> A Estatística se refere às técnicas com as quais entendemos e resumimos os dados. É uma área extensa mas que aqui, será abordada de forma superficial.

Descrevendo um Conjunto de Dados

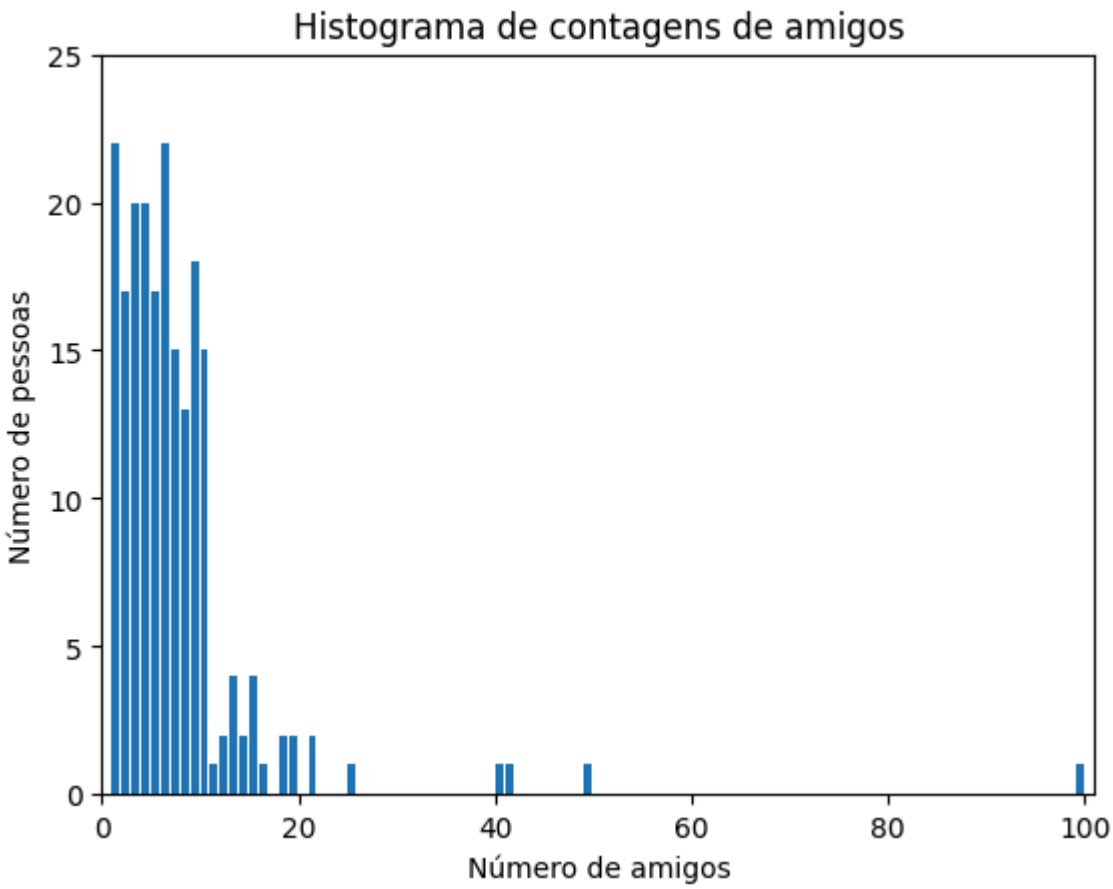
Com o crescimento forte da *Facedata*, nossa *CEO* têm interesse na descrição da quantidade de amigos que um usuário tem em geral. Usando as técnicas que vimos na introdução, você pode facilmente produzir essas informações. Mas agora você se depara com o problema de como descrevê-las. Uma descrição óbvia de qualquer conjunto de dados é simplesmente os próprios dados:

In [2]:
```
num_amigos = [100,49,41,40,25,21,21,19,19,18,18,16,15,
              15,15,15,14,14,13,13,13,13,12,12,11,10,
              10,10,10,10,10,10,10,10,10,10,10,10,10,
              10,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,
              8,8,8,8,8,8,8,8,8,8,8,8,7,7,7,7,7,7,
              7,7,7,7,7,7,7,7,6,6,6,6,6,6,6,6,6,6,
              6,6,6,6,6,6,6,6,6,6,6,6,5,5,5,5,5,5,5,
              5,5,5,5,5,5,5,5,5,5,4,4,4,4,4,4,4,4,4,
              4,4,4,4,4,4,4,4,4,4,4,3,3,3,3,3,3,3,3,
              3,3,3,3,3,3,3,3,3,3,3,3,2,2,2,2,2,2,2,
              2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,
              1,1,1,1,1,1,1,1,1,1,1,1,1]
```

Para um conjunto de dados *pequeno*, essa pode até ser a melhor descrição. Mas para um conjunto de dados *grande*, isso é pesado e provavelmente nebuloso (imagine olhar para uma lista de 1 milhão de números!). Por esse motivo, usamos estatísticas descritivas para resumir e comunicar características relevantes de nossos bancos de dados.

