Centro Universitário de Excelência
Sistemas de Informação

Introdução a Análise Exploratória de Dados

Autor: Antônio Gabriel Gomes Falcão do Nascimento
Giovanna Oliveira Araujo
Gustavo Costa Pereira
João Magalhães Mira Neto

Feira de Santana, 2026

Agenda

O objetivo desta apresentação é demonstrar a aplicação de análise de dados criada pelo grupo.

1. Medidas de Tendência Central
Lógica das funções de Média, Mediana e Moda

2. Medidas de Dispersão e Variabilidade:
Lógica das funções de Variância, Desvio Padrão e Covariância

3. Conjunto de Itens Únicos (Itemset):
Lógica da função de Itemset

4. Distribuição e Frequência:
Explicação das funções de Frequência Absoluta, Frequência Relativa e Frequência Acumulada

5. Medidas de Posição (Quartis):
Explicação de Quartis

6. Análise de Probabilidade:
Explicação de Probabilidade Condicional

7. Análise exploratória - Spotify Data:
Métodos e resultados da análise aplicada aos dados do Spotify Data

1. Medidas de Tendência Central

- **Média Aritmética**: Soma de todos os valores dividida pela contagem total; indica o centro de equilíbrio.

```python
def mean(self, column):  2 usages  Antonio Gabriel +1 *
    values = self.dataset[column]

    if isinstance(values[0], str):
        print("Erro: coluna não numérica")
        return None

    if not values:
        return 0.0

    return sum(values) / len(values)
```

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

- **Mediana:** Valor central que divide o dataset ordenado ao meio; é robusta contra valores discrepantes (outliers)

```python
def median(self, column):  2 usages  Antonio Gabriel +1 *
    values = sorted(self.dataset[column])
    n = len(values)
    mid = n // 2
    if all(isinstance(v, (int, float)) for v in values):
        if n % 2:
            return values[mid]
        else:
            return (values[mid - 1] + values[mid]) / 2
    else:
        return values[mid]
```

Ímpar: $\frac{n+1}{2}$ Par: $\frac{X_{n/2} + X_{(n/2)+1}}{2}$

1. Medidas de Tendência Central

- **Moda**: O valor ou categoria que aparece com maior frequência no conjunto de dados.

```python
def mode(self, column):  2 usages  Antonio Gabriel +1 *
    values = self.dataset[column]
    frequency = {}
    for value in values:
        frequency[value] = frequency.get(value, default: 0) + 1
    max_freq = max(frequency.values())
    return [key for key, freq in frequency.items() if freq == max_freq]
```

2. Medidas de Dispersão e Variabilidade

- **Variância:** Média aritmética dos quadrados dos desvios em relação à média; mede quão longe os dados estão do centro.

```python
def variance(self, column):  2 usages  NanaGio +1 *
    if column not in self.dataset:
        return None

    dados = self.dataset[column]#extraindo os dados das colunas

    if len(dados) == 0:#caso a coluna esteja vazia
        return None

    media = sum(dados) / len(dados)#tirando a média da coluna

    soma_quadrados = sum((x - media) ** 2 for x in dados)#ao quadrado de cada desvio

    variancia_populacional = soma_quadrados / len(dados)#média novamente

    return variancia_populacional
```

$$s^2 = \frac{\sum_{i-1}^{n}(x_i - \bar{x})^2}{n-1}$$

- **Desvio Padrão:** Raiz quadrada da variância; expressa a dispersão na mesma unidade original dos dados.

```python
def stdev(self, column):  1 usage  NanaGio +1 *
    variancia_populacional = self.variance(column)#extraindo dados

    if variancia_populacional is None:#caso a coluna esteja vazia
        return None

    desvio = variancia_populacional ** 0.5#raiz quadrada

    return desvio
```

$$s = \sqrt{\frac{\sum_{i-1}^{n}(x_i - \bar{x})^2}{n-1}}$$

2. Medidas de Dispersão e Variabilidade

- **Covariância:** Indica a direção da relação linear entre duas variáveis. Se positiva, ambas tendem a crescer juntas; se negativa, uma cresce enquanto a outra diminui.

```python
def covariance(self, column_a, column_b):  1 usage  NanaGio +1 *
    valores_A = self.dataset[column_a]#extraindo os dados das colunas
    valores_B = self.dataset[column_b]#extraindo os dados das colunas

    n = len(valores_A)#len para saber o tamanho
    media_A = sum(valores_A) / len(valores_A)#média
    media_B = sum(valores_B) / len(valores_B)#média

    desvios_A = []#armazenar os devios da coluna A
    for x in valores_A:#listando os desvios
        desvios_A.append( x - media_A)

    desvios_B = []#armazenar os devios da coluna B
    for x in valores_B:#listando os desvios
        desvios_B.append( x - media_B)

    soma_produtos = 0#começand do zero a soma dos produtos

    for da, db in zip(desvios_A, desvios_B):#zip para unir em pares
        soma_produtos += (da * db)#armazenando a soma dos produtos

    resultado = soma_produtos / n #média de relacionamento

    return float(resultado)
```

$$Cov(X,Y) = \frac{\sum_{i-1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1} \quad (5)$$

