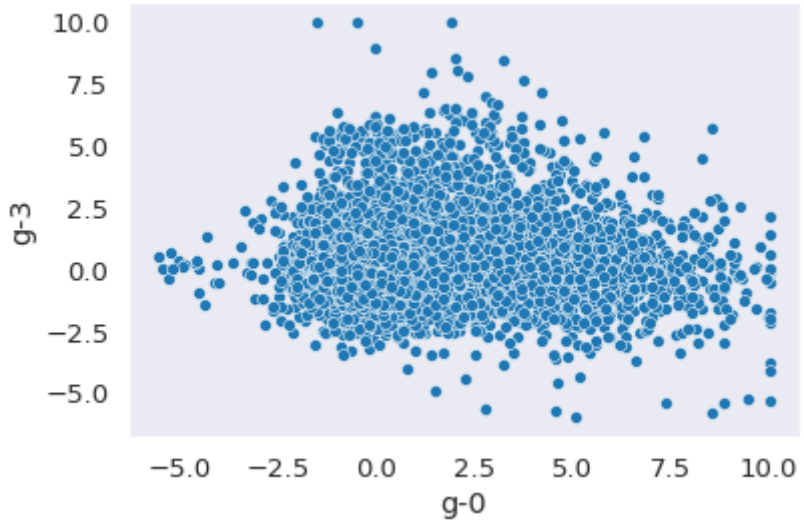
```
dados[['g-0','g-3']]
```

	g-0	g-3
0	1.0620	-0.6208
1	0.0743	0.0604
2	0.6280	-0.0764
3	-0.5138	0.5288
4	-0.3254	0.6919
...	...	...
23809	0.1394	-0.5080
23810	-1.3260	0.9905
23811	0.3942	-0.7389
23812	0.6660	0.2044
23813	-0.8598	0.7952

23814 rows × 2 columns

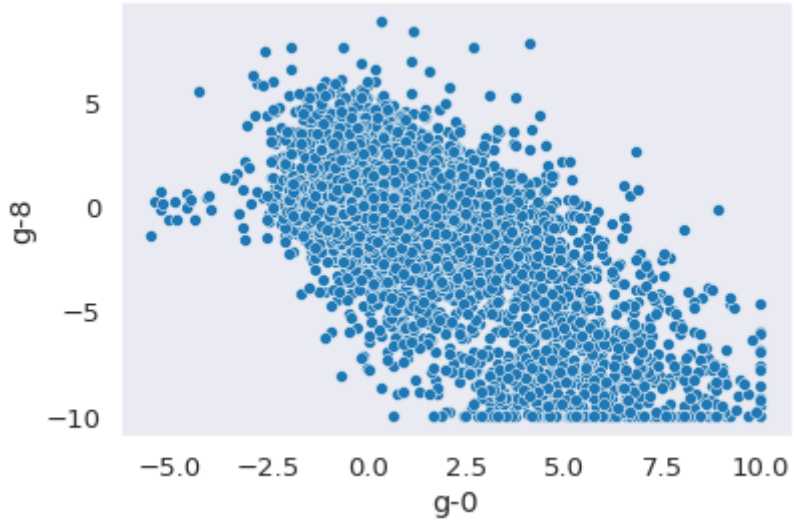
```
sns.scatterplot('g-0', 'g-3', data=dados)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f4710f50950>
```



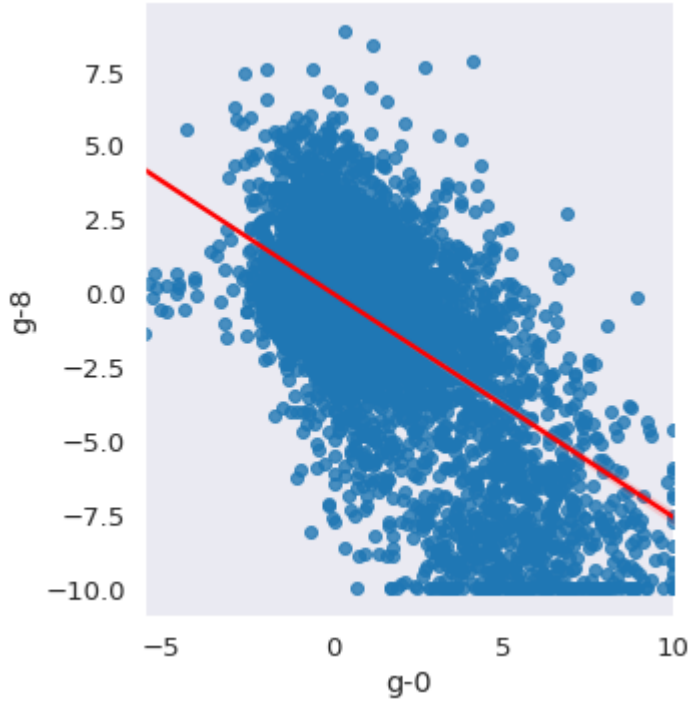

```
sns.scatterplot('g-0', 'g-8', data=dados)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f4710fe1950>
```




```
sns.lmplot('g-0', 'g-8', data=dados, line_kws={'color':'red'})
```

