**Data Visualization – Explorando com Seaborn**

Link para o drive desse curso:

<https://drive.google.com/drive/folders/19fh9DUEh_xbR6bDFguL3oTYZO9tVx4la>.

Biblioteca que usaremos:

- <http://pandas.pydata.org/pandas-docs/stable/>.

- <http://seaborn.pydata.org/introduction.html>.

1. **Aula 1 – Importando e Traduzindo Dados:**
   1. Aprendemos a importar a biblioteca pandas, os dados e a atribuir eles à uma variável.
   2. Traduzi ou não uma base de dados é sempre uma discussão importante a se ter. O ideal é conversar com a equipe e decidir se seria uma boa prática ou não a depender do objetivo e exibição desses dados.
      1. Para processos lúdicos e educativos, seria interessante traduzir, já pra publicação em artigos, por exemplo, não.
      2. Utilizando os dados.columns, conseguimos visualizar o index das colunas em forma de lista, ao invés de apenas renomear:

dados.columns



* + 1. Para renomear criamos um dict, e usamos o df.rename(columns=rename) como já sabemos fazer.
    2. Para renomear os campos de um DataFrame precisamos criar um dicionário, just like the columns, contendo o que queremos substituir para a substituição, aí usamos a função df.s.map(dict) e atribuímos essa series mapeada a ela mesma:

sim\_nao = {'No' : 'Não', 'Yes': 'Sim'}

gorjetas.sobremesa = gorjetas.sobremesa.map(sim\_nao)

gorjetas.head(1)



* + - 1. Esse mesmo processo se repete para os campos das outras variáveis:

dias = {'Sun': 'domingo', 'Sat': 'sabado', 'Thur': 'quinta', 'Fri': 'sexta'}

gorjetas.dia = gorjetas.dia.map(dias)

gorjetas.head(1)



refeição = {'Dinner': 'Jantar', 'Lunch': 'Almoço'}

gorjetas.refeicao = gorjetas.refeicao.map(refeicao)

gorjetas.head(1)



* 1. Para saber mais: Principais comandos
     1. Df.tail(): exibe as cinco últimas observações;
     2. Link para a documentação: <http://pandas.pydata.org/pandas-docs/stable/>.
  2. O que aprendemos:
     1. Importamos um arquivo chamado tips.csv e armazenamos numa variável chamada dados;
     2. Conhecemos os dados que vamos analisar;
     3. Por questões didáticas, decidimos traduzir nossa base de dados do inglês para o português;
     4. Armazenamos os dados traduzidos em uma nova variável chamada gorjetas.

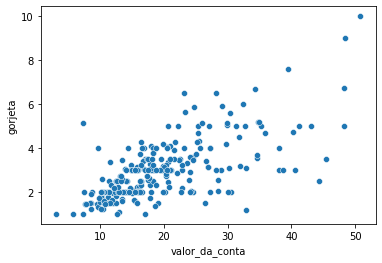
1. **Aula 2 – Comparando Distribuições de Maneira Explanatória:**
   1. A biblioteca feita para desenvolvimento e produção de gráficos é a matplotlib, mas nós utilizamos, hoje em dia, o pandas para análise, modificação e afins dos nossos dados, tendo essa biblioteca sido desenvolvida 10 anos depois da matplotlib.
      1. Nesse contexto a seaborn foi construída, sendo capaz de gerar gráficos bem mais sofisticados (usando a matplotlib por baixo dos panos) juntamente com o pandas e uma linguagem de bem mais alto nível do que a matplotlib poderia fazer.
      2. Importamos o seaborn como sns por convenção:

import seaborn as sns

* 1. Análise 1 - Valor da Conta e Gorjeta:
     1. Podemos gerar um gráfico scatterplot a partir das variáveis x = valor total da conta, com o y = valor da gorjeta, passando como a base de dados data = gorjetas. Dessa forma o gráfico será distribuído a partir do menor pro maior valor da conta no eixo x e da menor pra maior gorjeta no y:

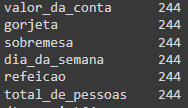
valor\_gorjeta = sns.scatterplot(x = 'valor\_da\_conta', y = 'gorjeta', data = gorjetas)

valor\_gorjeta



* + - 1. Podemos notar que parece ter uma progressão linear, onde conforme o valor da conta aumenta a gorjeta também aumenta.
    1. Para saber qual a contagem de observações não nulas usamos o count():

gorjetas.count()

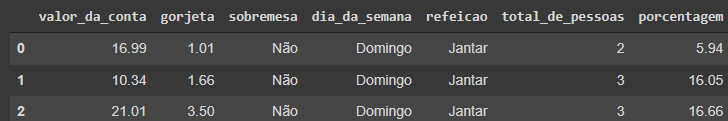


* 1. Podemos criar uma coluna com a porcentagem da gorjeta com relação ao valor total da conta:

gorjetas['porcentagem'] = (gorjetas.gorjeta / gorjetas.valor\_da\_conta) \* 100

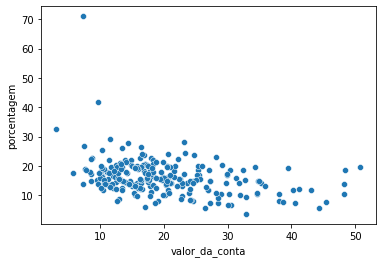
gorjetas.porcentagem = gorjetas.porcentagem.round(2)

gorjetas.head()



