MBA em Ciência de Dados

Técnicas Avançadas de Captura e Tratamento de Dados

Módulo III - Aquisição e Transformação de Dados

Transformação de atributos

Material Produzido por Moacir Antonelli Ponti

CeMEAI - ICMC/USP São Carlos

Referência complementar

DIEZ, David M.; BARR, Christopher D.; CETINKAYA-RUNDEL, Mine. **OpenIntro statistics**. 3.ed. OpenIntro, 2015. Capítulo 1.

Distribuições de variáveis

Variáveis possuem distribuições diferentes de valores possíveis devido a sua natureza.

O senso comum vai assumir que temos uma das distribuições abaixo:

- uniforme
- normal

... e que correlações são sempre:

- lineares

Mas isso nem sempre acontece!

Pode ser útil transformar as variáveis para permitir melhor analisar os dados, inclusive suas correlações em particular com dados de natureza **exponencial**

In [1]:
```
# carregando as bibliotecas necessárias
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# carregando dados
data = pd.read_csv("./dados/municipios_mba.csv")
data.head()
```

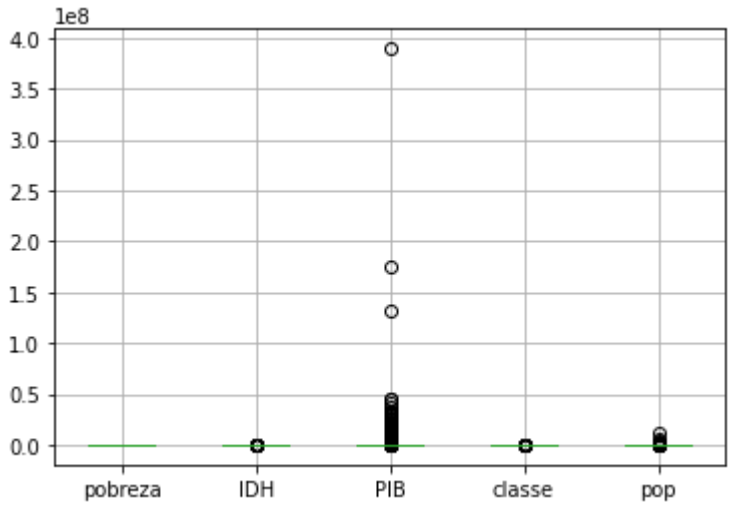
Out[1]:

	gid	UF	nome	Censo	PIB	pop	classe	desemprego	po
0	752	ACRE	Acrelândia	2010.0	151120.015625	12241	2	5.2	33
1	747	ACRE	Assis Brasil	2010.0	48347.300781	5662	1	5.0	32
2	748	ACRE	Brasiléia	2010.0	194979.828125	20238	1	3.0	31
3	754	ACRE	Bujari	2010.0	88708.031250	6772	2	4.8	33
4	751	ACRE	Capixaba	2010.0	89052.679688	9287	1	4.4	33

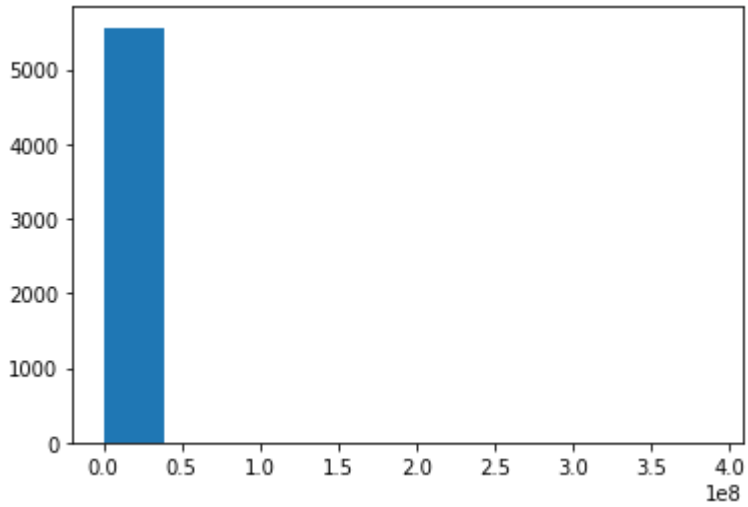
Vamos estudar as variáveis abaixo, exibindo seus gráficos de caixa.