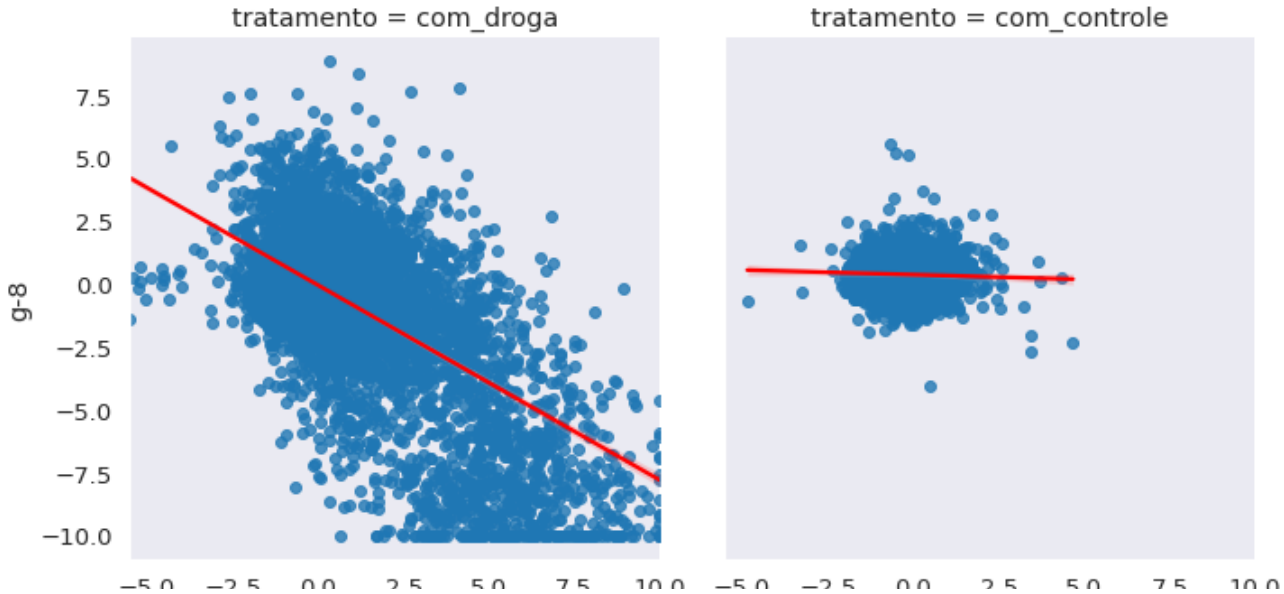
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<seaborn.axisgrid.FacetGrid at 0x7f470f6b7b50>
```



Ampliando a exploração de dados que possam identificar outras relações.

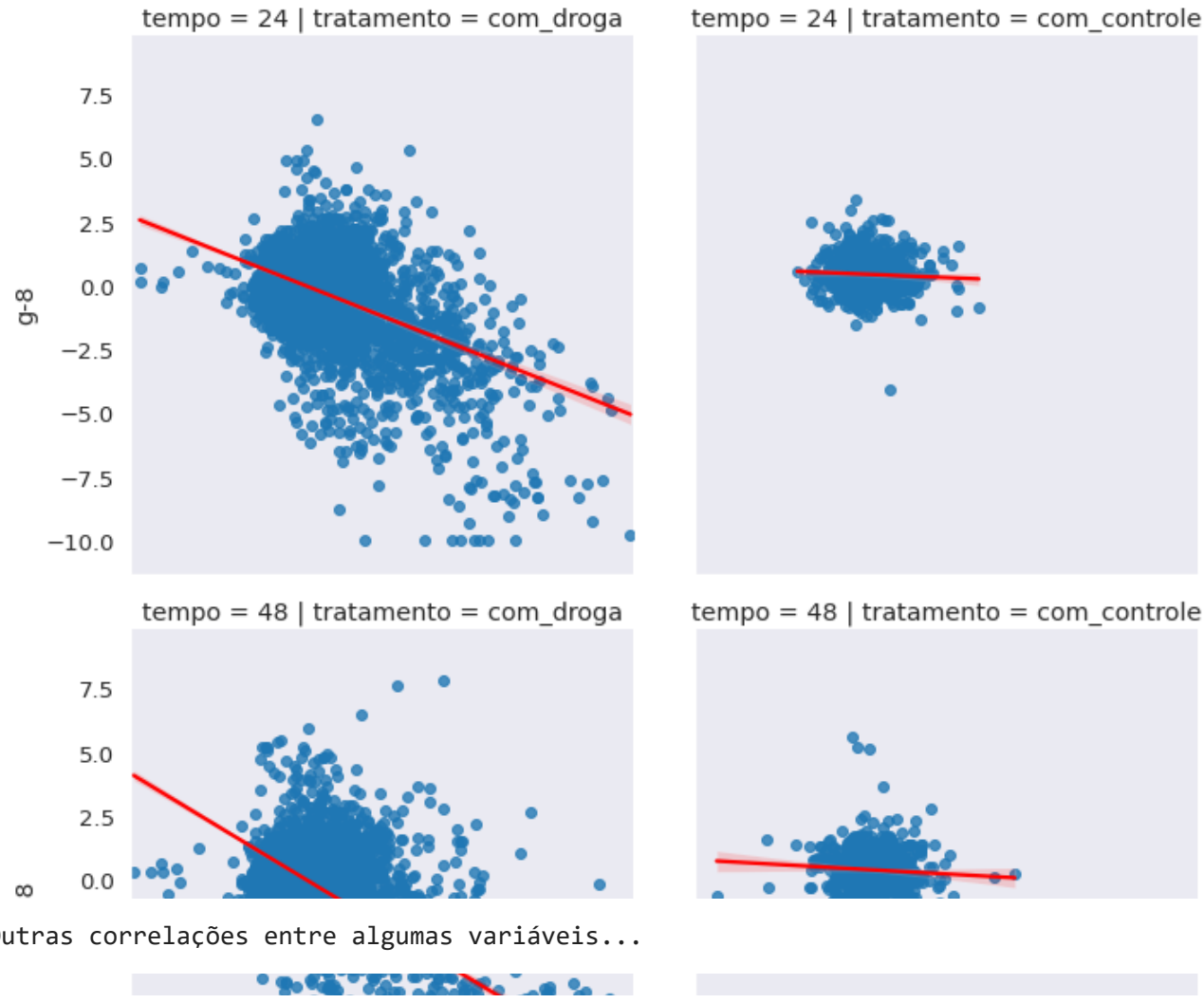
```
sns.lmplot('g-0', 'g-8', data=dados, line_kws={'color':'red'},col = 'tratamento')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<seaborn.axisgrid.FacetGrid at 0x7f4706e2f1d0>
```




```
sns.lmplot('g-0', 'g-8', data=dados, line_kws={'color':'red'},col = 'tratamento', row='tem
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<seaborn.axisgrid.FacetGrid at 0x7f4706dcdd10>
```



Outras correlações entre algumas variáveis...