Como primeira abordagem, você coloca a contagem de amigos em um histograma como o da figura abaixo:

In [3]:
```
import matplotlib.pyplot as plt

plt.hist(num_amigos, range(max(num_amigos) + 1), width = .8)
plt.axis([0, 101, 0, 25])
plt.title("Histograma de contagens de amigos")
plt.xlabel("Número de amigos")
plt.ylabel("Número de pessoas")
plt.show()
```



Infelizmente, este gráfico ainda é muito difícil de ser comentado. Assim, você começa a gerar algumas estatísticas. Provavelmente, a estatística mais simples é o total de observações:

In [4]:
```
num_obs = len(num_amigos)           # 204
```

Você provavelmente também está interessado nos maiores e menores valores:

In [ ]:
```
maior_valor = max(num_amigos)       # 100
menor_valor = min(num_amigos)       # 1
```

que são casos especiais do interesse em saber os valores em ordens específicas:

In [ ]:
```
valores_ordenados = sorted(num_amigos)
menor_valor = valores_ordenados[0]          # 1
segundo_menor_valor = valores_ordenados[1]  # 1
segundo_maior_valor = valores_ordenados[-2] # 49
```

Mas estamos apenas começando...

> A partir de agora, teremos a tendência de utilizar a biblioteca `Pandas` para quase tudo! Ela foi criada para facilitar a manipulação de dados, tanto para gráficos quanto para estatística.

Assim, vamos tender a transformar nossos objetos do tipo `lista` para `pd.Series` ou `pd.DataFrame` da seguinte forma:

In [ ]:
```
# Objeto 'num_amigos'
type(num_amigos) # list
```

```
# Convertendo para pd.Series
import pandas as pd
num_amigos = pd.Series(num_amigos)
num_amigos

# Objeto 'num_amigos' atualizado
type(num_amigos) # pandas.core.series.Series (isso significa que e um pd.Series)
```

Tendências Centrais

Normalmente, queremos alguma noção de onde nossos dados estão centralizados. Mais comumente, usaremos a média (aritmética), que é apenas a soma dos dados divididos por sua contagem:

In [ ]:
```
print(num_amigos.mean())              # 7.33

# Esse calculo pode ser feito a mao:
print(sum(num_amigos) / len(num_amigos))  # 7.33
```

Se você tem dois valores, a média é simplesmente o valor no meio do caminho entre eles. À medida em que você adiciona mais pontos, a média muda, mas sempre depende do valor de cada ponto. Por exemplo, se você tiver 10 pontos de dados e aumentar o valor de qualquer um deles em 1, você aumentará a média em 0,1.

Às vezes, também estaremos interessados na mediana, que é o valor mais central (se o número de dados for ímpar) ou a média dos dois valores mais centrais (se o número de dados for par). Por exemplo, se temos cinco dados em um vetor ordenado `x`, a mediana é `x[5 // 2]` ou `x[2]`. Se tivermos seis pontos de dados, queremos a média de `x[2]` (o terceiro valor) e `x[3]` (o quarto valor).

Observe que, ao contrário da média, a mediana não depende totalmente de todos os valores dos dados. Por exemplo, se você aumentar o valor máximo (ou diminuir o valor mínimo), os valores centrais permanecerão inalterados, o que significa que a mediana também.

Assim, vamos calcular o número mediano de amigos:

In [ ]:
```
print(num_amigos.median())    # 6
```

Claramente, a média é mais simples de se calcular e varia suavemente à medida que nossos dados mudam. Em contraponto, para encontrar a mediana, precisamos ordenar nossos dados.

> Existem truques não triviais para calcular eficientemente medianas sem ordenar os dados. No entanto, estão além do escopo deste curso. Uma vez compreendido o conceito, vamos nos concentrar em utilizar funções prontas que são bem eficientes.

Ao mesmo tempo, a média é muito sensível a valores discrepantes (carinhosamente chamados de *outliers*). Se nosso usuário mais amigável tivesse 200 amigos (ao invés de 100), a média aumentaria para 7,82, enquanto a mediana permaneceria a mesma. Se os *outliers*

são dados ruins (ou não representativos de qualquer fenômeno que estamos tentando entender), então a média pode nos dar um resumo enganoso.