In [2]:
```
attrs = ['pobreza', 'IDH', 'PIB', 'classe', 'pop']
attrs_cat = attrs + ['UF', 'nome']
data.boxplot(attrs)
```

Out[2]: <matplotlib.axes._subplots.AxesSubplot at 0x7efeec668730>



In [3]:
```
h = plt.hist(data['PIB'])
```

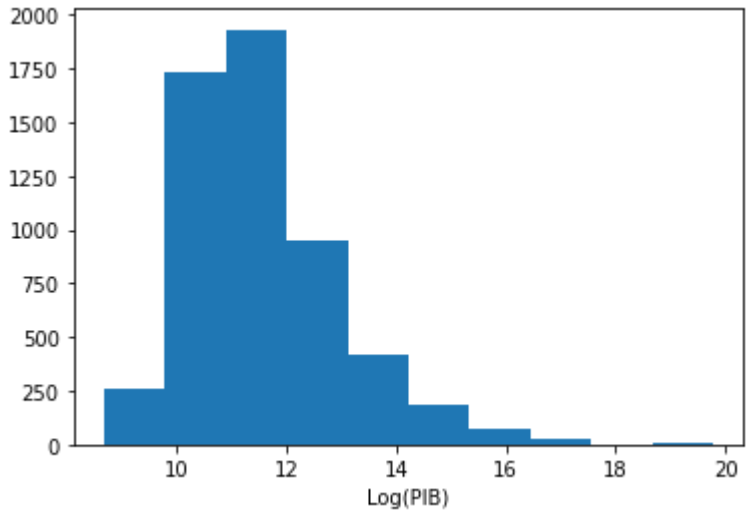


Parece difícil entender a variável PIB!

Talvez se aplicarmos uma transformação **logaritmica**!

In [4]:
```
x_PIB = np.array(data['PIB'])
h = plt.hist(np.log(x_PIB+1));
plt.xlabel('Log(PIB)')
```

Out[4]:
```
Text(0.5, 0, 'Log(PIB)')
```

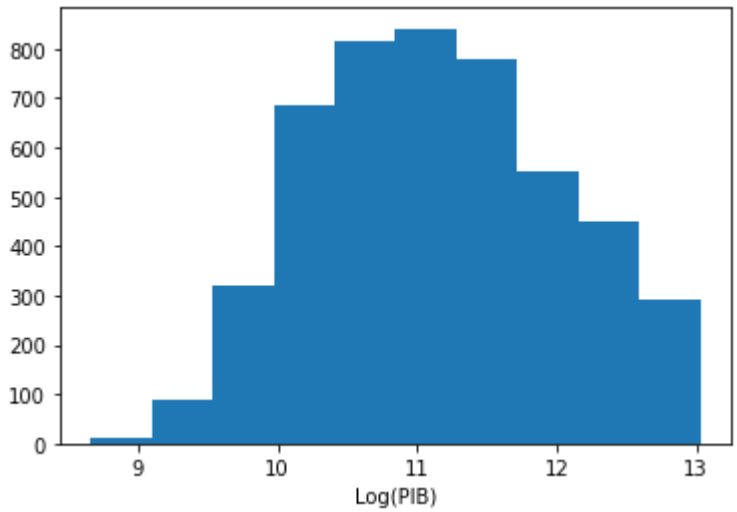


Além da transformação, pode ser útil também remover outliers antes!