```
dados.loc[:,'g-0':'g-771']
```

```
dados.loc[:,'g-0':'g-771'].corr()
```

	g-0	g-1	g-2	g-3	g-4	g-5	g-6	g-7
g-0	1.000000	0.086032	0.176188	0.011266	0.403153	-0.165478	0.238348	-0.37445
g-1	0.086032	1.000000	-0.054518	-0.027855	0.193647	-0.151590	-0.122541	0.14453
g-2	0.176188	-0.054518	1.000000	0.042633	0.013968	0.018516	0.015190	-0.14420
g-3	0.011266	-0.027855	0.042633	1.000000	-0.033039	0.042231	-0.043177	-0.17569
g-4	0.403153	0.193647	0.013968	-0.033039	1.000000	-0.057449	0.206405	-0.03274
...	...	...	...	...	...	...	...	..
g-767	-0.052622	-0.003378	-0.053149	-0.179370	0.124425	-0.081016	-0.000886	0.13733
g-768	0.300241	-0.030756	0.115415	0.028452	0.115579	-0.139000	-0.005791	-0.41943
g-769	-0.127375	-0.006866	0.014489	-0.206077	-0.172727	-0.068537	-0.278587	-0.08255
g-770	-0.514201	-0.095108	-0.078661	-0.002142	-0.363673	0.144631	-0.143428	0.40245
g-771	0.405908	-0.032233	0.103985	0.058219	0.250713	-0.093237	0.075019	-0.44130

772 rows × 772 columns

Notas quanto às correlações: Relações próximas de (-1) e (+1) são fortes, são proporcionalmente correlacionados, uma variável afeta diretamentea outra variável. Qdo relações estão próximas de zero, são relações fracas, uma variável não afeta a outra. Valores negativos ou positivos vão se refletir na curva gráfica das expressões gênicas avaliadas. Correlação não necessariamente implica em causalidade.

Calculando a correlação, gerando a diagonal da matriz de correlação (Numpy) - biblioteca Seaborn - https://seaborn.pydata.org/examples/many_pairwise_correlations.html

Clique duas vezes (ou pressione "Enter") para editar

```
corr = dados.loc[:,'g-0':'g-50'].corr()
```

```
import numpy as np
import matplotlib.pyplot as plt

# Compute the correlation matrix
corr = dados.loc[:,'g-0':'g-50'].corr()
```

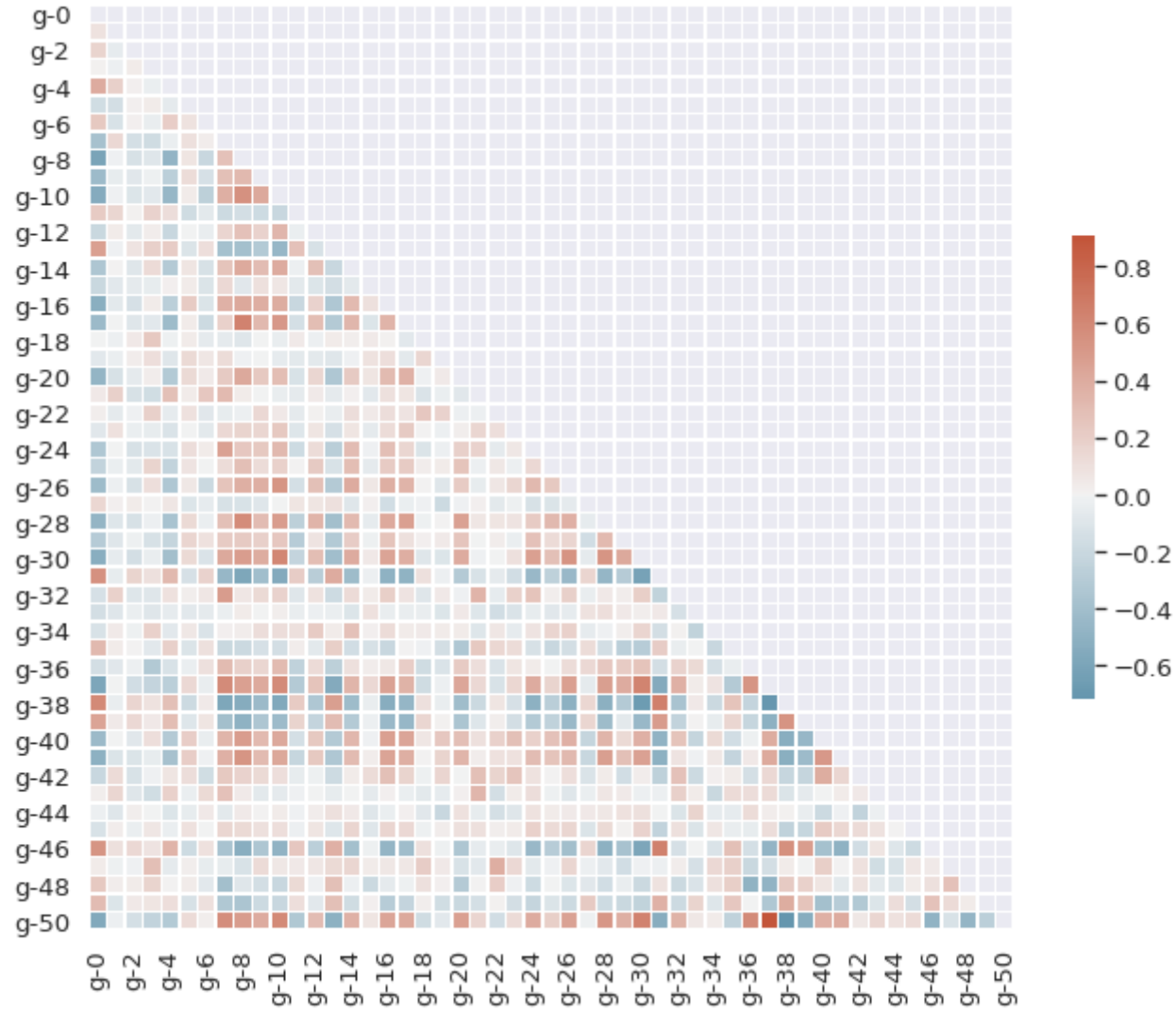
```
# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink":.5})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4706912110>
```



Clique duas vezes (ou pressione "Enter") para editar

Clique duas vezes (ou pressione "Enter") para editar