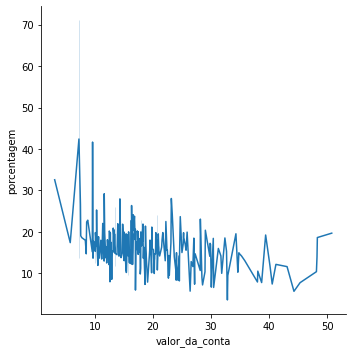
* + 1. E agora refazer o gráfico para ver se é proporcional a partir da porcentagem:

porcentagem\_conta = sns.scatterplot(x = 'valor\_da\_conta', y = 'porcentagem', data = gorjetas)



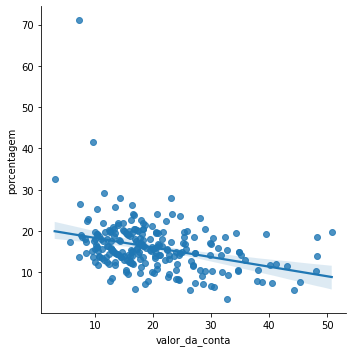
* + - 1. Podemos ver que, com relação ao outro gráfico, as contas com valores maiores deram mais gorjeta, mas, quando vemos esse gráfico, podemos concluir que não foram proporcionais, ou seja, visualmente o valor da conta não é proporcional ao valor da gorjeta.
  1. Podemos gerar o mesmo gráfico que acima, mas utilizando linhas ao invés de pontos a partir do sns.relplot(x = x, y = y, kind = ’line’, data = data). Se não passarmos o kind ele fará exatamente o mesmo gráfico que acima:

porcentagem\_conta\_linha = sns.relplot(x = 'valor\_da\_conta', y = 'porcentagem', kind = 'line', data = gorjetas)

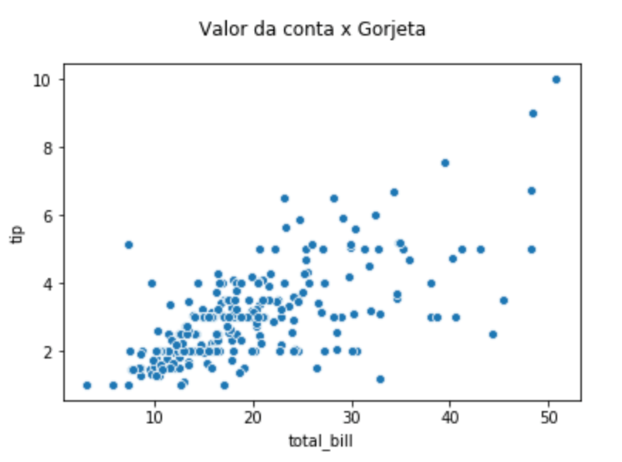


* + 1. E podemos ser ainda mais precisos na análise juntando ambos os gráficos com o sns.lmplot(x = x, y = y, data = data):

sns.lmplot(x = 'valor\_da\_conta', y = 'porcentagem', data= gorjetas)



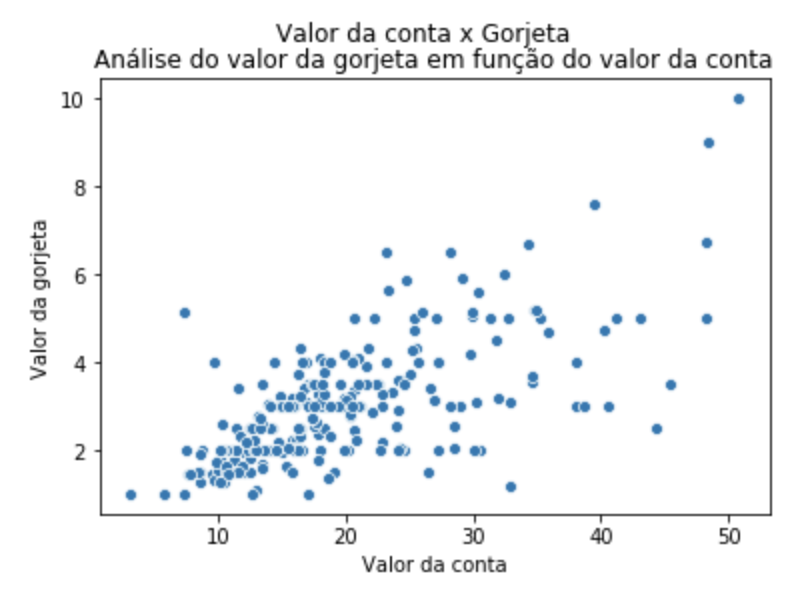
* + 1. Agora com esse gráfico podemos ter certeza que conforme aumenta o valor da conta, a porcentagem da gorjeta é menor, caindo o valor.
  1. Para saber mais:
     1. Podemos visualizar somente a imagem do nosso gráfico ou adicionar subtítulos, sem exibir o endereço da memória dele, com o get\_figure() e/ou o suptitle(‘titulo’), podendo ambos estarem juntos:
     2. primeiro\_plot.figure.suptitle('Valor da conta x Gorjeta'):