Por fim, a moda é o valor que aparece mais vezes no nosso conjunto de dados:

In [ ]:
```
print(num_amigos.mode())    # [1, 6]
```

Diferente das medidas anteriores, nem sempre a moda é única. Como vimos anteriormente, existem 2 modas neste conjunto de dados. As modas serão os valores que possuem as barras mais altas no histograma. Neste caso, temos um empate entre `1` e `6`.

A função do Python `stat.mode` se limita a exibir somente uma moda. Especificamente, a primeira que ele encontra. Tenha cuidado. Prefira utilizar a função `stat.multimode`.

Dispersão

A dispersão refere-se a medida de quão espalhados são nossos dados. Normalmente, são estatísticas das quais quanto menos espalhado são os dados, mais próxima de zero estarão. Por exemplo, uma medida muito simples é a amplitude, que é apenas a diferença entre o maior e o menor elemento:

In [ ]:
```
# amplitude
num_amigos.max() - num_amigos.min()  # 99
```

A amplitude é precisamente zero quando os valores máximo e mínimo são iguais. Isso só pode acontecer se os elementos de `x` forem iguais. Quando isso ocorre, representa que os dados estão o menos dispersos possível. Por outro lado, se a amplitude for grande, o valor máximo será *muito maior* que o valor mínimo. Consequentemente, os dados estarão mais espalhados.

Assim como a mediana, a amplitude não depende de todo o conjunto de dados. Um conjunto de dados cujos valores são todos 0 ou 100 tem a mesma amplitude de um conjunto de dados cujos valores são 0, 100 e muitos 50s. Intuitivamente, o primeiro conjunto de dados *deveria* ter maior dispersão.

Uma medida mais complexa de dispersão é a variância:

In [ ]:
```
num_amigos.var() # 81.54
```

Agora, qualquer que seja a unidade em que nossos dados estejam (por exemplo, "amigos"), todas as nossas medidas de tendência central (média, mediana, moda) estão nessa mesma unidade. A amplitude será também nessa mesma unidade. Por outro lado, a variância tem unidades que são o quadrado das unidades originais (por exemplo, `amigos ao quadrado'). Como pode ser difícil entender isso, geralmente analisamos o desvio padrão:

In [ ]:
```
num_amigos.std()  # 9.03
```

Correlação

Nossa excelentíssima *CEO* tem uma teoria de que a quantidade de tempo que as pessoas passam nas redes sociais está relacionada ao número de amigos que elas têm. A lista chamada `minutos_diarios` mostra quantos minutos por dia cada usuário gasta na *Facedata*, e você a ordenou de modo que seus elementos correspondam aos elementos da nossa lista anterior `num_amigos`. Gostaríamos de investigar a relação entre essas duas métricas.

Veremos primeiro a covariância, o análogo pareado da variância. Enquanto a variância mede como uma única variável se desvia de sua média, a covariância mede como duas variáveis variam em conjunto de suas médias:

In [6]:
```
# Dados
num_amigos = [100,49,41,40,25,21,21,19,19,18,18,16,15,
              15,15,15,14,14,13,13,13,13,12,12,11,10,
              10,10,10,10,10,10,10,10,10,10,10,10,10,
              10,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,
              8,8,8,8,8,8,8,8,8,8,8,8,8,7,7,7,7,7,7,
              7,7,7,7,7,7,7,7,6,6,6,6,6,6,6,6,6,6,6,
              6,6,6,6,6,6,6,6,6,6,6,6,5,5,5,5,5,5,5,
              5,5,5,5,5,5,5,5,5,5,4,4,4,4,4,4,4,4,4,
              4,4,4,4,4,4,4,4,4,4,4,3,3,3,3,3,3,3,3,
              3,3,3,3,3,3,3,3,3,3,3,3,2,2,2,2,2,2,2,
              2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,
              1,1,1,1,1,1,1,1,1,1,1,1,1]

minutos_diarios = [1,68.77,51.25,52.08,38.36,44.54,57.13,51.4,
                   41.42,31.22,34.76,54.01,38.79,47.59,49.1,
                   27.66,41.03,36.73,48.65,28.12,46.62,35.57,
                   32.98,35,26.07,23.77,39.73,40.57,31.65,31.21,
                   36.32,20.45,21.93,26.02,27.34,23.49,46.94,30.5,
                   33.8,24.23,21.4,27.94,32.24,40.57,25.07,19.42,
                   22.39,18.42,46.96,23.72,26.41,26.97,36.76,40.32,
                   35.02,29.47,30.2,31,38.11,38.18,36.31,21.03,30.86,
                   36.07,28.66,29.08,37.28,15.28,24.17,22.31,30.17,
                   25.53,19.85,35.37,44.6,17.23,13.47,26.33,35.02,
                   32.09,24.81,19.33,28.77,24.26,31.98,25.73,24.86,
                   16.28,34.51,15.23,39.72,40.8,26.06,35.76,34.76,
                   16.13,44.04,18.03,19.65,32.62,35.59,39.43,14.18,
                   35.24,40.13,41.82,35.45,36.07,43.67,24.61,20.9,
                   21.9,18.79,27.61,27.21,26.61,29.77,20.59,27.53,
                   13.82,33.2,25,33.1,36.65,18.63,14.87,22.2,36.81,
                   25.53,24.62,26.25,18.21,28.08,19.42,29.79,32.8,
                   35.99,28.32,27.79,35.88,29.06,36.28,14.1,36.63,
                   37.49,26.9,18.58,38.48,24.48,18.95,33.55,14.24,
                   29.04,32.51,25.63,22.22,19,32.73,15.16,13.9,27.2,
                   32.01,29.27,33,13.74,20.42,27.32,18.23,35.35,28.48,
                   9.08,24.62,20.12,35.26,19.92,31.02,16.49,12.16,30.7,
                   31.22,34.65,13.13,27.51,33.2,31.57,14.1,33.42,17.44,
                   10.12,24.42,9.82,23.39,30.93,15.03,21.67,31.09,33.29,
                   22.61,26.89,23.48,8.38,27.81,32.35,23.84]

horas_diarias = [x / 60 for x in minutos_diarios]
```