```
corr_celular = dados.loc[:,'c-0':'c-50'].corr()
```

Clique duas vezes (ou pressione "Enter") para editar

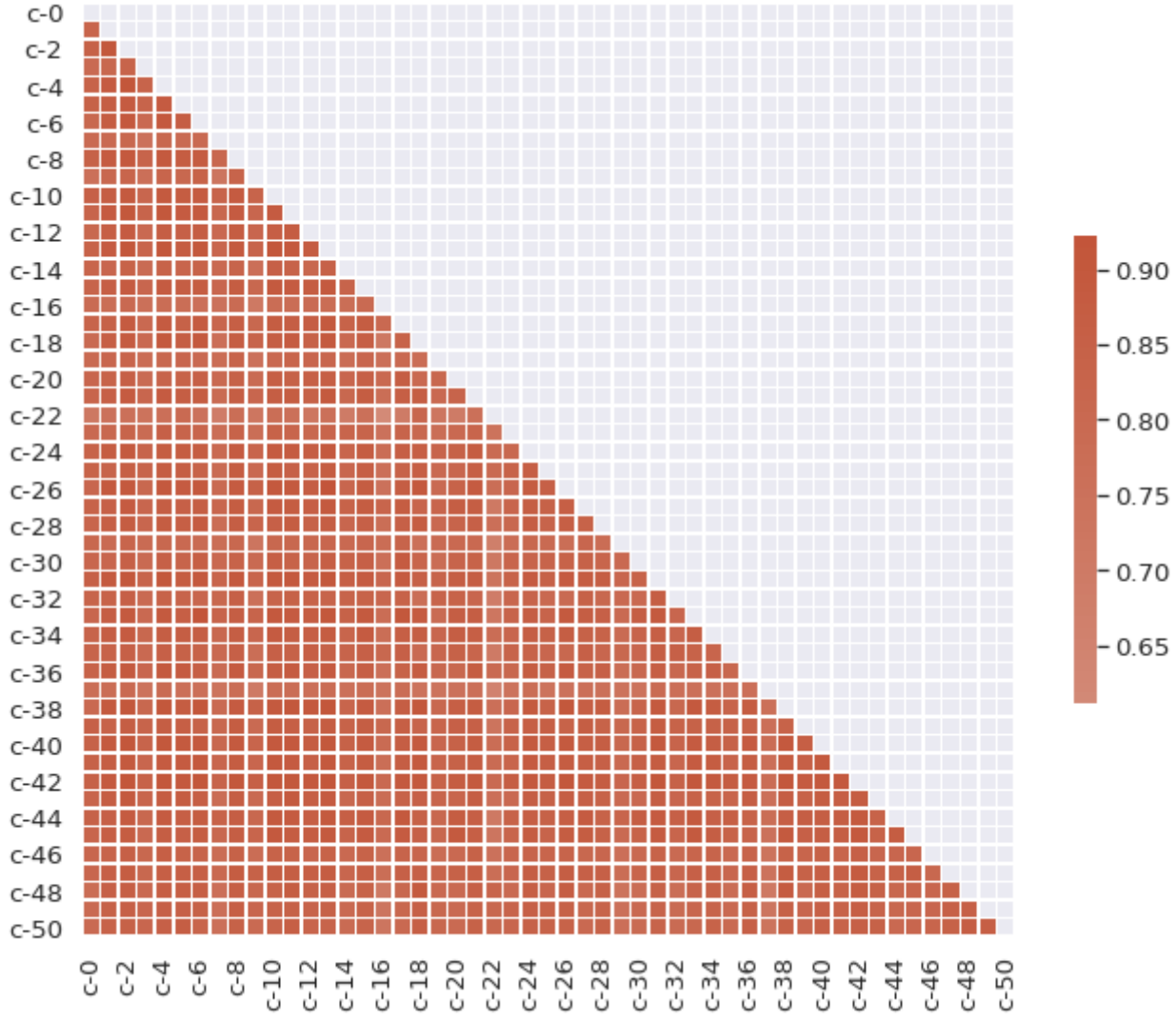
```
# Compute the correlation matrix
import matplotlib.pyplot as plt
# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr_celular, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr_celular, mask=mask, cmap=cmap, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f47067aee50>
```



Clique duas vezes (ou pressione "Enter") para editar

Clique duas vezes (ou pressione "Enter") para editar

Sumário da etapa 3

Estudo das correlações; plotagem da curva (reta) na relação entre duas variaveis; Calcular correlações; Opções na formulação e como analisar correlações.

Clique duas vezes (ou pressione "Enter") para editar

Etapa 4 - Merge de Dados e Análise de Resultados

Nesta etapa, se processou a importação arquivo .csv com resultados dos experimentos para começar a conhecer os mecanismos de ação como ativação ou inibição de algum 'gen', fazer merge das tabelas, e novas análises

Clique duas vezes (ou pressione "Enter") para editar

```
import pandas as pd
dados_resultados = pd.read_csv('https://github.com/alura-cursos/imersaodados3/blob/main/da
dados_resultados.head()
```

	id	5-alpha_reductase_inhibitor	11-beta-hsd1_inhibitor	acat_inhibitor	acetylchc
0	id_000644bb2	0	0	0	
1	id_000779bfc	0	0	0	
2	id_000a6266a	0	0	0	
3	id_0015fd391	0	0	0	
4	id_001626bd3	0	0	0	

5 rows × 207 columns

```
dados_resultados['acat_inhibitor'].unique()
```

```
array([0, 1])
```

```
dados_resultados['acetylcholine_receptor_agonist'].unique()
```

```
array([0, 1])
```

Para se conhecer quantas vezes um determinado mecanismo de ação foi ativado (somar coluna)

```
contagem_moa = dados_resultados.select_dtypes('int64').sum()
```

```
contagem_moa = dados_resultados.select_dtypes('int64').sum()
contagem_moa
```

```
5-alpha_reductase_inhibitor              17
11-beta-hsd1_inhibitor                   18
acat_inhibitor                           24
acetylcholine_receptor_agonist          190
acetylcholine_receptor_antagonist       301
                                       ...
ubiquitin_specific_protease_inhibitor     6
vegfr_inhibitor                         170
vitamin_b                                26
vitamin_d_receptor_agonist               39
wnt_inhibitor                            30
Length: 206, dtype: int64
```

```
contagem_moa = dados_resultados.select_dtypes('int64').sum().sort_values(ascending=False)
contagem_moa
```

```
nfkb_inhibitor                            832
proteasome_inhibitor                      726
cyclooxygenase_inhibitor                  435
dopamine_receptor_antagonist              424
serotonin_receptor_antagonist             404
                                         ...
protein_phosphatase_inhibitor               6
autotaxin_inhibitor                         6
diuretic                                    6
erbb2_inhibitor                             1
atp-sensitive_potassium_channel_antagonist  1
Length: 206, dtype: int64
```

```
dados_resultados.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23814 entries, 0 to 23813
Columns: 207 entries, id to wnt_inhibitor
dtypes: int64(206), object(1)
memory usage: 37.6+ MB
```

```
contagem_moa = dados_resultados.drop('id', axis=1).select_dtypes('int64').sum().sort_value
contagem_moa
```

```
nfkb_inhibitor                            832
proteasome_inhibitor                      726
cyclooxygenase_inhibitor                  435
dopamine_receptor_antagonist              424
serotonin_receptor_antagonist             404
                                         ...
protein_phosphatase_inhibitor               6
autotaxin_inhibitor                         6
diuretic                                    6
erbb2_inhibitor                             1
atp-sensitive_potassium_channel_antagonist  1
Length: 206, dtype: int64
```

Foco neste estágio é o de identificar quantas vezes um determinado experimento (linha) foi ativada e de que forma.