In [5]:
```
def remove_outliers_IQR(df, attributes):

    dfn = df.copy()

    for var in attributes:
        # verifica se variável é numerica
        if np.issubdtype(df[var].dtype, np.number):
            Q1 = dfn[var].quantile(0.25)
            Q2 = dfn[var].quantile(0.50)
            Q3 = dfn[var].quantile(0.75)
            IQR = Q3 - Q1

            # apenas inliers segundo IQR
            dfn = dfn.loc[(df[var] >= Q1-(IQR*1.5)) & (df[v
ar] <= Q3+(IQR*1.5)),:]

    return dfn
```

In [6]:
```
data_out = data.copy()
data_out = remove_outliers_IQR(data_out, ['PIB'])

# vamos adicionar essa nova variavel a base
x_PIB = np.array(data_out['PIB'])
h = plt.hist(np.log(x_PIB+1));
plt.xlabel('Log(PIB)')
```

Out[6]: Text(0.5, 0, 'Log(PIB)')



Após limpeza de outliers e transformação os dados tem característica Normal e assim, modelos que consideram essa distribuição podem ser aplicados.

O log é o inverso da função exponencial e assim, estamos buscando linearizar os dados exponenciais.

In [7]:
```
data_out['Log_PIB'] = np.log(x_PIB+1)
attrs = attrs + ['Log_PIB']
data_out[attrs].corr()
```

Out[7]:

	pobreza	IDH	PIB	classe	pop	Log_PIB
pobreza	1.000000	-0.942245	-0.170224	-0.507724	0.300206	-0.185381
IDH	-0.942245	1.000000	0.141545	0.419101	-0.221749	0.154362
PIB	-0.170224	0.141545	1.000000	0.346233	0.666058	0.902932
classe	-0.507724	0.419101	0.346233	1.000000	-0.223733	0.366278
pop	0.300206	-0.221749	0.666058	-0.223733	1.000000	0.649933
Log_PIB	-0.185381	0.154362	0.902932	0.366278	0.649933	1.000000

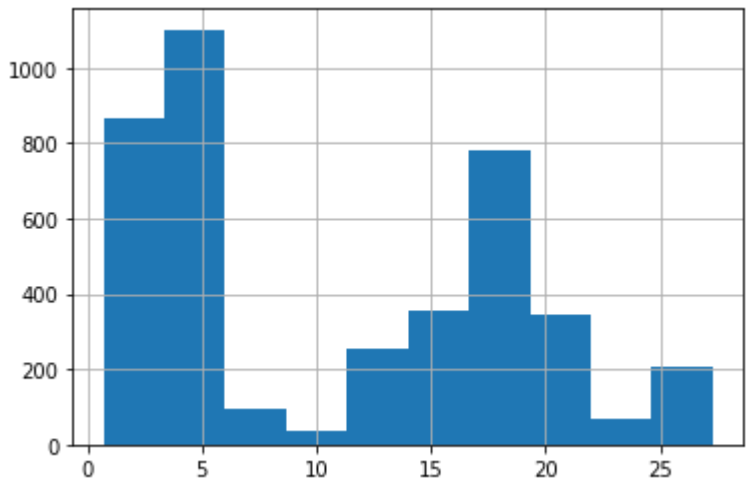
Observar que PIB e Log_PIB tem correlação 0.9

ou seja, ainda retemos muito da informação, apesar de transformada.

Agora vamos olhar para outras variáveis, que também não possuem distribuição bem comportada:

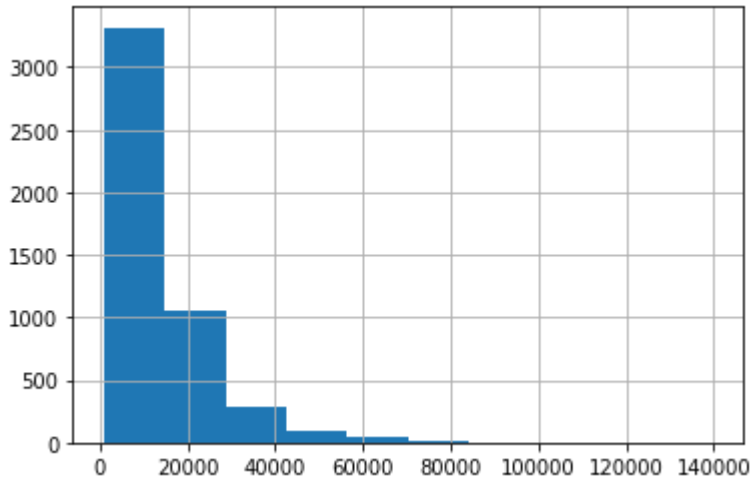
In [8]:
```
data_out['pobreza'].hist()
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7efee9f434c0>
```