In [11]:
```
# Transformando para pd.DataFrame (porque agora temos + de 1 coluna)
import pandas as pd
df_qtd_tempo = pd.DataFrame({
    "num_amigos": num_amigos,
    "minutos_diarios": minutos_diarios,
    "horas_diarias": horas_diarias
})
```

In [12]:
```
# covariancia
df_qtd_tempo.cov()
```

Out[12]:

	num_amigos	minutos_diarios	horas_diarias
num_amigos	81.543514	22.425435	0.373757
minutos_diarios	22.425435	100.785899	1.679765
horas_diarias	0.373757	1.679765	0.027996

- Quando os elementos correspondentes de `x` e `y` estão ambos acima de suas médias ou ambos abaixo de suas médias, um número positivo entra na conta;
- Quando um está acima de sua média e o outro abaixo, um número negativo entra na conta.

Assim, uma covariância positiva *grande* significa que `x` tende a crescer quando `y` aumenta e decrescer quando `y` diminui. Em contraponto, uma covariância negativa *grande* significa o oposto – que `x` tende a decrescer quando `y` aumenta e vice-versa. Uma covariância próxima de zero significa que não existe tal relação.

No entanto, esse número pode ser difícil de interpretar, por alguns motivos:

- Suas unidades são o produto das unidades dos elementos (por exemplo, amigo-minutos-por-dia), o que pode ser difícil de entender;
- Se cada usuário tivesse o dobro de amigos (mas o mesmo número de minutos), a covariância seria duas vezes maior. Mas, em certo sentido, as variáveis seriam igualmente correlacionadas. Dito de outra forma, é difícil dizer o que é uma covariância *grande*.

Por esta razão, é mais simples olhar para a correlação - simplesmente é a covariância padronizada em um intervalo:

In [13]:
```
# correlacao
df_qtd_tempo.corr()
```

Out[13]:

	num_amigos	minutos_diarios	horas_diarias
num_amigos	1.00000	0.24737	0.24737
minutos_diarios	0.24737	1.00000	1.00000
horas_diarias	0.24737	1.00000	1.00000