```
dados_resultados.sum()
```

```
id                                        id_000644bb2id_000779bfcid_000a6266aid_0015f
5-alpha_reductase_inhibitor
11-beta-hsd1_inhibitor
acat_inhibitor
acetylcholine_receptor_agonist
                                                                  ...
ubiquitin_specific_protease_inhibitor
vegfr_inhibitor
vitamin_b
vitamin_d_receptor_agonist
wnt_inhibitor
Length: 207, dtype: object
```

```
dados_resultados.head()
```

	id	5-alpha_reductase_inhibitor	11-beta-hsd1_inhibitor	acat_inhibitor	acetylchc
0	id_000644bb2	0	0	0	
1	id_000779bfc	0	0	0	
2	id_000a6266a	0	0	0	
3	id_0015fd391	0	0	0	
4	id_001626bd3	0	0	0	

5 rows × 207 columns

```
dados_resultados.drop('id', axis=1).sum (axis=1).sort_values(ascending = False)
```

```
21197    7
4849     7
19186    7
14316    7
20584    7
        ..
6862     0
6861     0
11997    0
6859     0
23813    0
Length: 23814, dtype: int64
```

```
dados_resultados.drop('id', axis=1).sum (axis=1)
```

```
0        1
1        0
2        3
3        0
4        1
        ..
23809    1
23810    1
23811    0
23812    1
23813    0
Length: 23814, dtype: int64
```

Pergunta-se algum controle dos experimentos, recebeu alguma ativação? A resposta esperada é nula! Vejamos! Tem-se duas bases de dados (BD): dados, dados_resultados.

```
dados.head()
```

	id	tratamento	tempo	dose	composto	g-0	g-1	g-2	g-3
0	id_000644bb2	com_droga	24	D1	b68db1d53	1.0620	0.5577	-0.2479	-0.6208
1	id_000779bfc	com_droga	72	D1	df89a8e5a	0.0743	0.4087	0.2991	0.0604
2	id_000a6266a	com_droga	48	D1	18bb41b2c	0.6280	0.5817	1.5540	-0.0764
3	id_0015fd391	com_droga	48	D1	8c7f86626	-0.5138	-0.2491	-0.2656	0.5288
4	id_001626bd3	com_droga	72	D2	7cbed3131	-0.3254	-0.4009	0.9700	0.6919

5 rows × 877 columns

```
dados_resultados.head()
```

	id	5-alpha_reductase_inhibitor	11-beta-hsd1_inhibitor	acat_inhibitor	acetylchc
0	id_000644bb2	0	0	0	
1	id_000779bfc	0	0	0	
2	id_000a6266a	0	0	0	
3	id_0015fd391	0	0	0	
4	id_001626bd3	0	0	0	

5 rows × 207 columns

criando a coluna n_moa

criando a coluna ativo_moa

```
dados_resultados['n moa'] = dados_resultados.drop('id', axis=1).sum (axis=1)
dados_resultados.head()
```

	id	5-alpha_reductase_inhibitor	11-beta-hsd1_inhibitor	acat_inhibitor	acetylchc
0	id_000644bb2	0	0	0	
1	id_000779bfc	0	0	0	
2	id_000a6266a	0	0	0	
3	id_0015fd391	0	0	0	
4	id_001626bd3	0	0	0	

5 rows × 208 columns

```
dados_resultados['n_moa'] = dados_resultados.drop('id', axis=1).sum(axis=1)
dados_resultados['ativo_moa'] = (dados_resultados['n_moa'] !=0)
dados_resultados.head()
```

	id	5-alpha_reductase_inhibitor	11-beta-hsd1_inhibitor	acat_inhibitor	acetylchc
0	id_000644bb2	0	0	0	
1	id_000779bfc	0	0	0	
2	id_000a6266a	0	0	0	
3	id_0015fd391	0	0	0	
4	id_001626bd3	0	0	0	

5 rows × 210 columns

Objetivo agora é criar uma lista das colunas de interesse com a função 'merge' p visualizar se 'controle' foi ativado nos experimentos...(levar nº moa ativos e se ele foi ativo ou não em relação ao 'controle'- lista das colunas.)

```
dados_combinados = pd.merge(dados, dados_resultados[['id','n_moa','ativo_moa']], on='id')
dados_combinados.head()
```

	id	tratamento	tempo	dose	composto	g-0	g-1	g-2	g-3
0	id_000644bb2	com_droga	24	D1	b68db1d53	1.0620	0.5577	-0.2479	-0.6208
1	id_000779bfc	com_droga	72	D1	df89a8e5a	0.0743	0.4087	0.2991	0.0604
2	id_000a6266a	com_droga	48	D1	18bb41b2c	0.6280	0.5817	1.5540	-0.0764
3	id_0015fd391	com_droga	48	D1	8c7f86626	-0.5138	-0.2491	-0.2656	0.5288
4	id_001626bd3	com_droga	72	D2	7cbed3131	-0.3254	-0.4009	0.9700	0.6919

5 rows × 879 columns