* + 1. Podemos também adicionar títulos com o set\_title(‘título’) e labels com o set(xlabel=’label’, ylabel=’label):

primeiro\_plot.set\_title('Análise do valor da gorjeta em função do valor da conta')

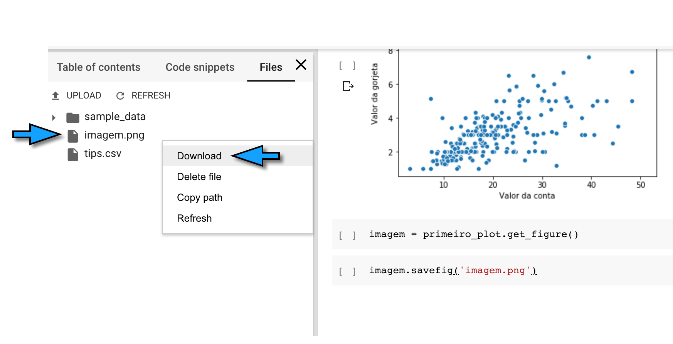
primeiro\_plot.set(xlabel='Valor da conta', ylabel='Valor da gorjeta')



* + 1. Podemos salvar o nosso gráfico com o img.savefig(‘nome.extensão’) e ela aparecerá no mesmo lugar onde fazemos upload de dados:

imagem = primeiro\_plot.get\_figure()

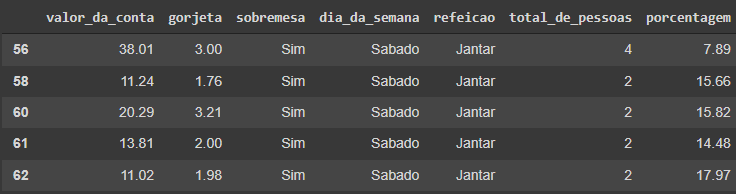
imagem.savefig('imagem.png')



* 1. O que aprendemos:
     1. Importamos o Seaborn para gerar diferentes gráficos;
     2. Analisamos de forma visual e descritiva o valor da gorjeta em relação ao valor da conta;
     3. Criamos diferentes tipos de gráficos.

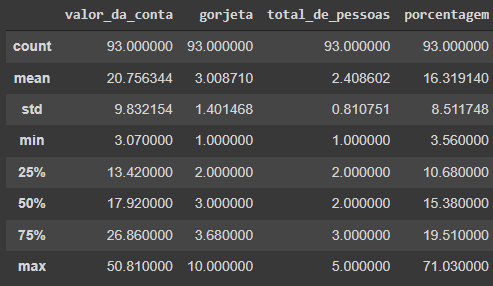
1. **Aula 3 – Analisando de Forma Visual e Descritiva:**
   1. Podemos fazer seleção de dados com condições no lugar da indexação:

gorjetas[gorjetas.sobremesa == 'Sim']



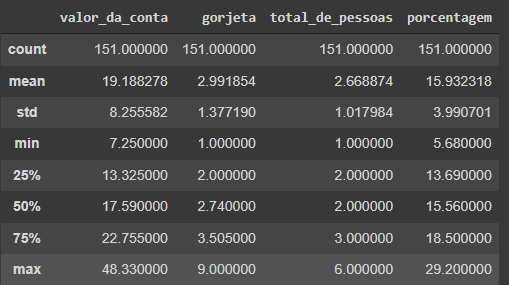
* + 1. Utilizamos o df.describe() para rebecer análises descritivas:

gorjetas[gorjetas.sobremesa == 'Sim'].describe()



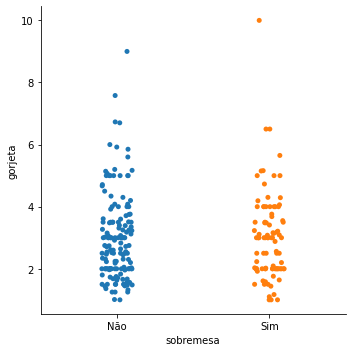
* + 1. Fizemos o mesmo para quem não pediu para fazer a comparação entre elas:

gorjetas[gorjetas.sobremesa == 'Não'].describe()