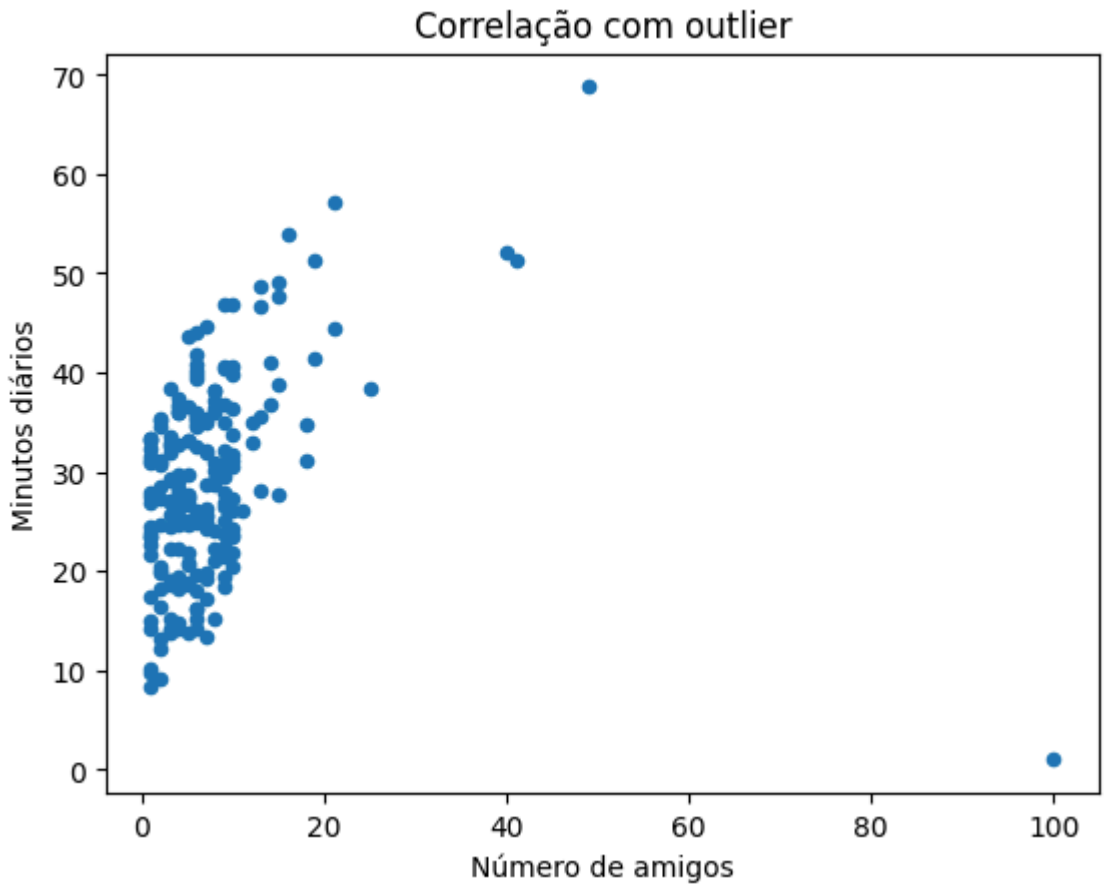
A correlação não tem unidade e sempre fica entre -1 (correlação positiva perfeita) e 1 (correlação negativa perfeita).

Um número como 0,25 representa uma correlação positiva relativamente fraca.

No entanto, uma coisa que deixamos de fazer foi examinar nossos dados:

In [14]:
```
df_qtd_tempo.plot.scatter(x='num_amigos', y='minutos_diarios')
plt.title("Correlação com outlier")
```

```
plt.xlabel("Número de amigos")
plt.ylabel("Minutos diários")
plt.show()
```



A pessoa com 100 amigos (que gasta apenas 1 minuto por dia no site) é um grande *outlier*, e a correlação pode ser muito sensível a *outliers*. O que acontece se o ignorarmos?

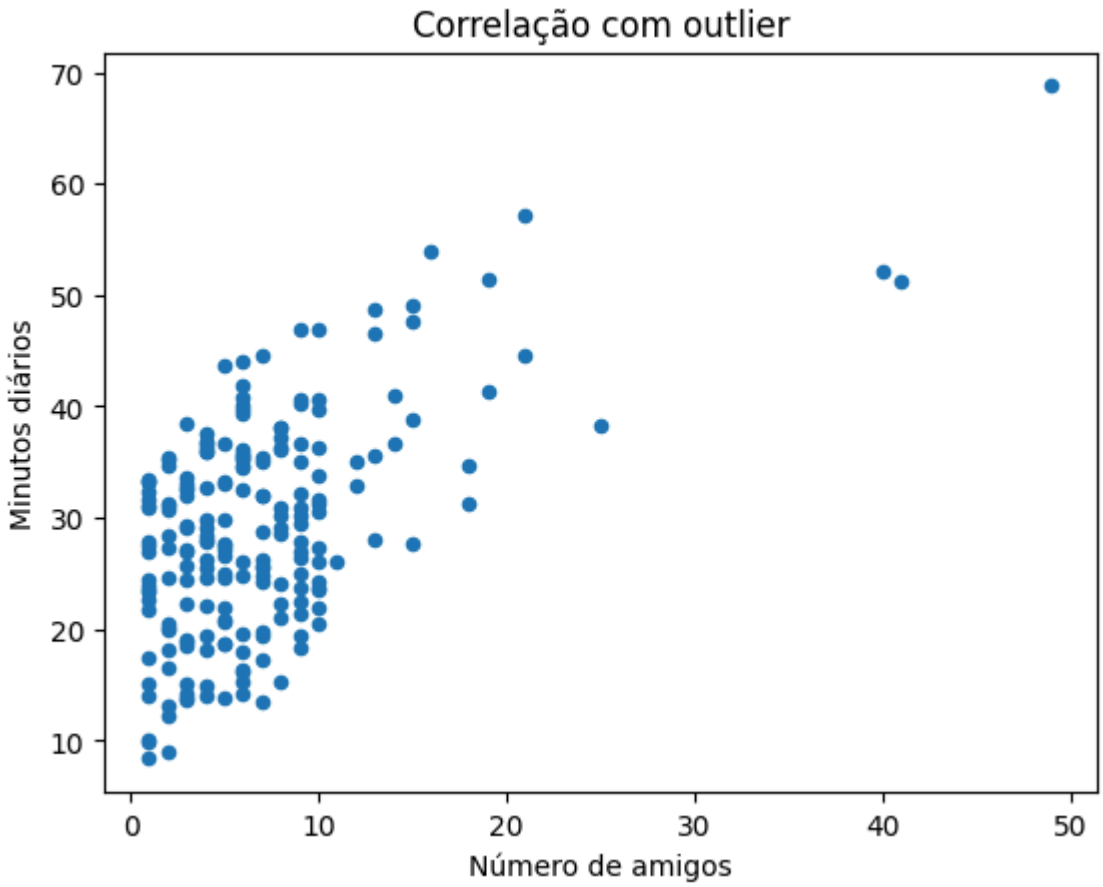
In [15]:
```
# Mantendo somente as observacoes onde temos menos de 100 amigos
# Ou seja, excluindo a linha onde o numero de amigos e > 100 (sabemos que so temos
df_qtd_tempo_sem_out = df_qtd_tempo[df_qtd_tempo['num_amigos'] < 100]

# correlacao (sem outlier)
df_qtd_tempo_sem_out.corr()
```