```
dados_combinados.query('tratamento =="com_controle"')['ativo_moa'].value_counts()
```

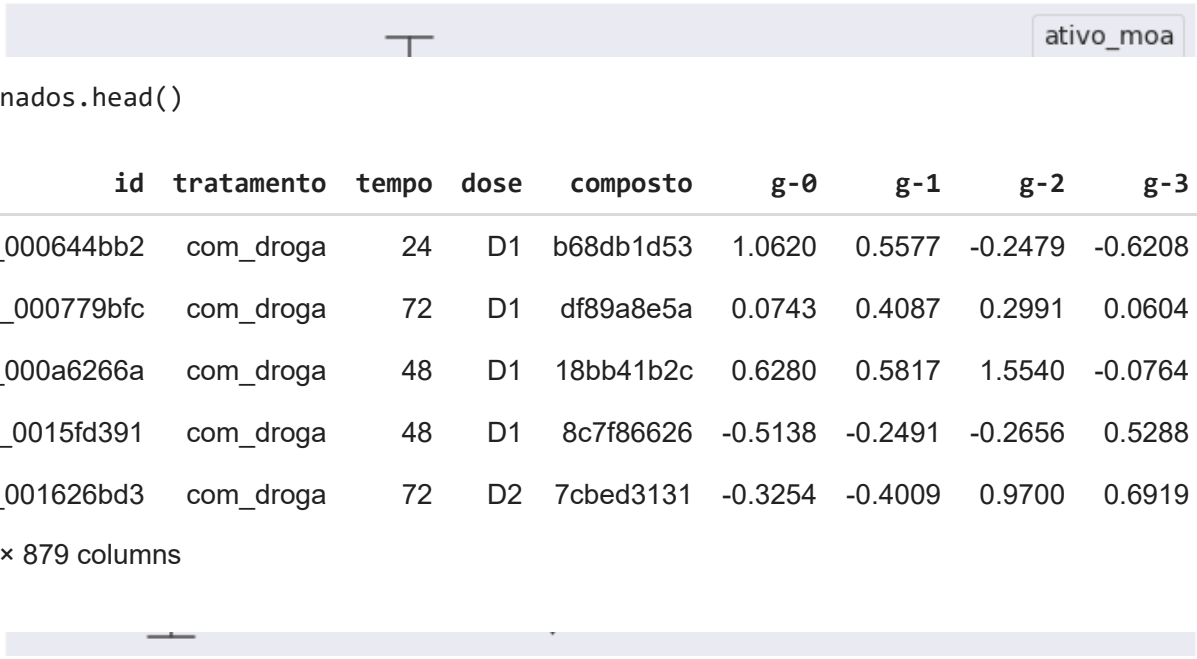
```
False    1866
Name: ativo_moa, dtype: int64
```

```
dados_combinados.query('tratamento =="com_droga"')['ativo_moa'].value_counts()
```

```
True     14447
False     7501
Name: ativo_moa, dtype: int64
```

```
composto_principal = dados_combinados['composto'].value_counts().index[:5]
plt.figure(figsize=(12,8))
sns.boxplot(data = dados_combinados.query('composto in @composto_principal'), y='g-0', x='
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f470658e510>
```




```
dados_combinados.head()
```

	id	tratamento	tempo	dose	composto	g-0	g-1	g-2	g-3
0	id_000644bb2	com_droga	24	D1	b68db1d53	1.0620	0.5577	-0.2479	-0.6208
1	id_000779bfc	com_droga	72	D1	df89a8e5a	0.0743	0.4087	0.2991	0.0604
2	id_000a6266a	com_droga	48	D1	18bb41b2c	0.6280	0.5817	1.5540	-0.0764
3	id_0015fd391	com_droga	48	D1	8c7f86626	-0.5138	-0.2491	-0.2656	0.5288
4	id_001626bd3	com_droga	72	D2	7cbed3131	-0.3254	-0.4009	0.9700	0.6919

5 rows × 879 columns

Sumário da etapa 4.

Procedemos a investigação de nova base de dados com os resultados dos mecanismos e dos experimentos; Identificou-se que um determinado composto pode ativar diferentes experimentos; Estruturamos DF para os novos resultados; Realizado merge das planilhas de dados iniciais e de resultados; Aprendemos a utiliz Utilizamos boxplot para comparar diferentes resultados

Contribuição do Gustavo Quadra — Hoje 06.05.21 às 17:29 Pessoal, gostaria de saber se estou entendendo a parte biológica da mesma forma que vocês.

id : identificador para um experimento G's : expressão gênica = Processo pelo qual a informação hereditária de um gene, forma uma proteína ou RNA C's : viabilidade celular = Analisar células de uma cultura celular, afim de avaliar sua atividade. composto / droga = Placebo (talvez) para casos com controle e remédios / substâncias diferentes para casos com droga MOA : Mecanismo de ação do Alvo : interação bioquímica entre uma droga utilizada e um alvo (enzima (terminados com -ase), proteína, etc)

Etapa 5 - Modelo de Machine Learning, Scikit-Learn e desafios envolvidos

Identificando-se possiveis tipos de problemas para se estabelecer o modelo de ML

1 - Com base no BD dados_combinados, fazer uma previsão se algum mecanismo de ação foi ativado ou não no experimento... (True ou False - classificação binária), e se ele ativou qualquer

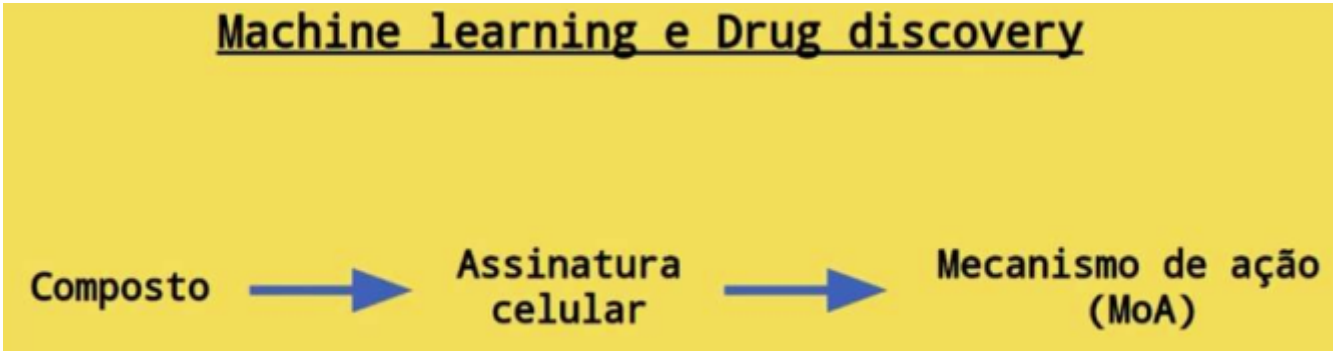
outro mecanismo de ação;

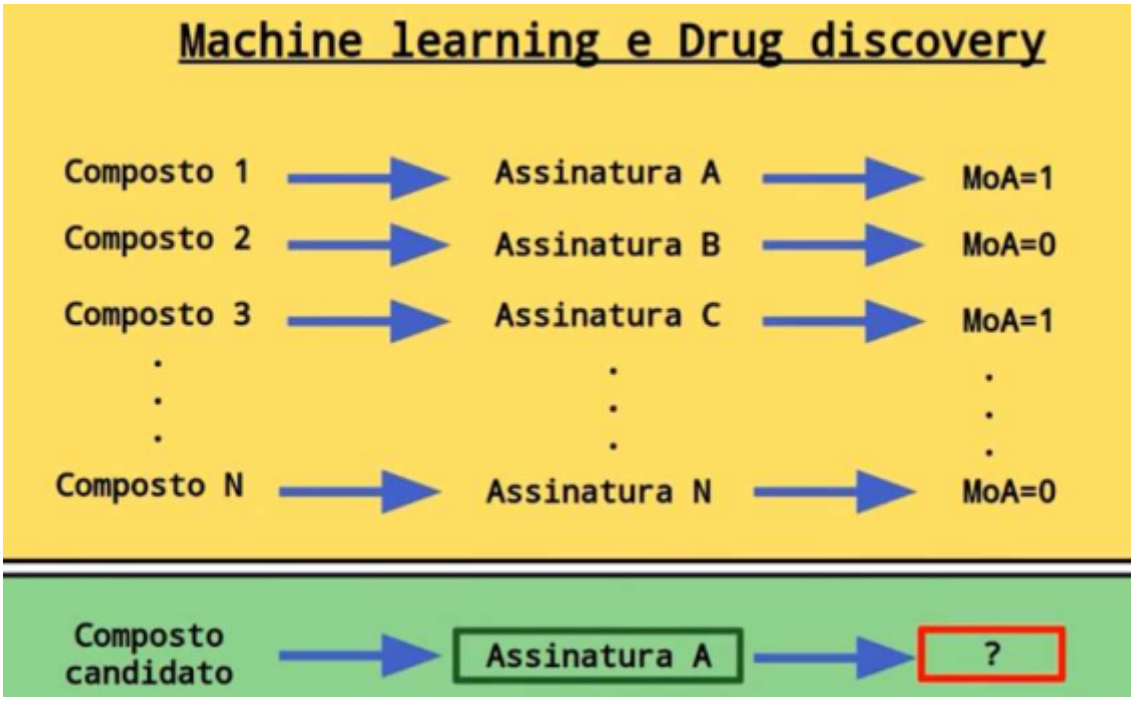
2 - qual ou quais composto foi ativado no conjunto de céluas ou gens - (cada linha é um composto), e se classificar os diversos compostos;

3 - Cada composto é uma classe específica - vejamos BD de resultados pois para cada 'id' - (cada linha é uma assinatura) pode-se se ter mais de um mecanismo de ação;

4 - ML poderá sinalizar se foi ativado o mecanismo de ação ou não e sinalizar também quais mecanismos e quantos foram ativados. (classificação multilabel).

5 - Conceitos aplicáveis estão resumidos nos slides a seguir:





Clique duas vezes (ou pressione "Enter") para editar