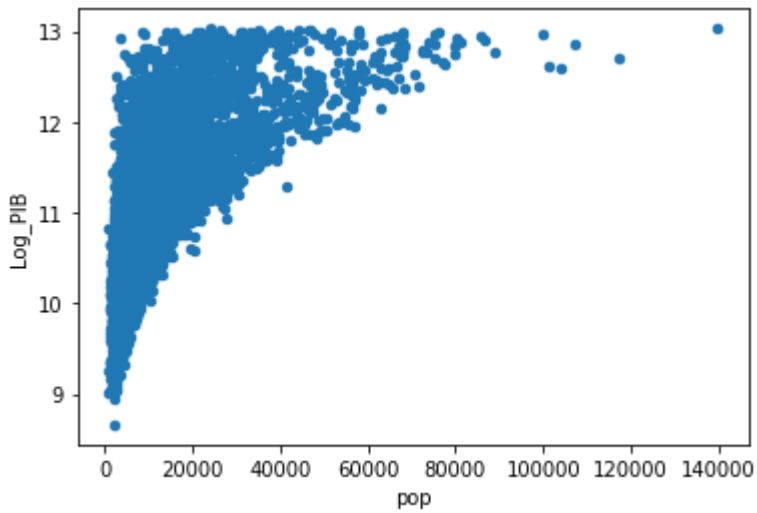
In [9]:
```
data_out['pop'].hist()
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7efee9eac430>
```



In [10]:
```
data_out.plot.scatter(x='pop', y='Log_PIB')
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7efee9ea2be0>
```

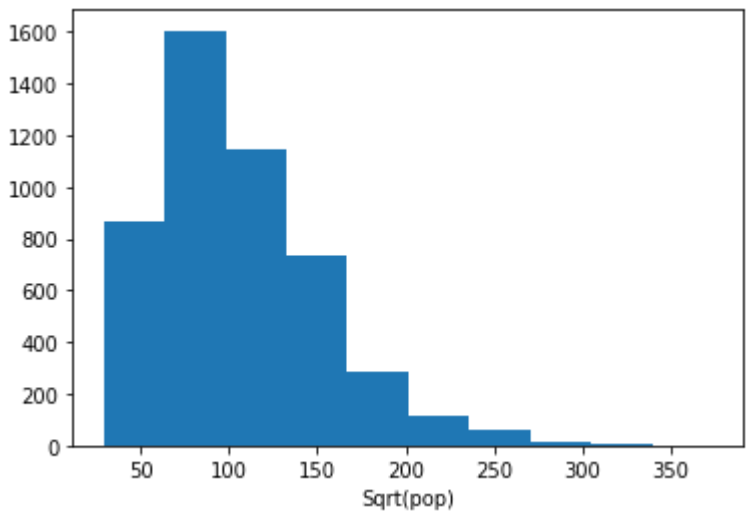


Outra função comumente utilizada (essa para linearizar dados de natureza quadrática), é a raiz quadrada.

In [11]:
```
x_pop = np.array(data_out['pop'])
h = plt.hist(np.sqrt(x_pop));
plt.xlabel('Sqrt(pop)')

data_out['Sqrt_pop'] = np.sqrt(x_pop)
attrs = attrs + ['Sqrt_pop']
data_out[attrs].corr()
```