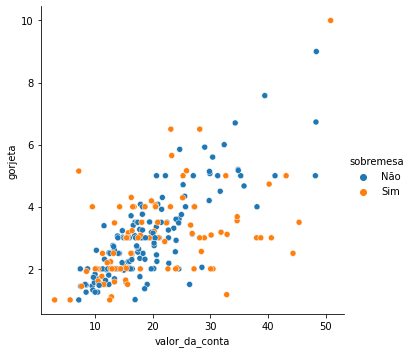
* 1. Podemos fazer um gráfico categórico com sns.catplot(x=x, y=y, data=data), dessa forma podemos comparar a gorjeta de acordo com quem pediu ou não sobremesa:

sns.catplot(x = 'sobremesa', y = 'gorjeta', data = gorjetas)



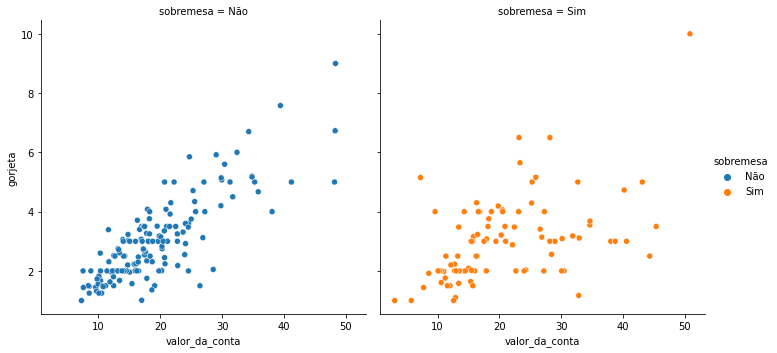
* + 1. Ainda assim não conseguimos notar uma diferença muito grande, então podemos criar o gráfico sns.relplot() como antes, mas passando um novo parâmetro chamado hue=’variável’, dessa forma ele mescla as duas categorias de uma variável no mesmo gráfico comparando outras duas:

sns.relplot(x = 'valor\_da\_conta', y = 'gorjeta', hue = 'sobremesa', data = gorjetas)



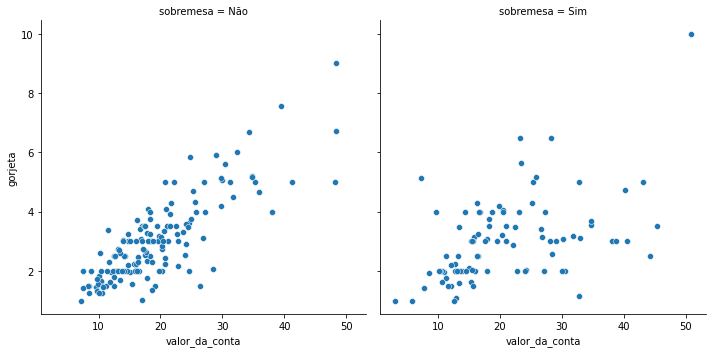
* + 1. Como ainda está meio confuso, podemos passar o parâmetro col=’variável’, dessa forma ele irá criar um gráfico para cada categoria dessas variáveis:

sns.relplot(x = 'valor\_da\_conta', y = 'gorjeta', hue = 'sobremesa', col = 'sobremesa', data = gorjetas)



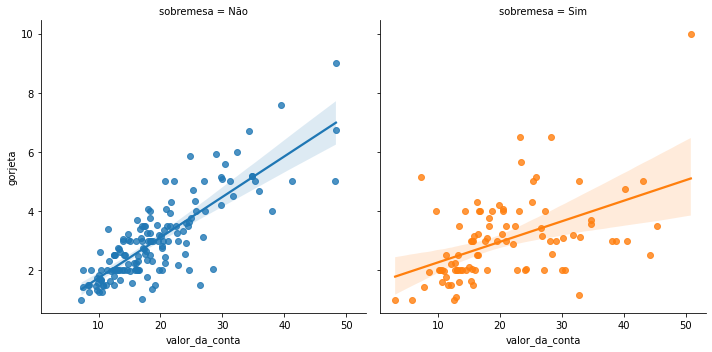
* + 1. Se tirarmos o hue=’variável’, os dois gráficos ficarão com a mesma tonalidade:

sns.relplot(x = 'valor\_da\_conta', y = 'gorjeta', col = 'sobremesa', data = gorjetas)



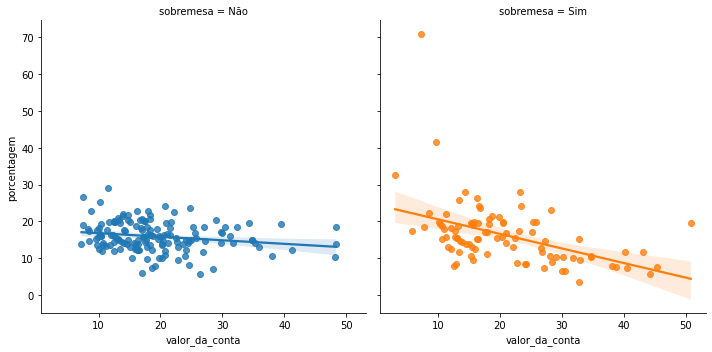
* + 1. Se quisermos traçar uma linha usamos o sns.lmplot() como já vimos. Podemos inclusive passar os exatos mesmos parâmetros que o sns.relplot():

sns.lmplot(x = 'valor\_da\_conta', y = 'gorjeta', hue='sobremesa', col='sobremesa', data = gorjetas)



