Professor Rodrigo Bossini
https://sites.google.com/site/professorrodrigobossini/

Data Science com Python 3
Visualização de dados

Passo 1 (Visualizando dados com matplotlib, gráfico de linha) A fim de visualizar nossos dados graficamente iremos utilizar a biblioteca matplotlib. Ela funciona da seguinte forma. Criamos um objeto que representa um gráfico e que mantém seus estado a cada método chamado. No final, podemos exportar o gráfico ou exibi-lo na tela. Veja um primeiro exemplo na Listagem 1.1. Trata-se de um gráfico que mostra a evolução do PIB de um país ao longo de algumas décadas.

Listagem 1.1

```
from matplotlib import pyplot as plt
years = [1950, 1960, 1970, 1980, 1990, 2000, 2010] # eixo x
gdp = [300.2, 543.3, 1075.9, 2862.5, 5979.6, 10289.7, 14958.3] #eixo y
#cria um gráfico
plt.plot (years, gdp, color='green', marker='o', linestyle='solid')
#adiciona título
plt.title ("GDP Nominal")
#rótulo do eixo Y
plt.ylabel ("Bilhões de $")
#mostra
plt.show()
```

Passo 2 (Gráficos de barra) A seguinte coleção de dados mostra a quantidade de Oscar's conquistados pelos filmes especificados. Neste cenário um gráfico de barra pode ser de interesse. Veja como criar um na Listagem 2.1.

Listagem 2.1

```
movies = ["Annie Hall", "Ben-Hur", "Casablanca", "Gandhi", "West Side
Story"]
num_oscars = [5, 11, 3, 8, 10]
xs = [i for i, _ in enumerate(movies)]
plt.bar(xs, num_oscars)
plt.ylabel("# de Premiações")
plt.xlabel ("#Meus filmes favoritos")
#nomeia o eixo x com nomes de filmes na barra central
plt.xticks ([i for i, _ in enumerate(movies)], movies)
plt.show()
```

Passo 3 (Gráficos de barra como histogramas) Também pode ser de interesse mostrar graficamente a distribuição de uma coleção de itens. A Listagem 3.1 mostra a distribuição de um grupo de alunos de acordo com a nota obtida.

Listagem 3.1

```
grades = [83, 95, 91, 87, 70, 0, 85, 82, 100, 67, 73, 77, 0]
decile = lambda grade: grade // 10 * 10 # calcula o decil
histogram = Counter (decile (grade) for grade in grades)
plt.bar (
    [
        x for x in histogram.keys()
    ],
    histogram.values(),
    8 # largura de 8 para cada barra
)
plt.axis ([-5, 105, 0, 5])# x: -5 a 105, y: 0 a 5
plt.xticks ([10 * i for i in range (11)])
plt.xlabel("Decil")
plt.ylabel("# de Alunos")
plt.title ("Distribuição das Notas do Teste 1")
plt.show()
```

Passo 4 (Não mostrar o valor 0 no eixo y pode não ser uma boa ideia) Note, pelo exemplo da Listagem 4.1, que deixar de exibir o valor 0 no eixo y pode causar uma impressão diferente da realidade. Neste exemplo, a diferença de 5 menções pode parecer maior do que de fato é.

Listagem 4.1

```
mentions = [500, 505]
years = [2013, 2014]
plt.bar ([2013, 2014], mentions, 0.4) #eixo x, eixo y, largura
plt.xticks (years)
plt.ylabel ("# de vezes que ouvimos alguém dizer data science")
plt.axis ([2012.5, 2014.5, 499, 506])
plt.title ("Olhe o grande aumento")
plt.show()
```

Veja que o exemplo da Listagem 4.2 especifica limites diferentes para os eixos, o que faz com que a diferença pareça menos importante e reflita melhor a realidade.

Listagem 4.2

```
mentions = [500, 505]
years = [2013, 2014]
plt.bar ([2013, 2014], mentions, 0.4) #eixo x, eixo y, largura
plt.xticks (years)
plt.ylabel ("# de vezes que ouvimos alguém dizer data science")
plt.axis([2012.5, 2014.5, 0, 550])
plt.show()
```

Passo 5 (Gráficos de linhas, múltiplas visualizações em um só gráfico) O gráfico de linhas possui diversas opções. A Listagem 5.1 mostra mais algumas.

Listagem 5.1

```
variance = [1, 2, 4, 8, 16, 32, 64, 128, 256]
bias_squared = [256, 128, 64, 32, 16, 8, 4, 2, 1]
total_error = [x + y for x, y in zip (variance, bias_squared)] #zip devolve
uma lista de tuplas
xs = [i for i, _ in enumerate (variance)]
#múltiplas chamadas para mostrar múltiplas séries do mesmo gráfico
plt.plot(xs, variance, 'g-', label="variance") #linha verde sólida
plt.plot(xs, bias_squared, 'r-.', label="bias^2") #linha de ponto tracejado
vermelho
plt.plot(xs, total_error, 'b:', label='total_error') #linha com pontilhado
azul
#loc=9 significa top center
plt.legend(loc=9)
plt.xlabel ("complexidade do modelo")
plt.title ("Compromisso entre polarização e variância")
plt.show()
```

Passo 6 (Gráficos de dispersão) Um gráfico de dispersão é a escolha certa para visualizar o relacionamento entre dois pares de conjuntos de dados. A Listagem 6.1 mostra o relacionamento entre o número de amigos que seus usuários têm e o número de minutos que eles passam no site por dia.

Listagem 6.1

```
friends = [70, 65, 72, 63, 71, 64, 60, 64, 67]
minutes = [175, 170, 205, 120, 220, 130, 105, 145, 190]
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
plt.scatter (friends, minutes)
#rotula cada posição
for label, friend_count, minute_count in zip(labels, friends, minutes):
    plt.annotate(
        label,
        xy = (friend_count, minute_count), # o ponto
        xytext = (5, -5), #o quão longe está o texto do ponto
        textcoords = 'offset points'
    )
plt.title ("Minutos Diários vs. Número de Amigos")
plt.xlabel ("# de amigos")
plt.ylabel ("minutos diários passados no site")
plt.show()
```

Passo 7 (Escolhendo a escala) Caso esteja gerando gráficos de dispersao para **variáveis comparáveis**, deixar a biblioteca escolher a escala pode não ser uma boa ideia.

7.1 A Listagem 7.1 mostra um exemplo em que os eixos mostram valores de notas, ou seja, tratam-se do mesmo fenômeno e são comparáveis.

Listagem 7.1

```
test_1_grades = [99, 90, 85, 97, 80]
test_2_grades = [100, 85, 60, 90, 70]
plt.scatter(test_1_grades, test_2_grades)
plt.title("Os eixos não são compatíveis")
plt.xlabel ("nota do teste 2")
plt.ylabel ("nota do teste 1")
plt.show()
```

7.2 A Listagem 7.2 informa que os eixos são comparáveis por lidarem com o mesmo fenômeno.

Listagem 7.2

```
test_1_grades = [99, 90, 85, 97, 80]
test_2_grades = [100, 85, 60, 90, 70]
plt.scatter(test_1_grades, test_2_grades)
plt.title("Os eixos não são compatíveis")
plt.xlabel ("nota do teste 2")
plt.ylabel ("nota do teste 1")
plt.axis("equal")
plt.show()
```

Bibliografia

GRUS, J. **Data Science from Scratch: First Principles with Python.** 1st ed. O'Reilly Media, 2015.