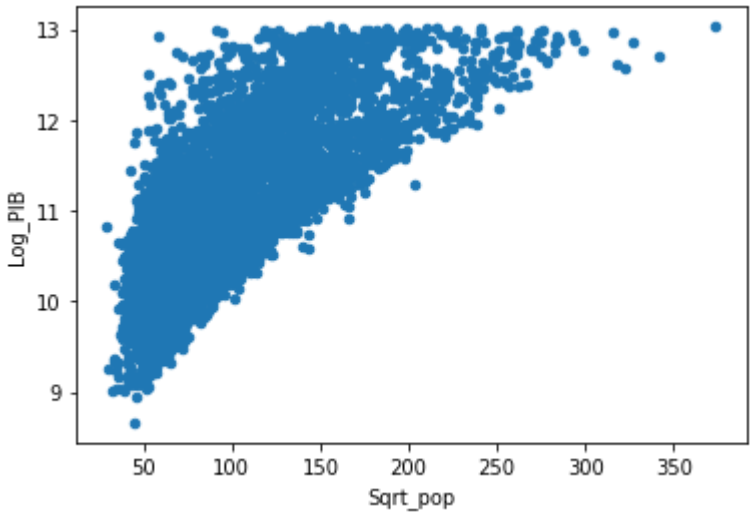
Out[11]:

	pobreza	IDH	PIB	classe	pop	Log_PIB	Sqrt_p
pobreza	1.000000	-0.942245	-0.170224	-0.507724	0.300206	-0.185381	0.3202
IDH	-0.942245	1.000000	0.141545	0.419101	-0.221749	0.154362	-0.2424
PIB	-0.170224	0.141545	1.000000	0.346233	0.666058	0.902932	0.6846
classe	-0.507724	0.419101	0.346233	1.000000	-0.223733	0.366278	-0.2467
pop	0.300206	-0.221749	0.666058	-0.223733	1.000000	0.649933	0.9664
Log_PIB	-0.185381	0.154362	0.902932	0.366278	0.649933	1.000000	0.7172
Sqrt_pop	0.320284	-0.242492	0.684638	-0.246793	0.966454	0.717255	1.0000



In [12]:
```
data_out.plot.scatter(x='Sqrt_pop', y='Log_PIB')
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7efee9e379a0>
```

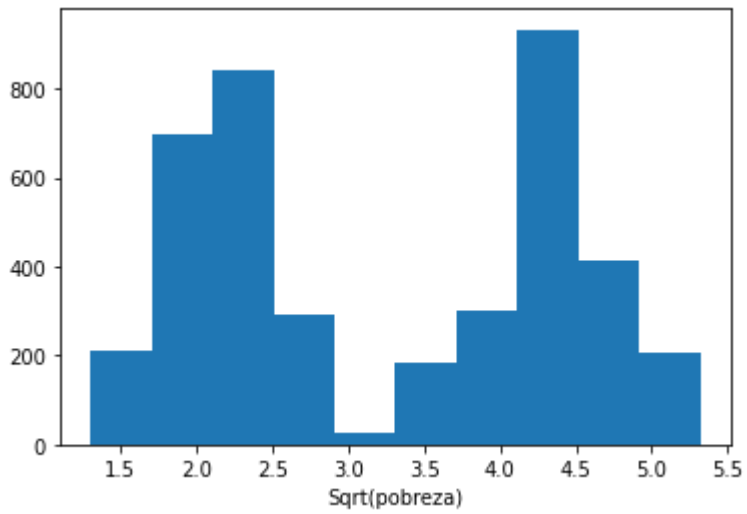


In [13]:
```
x_pobreza = np.array(data_out['pobreza'])
h = plt.hist(np.sqrt(x_pobreza+1));
plt.xlabel('Sqrt(pobreza)')

data_out['Sqrt_pobreza'] = np.sqrt(x_pobreza)
attrs = attrs + ['Sqrt_pobreza']
data_out[attrs].corr()
```

```
/home/maponti/.virtualenvs/mba_ds/lib/python3.8/site-packag
es/numpy/lib/histograms.py:839: RuntimeWarning: invalid val
ue encountered in greater_equal
  keep = (tmp_a >= first_edge)
/home/maponti/.virtualenvs/mba_ds/lib/python3.8/site-packag
es/numpy/lib/histograms.py:840: RuntimeWarning: invalid val
ue encountered in less_equal
  keep &= (tmp_a <= last_edge)
```