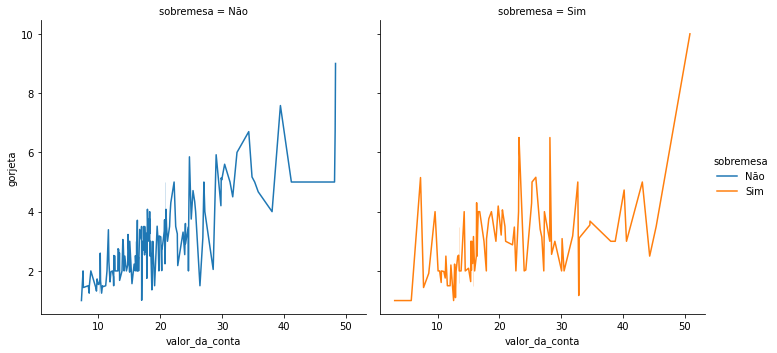
* + 1. Se ao invés de gorjeta utilizarmos a porcentagem veremos que a proporção de gorjeta de quem pediu sobremesa é menor do que a de quem não pediu, ou seja, visualmente existe uma diferença no valor da gorjeta daqueles que pediram sobremesa e não pediram sobremesa:

sns.lmplot(x = 'valor\_da\_conta', y = 'porcentagem', hue='sobremesa', col='sobremesa', data = gorjetas)



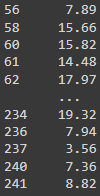
* + 1. Podemos ainda usar as linhas ao invés de pontos com o relplot:

sns.relplot(x = 'valor\_da\_conta', y = 'gorjeta', col = 'sobremesa', hue = 'sobremesa', kind = 'line', data = gorjetas)



* 1. Podemos pegar apenas uma series do nosso DataFrame fazendo uma seleção com o query:

sobremesa = gorjetas.query("sobremesa == 'Sim'").porcentagem



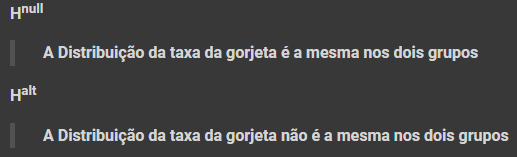
* + 1. Basicamente pegamos a porcentagem de todas as observações que pediram sobremesa do nosso df de gorjetas e atribuímos a variável sobremesa. Fizemos o mesmo para os que não pediram.
    2. Passando essas duas variáveis para a biblioteca ranksums da scipy.stats, ela nos devolve a estatística e o valor de p:

from scipy.stats import ranksums

ranksums(sobremesa, sem\_sobremesa)



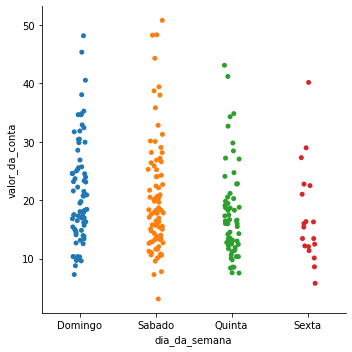
* + 1. Como nosso p é maior que 0.05, aceitamos a hipótese nula que determinamos no começo:



* + - 1. Ou seja, a variação das gorjetas nos grupos que pediram e não pediram sobremesa é insignificante, portanto, consideramos que sejam as mesmas.
  1. O que aprendemos:
     1. Fizemos uma análise descritiva das pessoas que pediram sobremesa e não pediram sobremesa com a função describe();
     2. Geramos gráficos alterando a tonalidade com parâmetro hue;
     3. Categorizamos uma informação em gráficos diferentes com o parâmetro col;
     4. Realizamos o teste de hipótese, analisando a porcentagem da gorjeta daqueles que pediram e não pediram sobremesa.

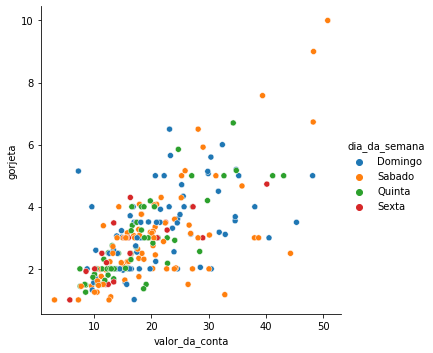
1. **Aula 4 – Comparando e Explorando Os Dias Da Semana:**
   1. Fizemos um gráfico categórico com catplot() com os dias da semana e a gorjeta:

sns.catplot(x = 'dia\_da\_semana', y = 'valor\_da\_conta', data = gorjetas)