Out[15]:

	num_amigos	minutos_diarios	horas_diarias
num_amigos	1.000000	0.573679	0.573679
minutos_diarios	0.573679	1.000000	1.000000
horas_diarias	0.573679	1.000000	1.000000

Sem o *outlier*, há uma correlação muito mais forte:

In [16]:
```
df_qtd_tempo_sem_out.plot.scatter(x='num_amigos', y='minutos_diarios')
plt.title("Correlação com outlier")
plt.xlabel("Número de amigos")
plt.ylabel("Minutos diários")
plt.show()
```



Você investiga e descobre que o valor discrepante era na verdade uma conta de teste interna que ninguém nunca se preocupou em remover. Então, sua exclusão é justificada.

Ressalvas Sobre Correlação

Uma correlação de zero indica que não há relação **linear** entre duas variáveis. No entanto, pode haver outros tipos de relações. Por exemplo, se:

In [ ]:
```
x = [-2, -1, 0, 1, 2]
y = [ 2,  1, 0, 1, 2]
```

então `x` e `y` têm correlação (linear) zero. Mas eles certamente têm uma relação – cada elemento de `y` é igual ao valor absoluto do elemento correspondente ao de `x`:

$$y = |x|.$$

Exemplo prático

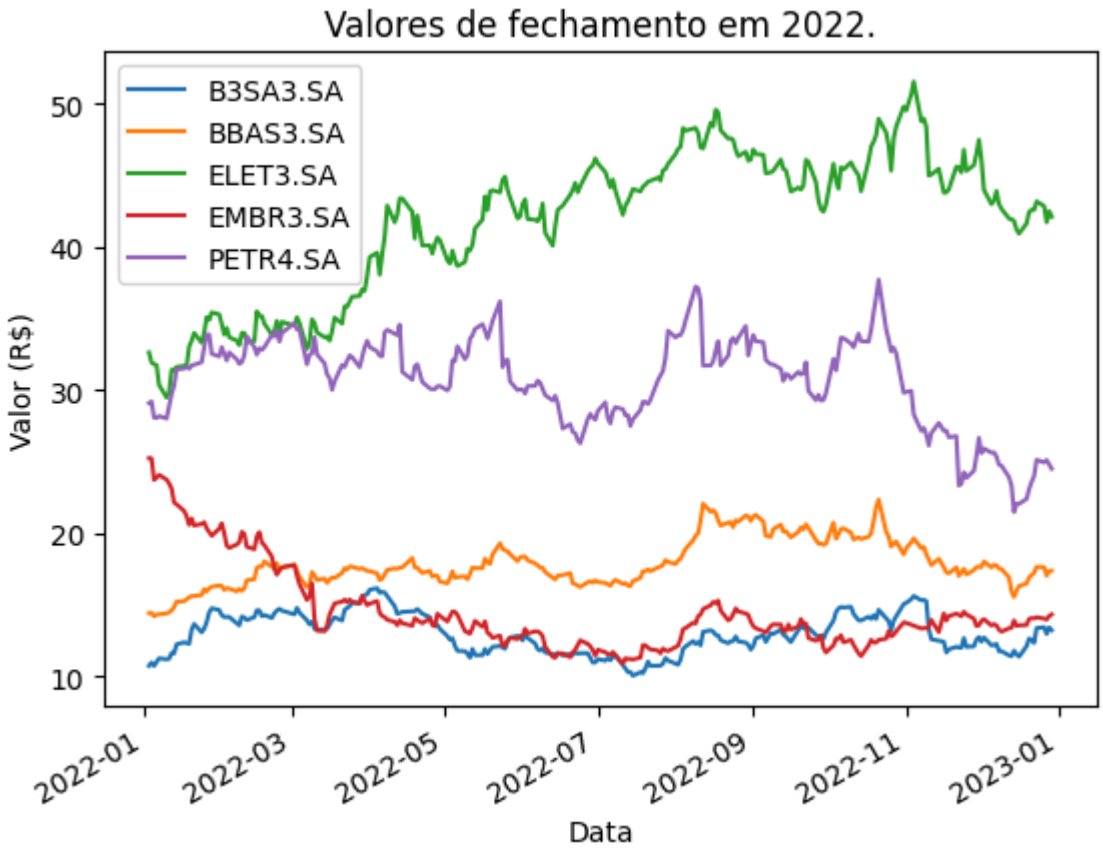
Vamos resgatar o exemplo de ações utilizado nas semanas anteriores:

In [ ]:
```
!pip install yfinance

import yfinance as yf

# B3SA3.SA: B3 S.A. - Brasil, Bolsa, Balcão
# BBAS3.SA: Banco do Brasil S.A.
# ELET3.SA: Centrais Elétricas Brasileiras S.A. - Eletrobrás
```

```
# EMBR3.SA: Embraer S.A.
# PETR4.SA: Petróleo Brasileiro S.A. - Petrobras
acoes_br = yf.download("B3SA3.SA BBAS3.SA ELET3.SA EMBR3.SA PETR4.SA",
                       start = "2022-01-01",
                       end = "2022-12-31")

# separando somente os valores de fechamento
acoes_br = acoes_br.iloc[:, 5:10]

# renomeando
acoes_br.columns = ['B3SA3.SA', 'BBAS3.SA',     'ELET3.SA',     'EMBR3.SA',     'PE
acoes_br.index.name = 'data'
```

Vamos analisar graficamente suas evolução ao longo do ano de 2023:

In [27]:
```
ax = acoes_br.plot()

ax.set_xlabel("Data")
ax.set_ylabel("Valor (R$)")
ax.set_title("Valores de fechamento em 2022.")
plt.show()
```



Vamos repetir as mesmas análises que fizemos anteriormente para ter conhecimento do relacionamento entre as ações dessas empresas:

In [24]:
```
acoes_br.corr()
```

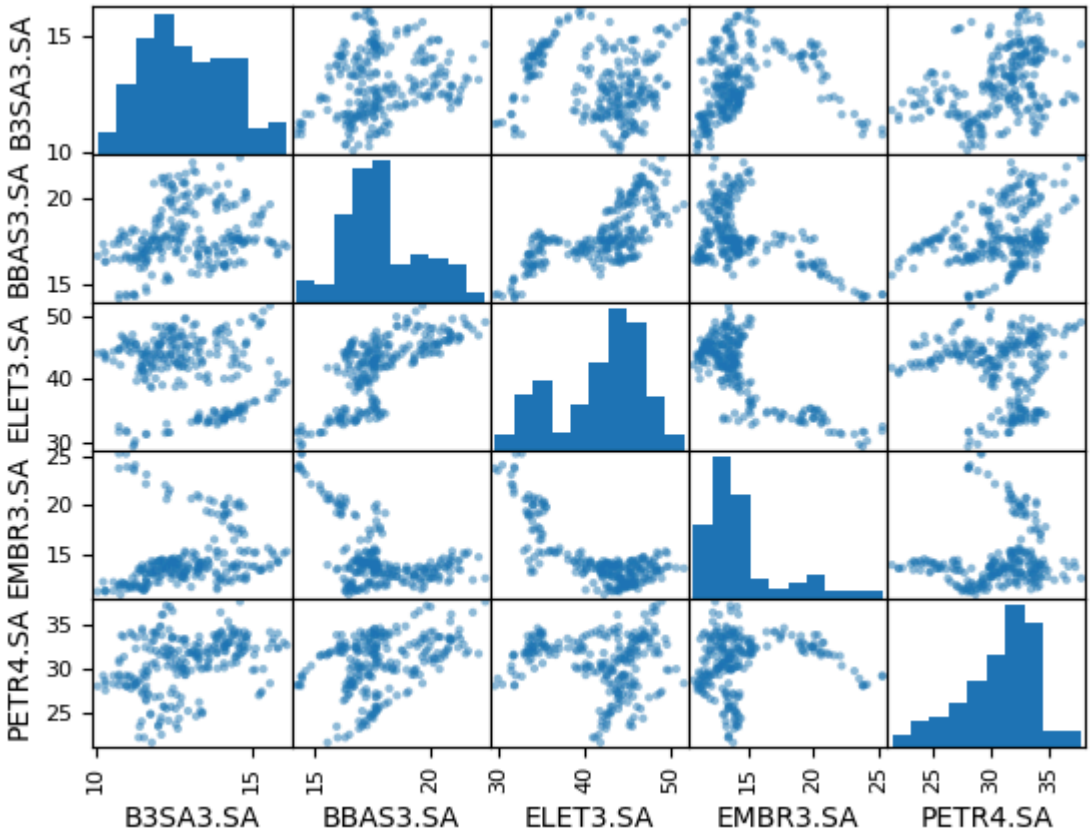
Out[24]:

	B3SA3.SA	BBAS3.SA	ELET3.SA	EMBR3.SA	PETR4.SA
B3SA3.SA	1.000000	0.227068	-0.163558	0.182042	0.366749
BBAS3.SA	0.227068	1.000000	0.708316	-0.476260	0.384282
ELET3.SA	-0.163558	0.708316	1.000000	-0.757648	-0.090468
EMBR3.SA	0.182042	-0.476260	-0.757648	1.000000	0.123630
PETR4.SA	0.366749	0.384282	-0.090468	0.123630	1.000000

Olhe para essa tabela (ou matriz) como o cálculo do coeficiente de correlação para cada possível par de variáveis (ou cada dimensão). Em destaque, temos o que chamamos de "diagonal principal". Nela, todos os valores são 1. Isso ocorre porque estamos calculando a correlação entre uma variável e ela mesmo. Outro detalhe interessante é que essa matriz é "simétrica". Ou seja, tudo que está acima da diagonal principal é idêntico ao que está abaixo. Isso ocorre porque, por exemplo, calcular a correlação entre a variável `X` e `Y` é equivalente a calcular a correlação entre `Y` e `X`.

Os números no ajudam muito na análise de um problema com muitas dimensões. Mas é sempre bem-vinda uma análise visual (quando possível). Podemos criar uma 'matriz' de gráficos de dispersão da seguinte forma:

In [26]:
```
graf = pd.plotting.scatter_matrix(acoes_br)
plt.show()
```



Vemos que as ações `BBAS3.SA` e `ELET3.SA` seguem um padrão semelhante. Isto resulta em uma forte correlação positiva de 0.708. Por outro lado, note que as ações `BBAS3.SA` e `EMBR3.SA` demonstram uma correlação forte negativa de -0.758. Ou seja, com essas informações, sabendo que a ação `BBAS3.SA` está se valorizando, um sábio investidor

tenderia a investir na ação `ELET3.SA`. Em contraste, ele fugiria de investimentos em `EMBR3.SA` nesse momento.

> Não necessariamente esse sábio investidor do exemplos ficaria rico através dessas análises. O mercado financeiro é muito mais complexo do que pensamos. No entanto, simples análises como essa ajudam (e muito) os analistas financeiros no mundo real.

Correlação e Causalidade

Em geral, **correlação não é causalidade**. Se `x` e `y` são fortemente correlacionados, isso pode significar que `x` causa `y`, `y` causa `x`, ou algum terceiro fator causa ambos.

Considere a relação entre `num_amigos` e `minutos_diarios`. É possível que ter mais amigos faça com que os usuários da *Facedata* passem mais tempo no site. Esse pode ser o caso se cada amigo postar uma certa quantidade de conteúdo por dia, o que significa que quanto mais amigos você tiver, mais tempo levará para se manter atualizado com suas atualizações.

No entanto, também é possível que quanto mais tempo os usuários gastam discutindo nos fóruns da *Facedata*, mais eles encontram e fazem amizade com pessoas de ideias semelhantes. Ou seja, passar mais tempo no site faz com que os usuários tenham mais amigos.

Uma terceira possibilidade é que os usuários mais ávidos por ciência de dados passem mais tempo no site (porque acham mais interessante) e coletem mais ativamente amigos de ciência de dados (porque não querem se associar a mais ninguém).

Referências Adicionais

- statistics, NumPy, SciPy, Pandas e statsmodels são alguns módulos que disponibilizam uma ampla variedade de funções estatísticas.