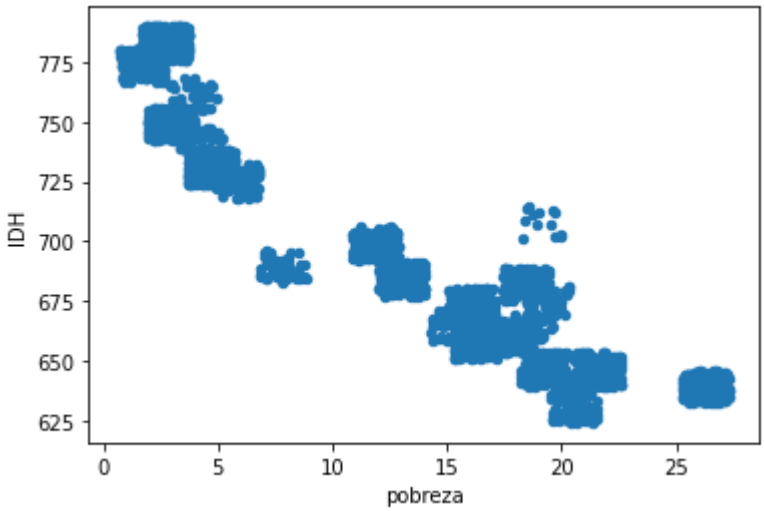
Out[13]:

	pobreza	IDH	PIB	classe	pop	Log_PIB	Sq
pobreza	1.000000	-0.942245	-0.170224	-0.507724	0.300206	-0.185381	0.3
IDH	-0.942245	1.000000	0.141545	0.419101	-0.221749	0.154362	-0.2
PIB	-0.170224	0.141545	1.000000	0.346233	0.666058	0.902932	0.6
classe	-0.507724	0.419101	0.346233	1.000000	-0.223733	0.366278	-0.2
pop	0.300206	-0.221749	0.666058	-0.223733	1.000000	0.649933	0.9
Log_PIB	-0.185381	0.154362	0.902932	0.366278	0.649933	1.000000	0.7
Sqrt_pop	0.320284	-0.242492	0.684638	-0.246793	0.966454	0.717255	1.0
Sqrt_pobreza	0.991053	-0.959522	-0.178007	-0.534099	0.298361	-0.195874	0.3



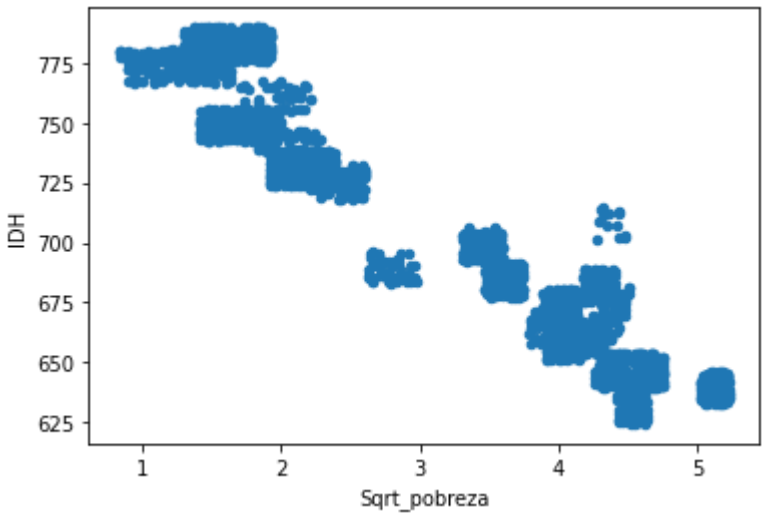
In [14]:
```
data_out.plot.scatter(x='pobreza', y='IDH')
```

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7efee9d68ac0>



In [15]:
```
data_out.plot.scatter(x='Sqrt_pobreza', y='IDH')
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7efee9cb7550>



Resumo:

- Encontrar transformações nos atributos permite novas interpretações, para além do que temos como intuição
- Transformações não excluem a limpeza e tratamento de dados
- Transformações podem funcionar como geradores de novos atributos