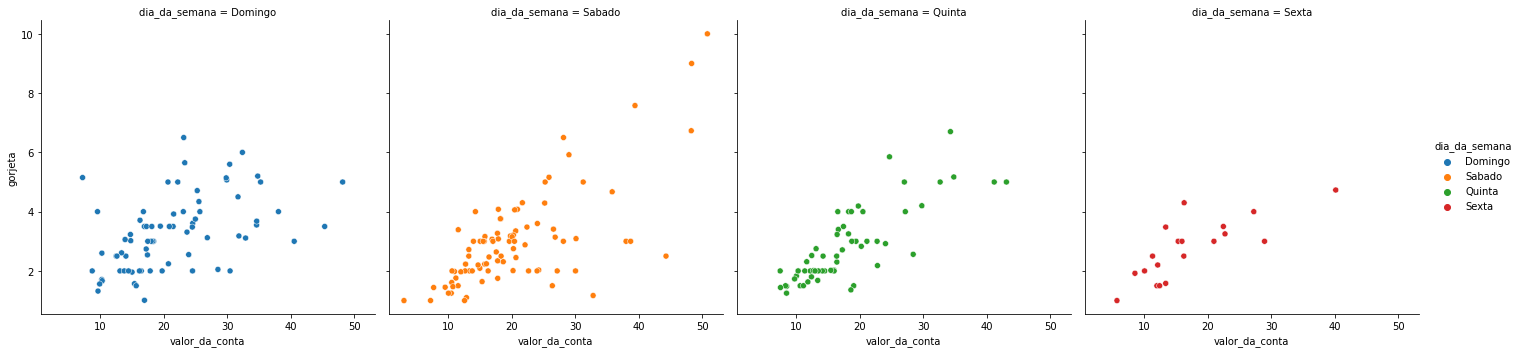
* + 1. Podemos ainda colocar todos juntos um relplot(hue=’dia\_da\_semana’):

sns.relplot(x = 'valor\_da\_conta', y = 'gorjeta', hue = 'dia\_da\_semana', data = gorjetas)

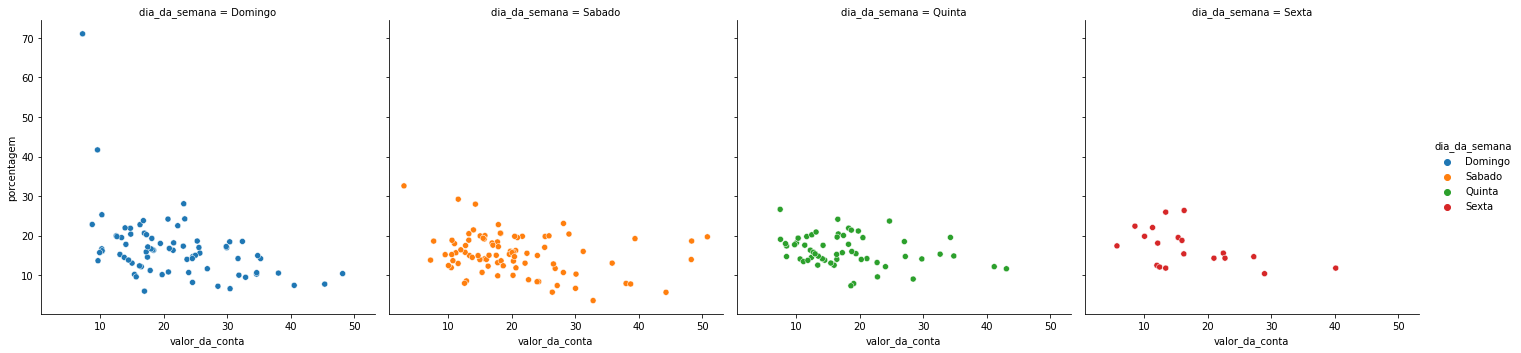


* + 1. Como ainda está confuso podemos separar um gráfico por dia e ainda fazer tanto do valor da gorjeta quanto da porcentagem:

sns.relplot(x = 'valor\_da\_conta', y = 'gorjeta', hue = 'dia\_da\_semana', col = 'dia\_da\_semana', data = gorjetas)