```
dados_combinados.head()
```

	id	tratamento	tempo	dose	composto	g-0	g-1	g-2	g-3
0	id_000644bb2	com_droga	24	D1	b68db1d53	1.0620	0.5577	-0.2479	-0.6208
1	id_000779bfc	com_droga	72	D1	df89a8e5a	0.0743	0.4087	0.2991	0.0604
2	id_000a6266a	com_droga	48	D1	18bb41b2c	0.6280	0.5817	1.5540	-0.0764
3	id_0015fd391	com_droga	48	D1	8c7f86626	-0.5138	-0.2491	-0.2656	0.5288
4	id_001626bd3	com_droga	72	D2	7cbed3131	-0.3254	-0.4009	0.9700	0.6919

Clique duas vezes (ou pressione "Enter") para editar

```
dados_resultados.head()
```

	id	5-alpha_reductase_inhibitor	11-beta-hsd1_inhibitor	acat_inhibitor	acetylchc
0	id_000644bb2	0	0	0	
1	id_000779bfc	0	0	0	
2	id_000a6266a	0	0	0	
3	id_0015fd391	0	0	0	
4	id_001626bd3	0	0	0	

5 rows × 210 columns

ML - Scikit Learn - https://scikit-learn.org/stable/

Selecionado inicialmente Regressão Logistica - modelo de classificação onde: x=assinatura celular e y=ativado ou não ativado

Para a fase de aprendizado do modelo, define-se uma porgentagem dos dados, e para a fase seguinte de testes, o test_size ficou definido 20%, para este projeto;

Para o aprendizado do modelo, e de forma a evitar que ele fique tendencioso, deve-se evitar usar os mesmos dados que ele usou para a faase de aprendizado;

x_treino, x_teste, y_treino, y_teste = train_test_split(x, y, test_size = .2)

No train_test_split(x, y) existem dataframes com os dados (4);

Assim, treina-se o modelo com parte dos dados, e depois se faz testes com o volume de dados restantes para avaliar a acuracidade do modelo.

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

```
x = dados_combinados.select_dtypes('float64')
y = dados_combinados['ativo_moa']
x_treino, x_teste, y_treino, y_teste = train_test_split(x, y, test_size = 0.2, stratify=y,

modelo_rlogistica = LogisticRegression(max_iter=1000, C=1)
modelo_rlogistica.fit(x_treino, y_treino)
modelo_rlogistica.score(x_teste, y_teste)
```

```
0.6353138778081041
```

Resultado acima significa quanto o modelo escolhido conseguiu acertar (acuracidade do modelo). Para efeito comparativo, foi escolhido o modelo Dummy, que sabidamente é menos eficiente.

Testando com o modelo de ML, Dummy - mais simplificado da RegressaoLogistica

```
from sklearn.dummy import DummyClassifier
from sklearn.metrics import accuracy_score

modelo_dummy = DummyClassifier('most_frequent')
modelo_dummy.fit(x_treino, y_treino)
previsao_dummy = modelo_dummy.predict(x_teste)
accuracy_score(y_teste, previsao_dummy)
```

```
0.6067604450976275
```

Confrontado os resultados com a proporcionalidade demonstrada no BD de resultados, vejamos:

```
dados_combinados['ativo_moa'].value_counts(normalize=True)
```

```
True     0.60666
False    0.39334
Name: ativo_moa, dtype: float64
```

Novo Teste com o Modelo de ML, Arvore de Decisão, para comparação

```
from sklearn.tree import DecisionTreeClassifier

x = dados_combinados.select_dtypes('float64')
y = dados_combinados['ativo_moa']
```

```
y = dados_combinados[ 'ativo_moa' ]
x_treino, x_teste, y_treino, y_teste = train_test_split(x, y, test_size = 0.2, stratify=y,

modelo_arvore = DecisionTreeClassifier(max_depth = 3)
modelo_arvore.fit(x_treino, y_treino)
modelo_arvore.score(x_teste, y_teste)
```