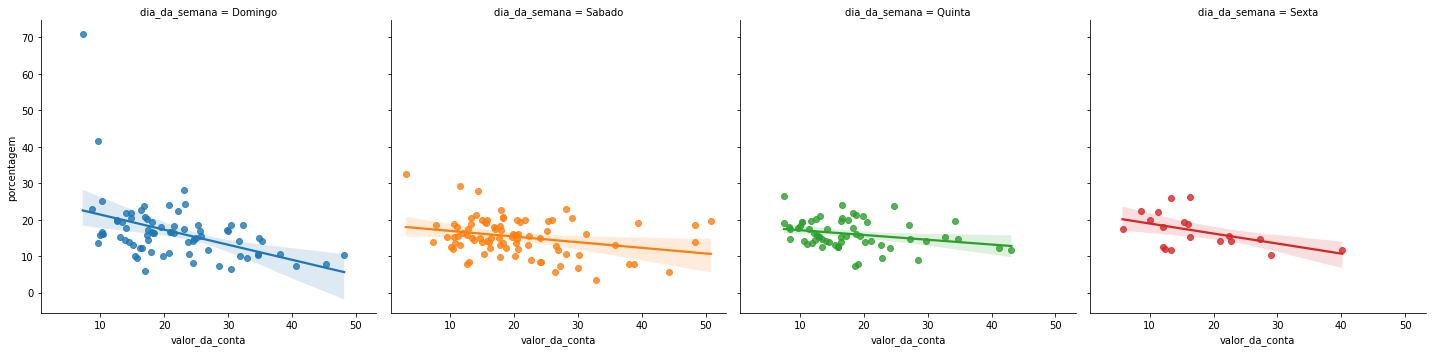


sns.relplot(x = 'valor\_da\_conta', y = 'porcentagem', hue = 'dia\_da\_semana', col = 'dia\_da\_semana', data = gorjetas)



* + 1. Ou então, pra melhorar, colocar uma linha:

sns.lmplot(x = 'valor\_da\_conta', y = 'porcentagem', hue = 'dia\_da\_semana', col = 'dia\_da\_semana', data = gorjetas)



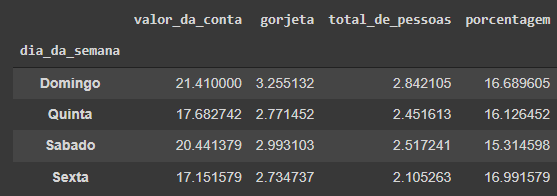
* 1. Podemos obter a média das gorjetas de várias formas, desde a mais geral, onde obtemos a média de todos os dias da semana, até agrupando todos os dados por dia e fazendo a média deles:

media\_geral\_gorjetas = gorjetas.gorjeta.mean()

print(f'A média geral das gorjetas é de {media\_geral\_gorjetas:.2f}')

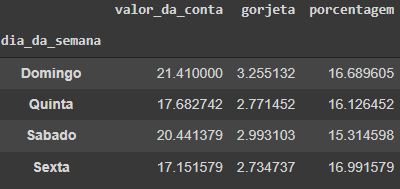


gorjetas.groupby(by = 'dia\_da\_semana').mean()



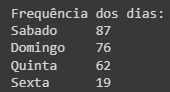
* + 1. Como não temos como ter 2.8 pessoas, podemos selecionar os dados que queremos obter a média passando o índice delas dentro de [[]] depois do mean():

gorjetas.groupby(by = 'dia\_da\_semana').mean()[['valor\_da\_conta', 'gorjeta', 'porcentagem']]

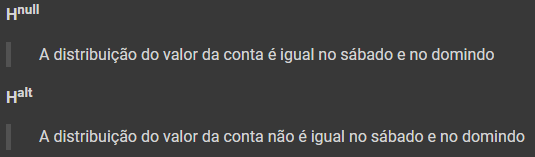


* + 1. Podemos observar que a porcentagem é bem parecida enquanto que a média do domingo é a maior de todas. Para ver o porquê, podemos obter a frequência de cada dia para ver qual é o dia que tiveram mais clientes:

print(f'Frequência dos dias:\n{gorjetas.dia\_da\_semana.value\_counts()}')



* + 1. Com isso vemos que, na realidade, o domingo é o segundo mais frequente, ou seja, tiveram mais pessoas no sábado do que no domingo e mesmo assim e sábado é o que possui a menor média.
    2. Com isso surge outra dúvida: será que no domingo as pessoas dão gorjetas maiores?
  1. Para responder à pergunta acima fizemos um teste de hipótese onde:



* + 1. Separamos os dados de domingo e sábado:

valor\_conta\_domingo = gorjetas.query("dia\_da\_semana == 'Domingo'").valor\_da\_conta

valor\_conta\_sabado = gorjetas.query("dia\_da\_semana == 'Sabado'").valor\_da\_conta

* + 1. E fizemos a análise de ranksums():

r2 = ranksums(valor\_conta\_domingo, valor\_conta\_sabado)

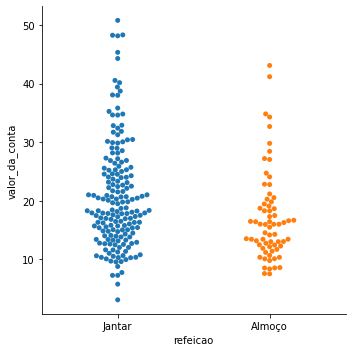
print(f'O valor de p é {r2.pvalue}')



* + 1. Ou seja, não recusamos H0, portanto, consideramos que a distribuição do valor da conta no sábado e domingo são iguais.
  1. O que aprendemos:
     1. Analisamos o valor da conta e da gorjeta de cada dia da semana;
     2. Em nossa análise descritiva, conferimos também a frequência de cada dia;
     3. Geramos também diferentes tipos de gráficos para uma análise exploratória visual.