```
0.6105395758975436
```

```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import make_multilabel_classification
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

x = dados_combinados.select_dtypes('float64')
y = dados_combinados['ativo_moa']
x_treino, x_teste, y_treino, y_teste = train_test_split(x, y, test_size = 0.2, stratify=y,

sklearn.tree.plot_tree(decision_tree, * max_depth = None,
                       feature_names=x_treino.columns,
                       class_names=['Não Ativo', 'Ativo'],
                       label='all',
                       filled=True,
                       impurity=True,
                       node_ids=False,
                       proportion=False,
                       rotate='deprecated',
                       rounded=True,
                       precision=3,
                       ax=ax,
                       fontsize=10)
plt.show()
```

```
  File "<ipython-input-20-ee82a4277f6b>", line 11
    sklearn.tree.plot_tree(decision_tree, * max_depth = None,
                                          ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

```
from  sklearn import tree
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(15,10), facecolor='k')
tree.plot_tree(modelo_arvore,
               ax=ax,
               fontsize=10,
               rounded=True,
               filled=True,
               feature_names=x_treino.columns,
               class_names=['Não Ativo', 'Ativo'])
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-21-893280e3e7d6> in <module>()
      3
      4 fig, ax = plt.subplots(figsize=(15,10), facecolor='k')
----> 5 tree.plot_tree(modelo_arvore,
      6                ax=ax,
      7                fontsize=10,

NameError: name 'modelo_arvore' is not defined
```

SEARCH STACK OVERFLOW

```
from sklearn.tree import DecisionTreeClassifier

x = combinados.select_dtypes('float64')
y = combinados['ativo_moa']
x_treino, x_teste, y_treino, y_teste = train_test_split(x, y, test_size = .2, random_state
                                                        stratify=y)
teste=[]
treino=[]
```

```
treino=[]
for i in range(1,15):
  modelo_arvore = DecisionTreeClassifier(max_depth=i)
  modelo_arvore.fit(x_treino, y_treino)
  teste.append(modelo_arvore.score(x_teste, y_teste))
  treino.append(modelo_arvore.score(x_treino, y_treino))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-19-feb090b530eb> in <module>()
      2 from sklearn.tree import DecisionTreeClassifier
      3
----> 4 x = combinados.select_dtypes('float64')
      5 y = combinados['ativo_moa']
      6 x_treino, x_teste, y_treino, y_teste = train_test_split(x, y, test_size =
.2, random_state=376,

NameError: name 'combinados' is not defined
```

SEARCH STACK OVERFLOW

```
testes
```

```
treino
```

Plotar grafico com os valores de treino e testes p observar o aprimoramento do modelo de ML a medida que vai refazendo o seu treinamento...

```
from sklearn.tree import DecisionTreeClassifier

x = combinados.select_dtypes('float64')
y = combinados['ativo_moa']
x_treino, x_teste, y_treino, y_teste = train_test_split(x, y, test_size = .2, random_state
                                                        stratify=y)
teste=[]
treino=[]
for i in range(1,15):
    modelo_arvore = DecisionTreeClassifier(max_depth=i)
    modelo_arvore.fit(x_treino, y_treino)
    teste.append(modelo_arvore.score(x_teste, y_teste))
    treino.append(modelo_arvore.score(x_treino, y_treino))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-13-4a570ec56bab> in <module>()
      1 from sklearn.tree import DecisionTreeClassifier
      2
----> 3 x = combinados.select_dtypes('float64')


 2  random_state=376
```

Teste com o modelo RandomForest

```
from sklearn.ensemble import RandomForestClassifier


x = dados_combinados.drop(['id', 'n_moa', 'ativo_moa', 'composto'], axis=1)
x = pd.get_dummies(x, columns=['tratamento', 'dose', 'tempo'])
y = dados_combinados['ativo_moa']
x_treino, x_teste, y_treino, y_teste = train_test_split(x, y, test_size = 0.2, stratify=y,

modelo_randomforest = RandomForestClassifier()
modelo_randomforest.fit(x_treino, y_treino)
modelo_randomforest.score(x_teste, y_teste)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-22-b3efb64e84a4> in <module>()
      2
      3
----> 4 x = dados_combinados.drop(['id', 'n_moa', 'ativo_moa', 'composto'], axis=1)
      5 x = pd.get_dummies(x, columns=['tratamento', 'dose', 'tempo'])
      6 y = dados_combinados['ativo_moa']

NameError: name 'dados_combinados' is not defined
```

SEARCH STACK OVERFLOW

```
modelo_rlogistica = LogisticRegression(max_iter=1000, C=1)
modelo_rlogistica.fit(x_treino, y_treino)
modelo_rlogistica.score(x_teste, y_teste)
```

Sumário da etapa 5

Exercitado modelos de Machine Learning e Scikit-Learn; tipos de problemas que podem ser resolvidos com ML; classificação para rodar o modelo; árvore de decisão; steps para se

aprofundar e melhorar o modelo de árvore de decisão; reflexão, análise e interpretação dos

Clique duas vezes (ou pressione "Enter") para editar

Conclusão do projeto:

Conclusão prejudicada devido a mensagem de erro, em determinado ponto da etapa 5, que demandaria um tempo maior do que o disponível nesta data.

Para mim, esta é uma motivação a mais para continuidade dos estudos e resolução das questões em aberto nesta última etapa, em período pós ImersãoDadosAlura. Tenho muito a aprender e para tanto estarei me dedicando aos pontos, sites e referências recomendados pelos especialistas da Alura, que demontraram um alto nível de conhecimento em DataScience.

Agradecimentos

Agradeço sinceramente a Alura por me proporcionar uma experiência efetivamente incrível, em especial aos tutores mencionados na abertura.

Foram dias de aprendizado, reflexões, desafios e de despertar a motivação para a continuidade dos estudos e para explorar o universo de DataScience. Diversas lições aprendidas!

Agradeço à comunidade da Imersão Dados 3ª edição, presentes na plataforma Discord. Foi incrível também compartilhar o espaço(virtual) com vocês, e receber ajuda nos momentos de aperto. Meu muito obrigado!

Clique duas vezes (ou pressione "Enter") para editar

Clique duas vezes (ou pressione "Enter") para editar

Clique duas vezes (ou pressione "Enter") para editar

Bibliografia:

Post sobre pesquisa na indústria farmacêutica - acesso em 04.05.2021 - https://science.talknmb.com.br/pesquisa-clinica-industria-farmaceutica/

Calculo da correlação: https://seaborn.pydata.org/examples/many_pairwise_correlations.html

0s conclusão: 19:52