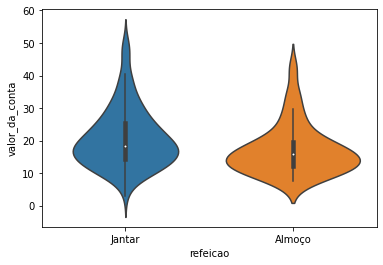
1. **Aula 5 – Distribuição de Frequência e Teste de Hipótese:**
   1. Começamos com as análises visuais da janta com relação ao valor da conta. Fizemos um catplot() e passamos um parâmetro novo: kind=’swarm’. Esse tipo faz com que as bolinhas fiquem bem separadas não sobrepondo umas as outras, assim conseguimos ter uma noção melhor de toda a dispersão dos dados:

sns.catplot(x = 'refeicao', y = 'valor\_da\_conta', kind = 'swarm', data = gorjetas)



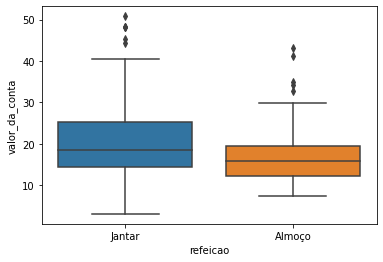
* + 1. Podemos ainda utilizar o sns.violinplot(), literalmente de violino, pra exibir nossos dados:

sns.violinplot(x = 'refeicao', y = 'valor\_da\_conta', data = gorjetas)



* + 1. Para ter uma melhor visualização da mediana, quartis e outliers dos nossos dados podemos fazer um sns.boxplot():

sns.boxplot(x = 'refeicao', y = 'valor\_da\_conta', data = gorjetas)



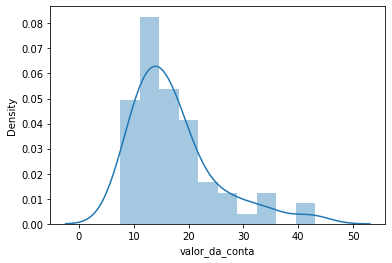
* + 1. Se quiser também podemos gerar um distplot(), mas diferente dos outros precisamos separar os dados que queremos fazer o gráfico, não simplesmente colocando x e y. Além disso podemos personalizar ele, exibindo ou não a linha na frente das barras com o kde=bool:

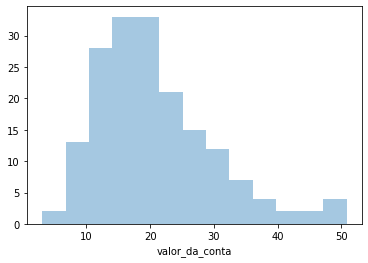
almoco = gorjetas.query('refeicao == "Almoço"').valor\_da\_conta

jantar = gorjetas.query('refeicao == "Jantar"').valor\_da\_conta

sns.distplot(almoco)

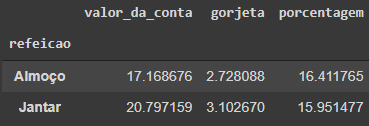
sns.distplot(jantar, kde=False)

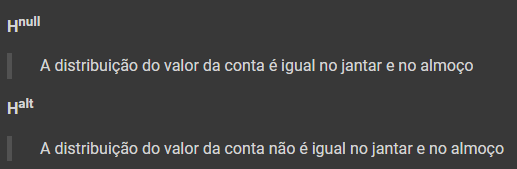




* + 1. O distplot() é um gráfico que tem, no eixo X, o valor da variável sendo exibida e no outro eixo, a frequência.
  1. Para ver se a distribuição da conta é igual no jantar e no almoço fizemos um groupby() e um ramksums() a partir da média dos grupos depois de definir nossas hipóteses:

gorjetas.groupby(by = 'refeicao').mean()[['valor\_da\_conta', 'gorjeta', 'porcentagem']]



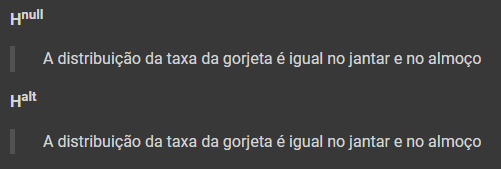


r3 = ranksums(jantar, almoco)

print(f'O valor de p é {r3.pvalue}')



* + 1. Como o valor de p foi menor que 0.05, recusamos H0 e aceitamos H1, ou seja, a distribuição do valor da conta não é igual no jantar e no almoço.
    2. Fizemos o mesmo cálculo, mas utilizando a porcentagem e chegamos na conclusão de que a distribuição da taxa da gorjeta é igual no jantar e no almoço, embora tenhamos visto acima que o valor da conta não é igual:



porcentagem\_almoco = gorjetas.query('refeicao == "Almoço"').porcentagem

porcentagem\_jantar = gorjetas.query('refeicao == "Jantar"').porcentagem

r4 = ranksums(porcentagem\_almoco, porcentagem\_jantar)

print(f'O valor de p é {r4.pvalue}')