3. Conjunto de Itens Únicos (Itemset)

Um itemset é um conjunto de valores únicos em uma coluna:

```python
def itemset(self, column):  1 usage  Nana +1
    """
    Retorna o conjunto de itens únicos em uma coluna.

    Parâmetros
    ----------
    column : str
        O nome da coluna (chave do dicionário do dataset).

    Retorno
    -------
    set
        Um conjunto com os valores únicos da coluna.
    """
    valores_unicos = set(self.dataset[column])# set -> separa os valores únicos

    return valores_unicos
```

4. Distribuição e Frequência

- **Frequência Absoluta:** Contagem simples de ocorrências de um valor ou intervalo específico.

```python
def absolute_frequency(self, column):  3 usages  João Mira +1 *
    dados = self.dataset[column]

    frequencia = {}  # espaço para os valores

    for item in dados:
        if item in frequencia:  # aqui ele pergunta se esse item ja existe no dicionario
            frequencia[item] += 1  # se existir, ele vai somar +1 a esse item ja existente
        else:
            frequencia[item] = 1  # caso não exista, ele cria uma entrada nova e seta o valor como 1

    return frequencia
```

$$\sum_{i-1}^{k} f_i = n$$

- **Frequência Relativa:** Porcentagem que cada valor representa em relação ao total da amostra.

```python
def relative_frequency(self, column):  2 usages  João Mira +1 *
    frequencia_absoluta = self.absolute_frequency(column)  # chama a função criada acima para fazer a contagem
    total = len(self.dataset[column])  # puxa o valor total de itens
    frequencia_relativa = {}  # espaço para as porcentagens

    for chave, contagem in frequencia_absoluta.items():  # separa as informações de frequencia absoluta em 2 campos
        frequencia_relativa[
            chave] = contagem / total  # faz com que o valor de frequencia relativa seja o resultado da divisão

    return frequencia_relativa
```

$$f_r = \frac{f_i}{n}$$

4. Distribuição e Frequência

- **Frequência Acumulada:** Soma sucessiva das frequências, permitindo visualizar a concentração dos dados até certo ponto.

```python
def cumulative_frequency(self, column, frequency_method='absolute'):  2 usages  João Mira +1 *
    if frequency_method == 'absolute':
        dados = self.absolute_frequency(column)  # chama a função que conta, caso tenha sido solicitada
    else:
        dados = self.relative_frequency(column)  # se não, chama a função de porcentagem

    if column == 'priority':
        ordem = ['baixa', 'media', 'alta']  # força uma ordem especifica caso a coluna trabalhada seja prioridade
    else:
        ordem = sorted(dados.keys())  # do contrario a ordem é alfabética

    acumulado = 0  # valor inicial setado em 0
    resultado = {}  # campo vazio, preenchido pós soma

    for chave in ordem:
        valor_atual = dados.get(chave,
                                default: 0)  # pega o valor relativo a cada idem na fila, seta 0 caso n exista nenhum representante

        acumulado = acumulado + valor_atual  # soma
        resultado[chave] = acumulado  # preenche o campo resultado com o valor da soma

    return resultado
```

$$F_i = \sum_{j=1}^{i} f_j \ (8)$$

4. Distribuição e Frequência

- **Histograma Por Buckets:** Agrupamento de dados contínuos em intervalos discretos para visualização de densidade.

$$k = 1 + 3,322 \cdot \log_{10}(n)$$

```python
def histogram(self, column, bins):  1 usage  João Mira +1 *
    valores = self.dataset[column]
    menor_valor,  valor_maior = min(valores), max(valores)
    numero_bins = 4
    tamanho_bin = (valor_maior - menor_valor) / numero_bins

    # Criação os limites dos intervalos (os buckets)
    limites = []
    for i in range(numero_bins + 1):
        ponto = menor_valor + (i * tamanho_bin)
        limites.append(ponto)

    # Geração do dicionário com tuplas como chaves (Início, Fim)
    # Ex: {(20.0, 35.0): 0, (35.0, 50.0): 0, ...}
    histograma = {}
    for i in range(numero_bins):
        intervalo = (limites[i], limites[i + 1])
        histograma[intervalo] = 0

    # Realização da contagem
    for valor in valores:
        indice = int((valor - menor_valor) / tamanho_bin)

        # Ajuste para o valor máximo
        if indice == numero_bins:
            indice -= 1

        # Recupera a chave (tupla) correspondente ao indice para incrementar (Ex: (20.0, 35.0))
        chave_intervalo = (limites[indice], limites[indice + 1])
        histograma[chave_intervalo] += 1

    return histograma
```

5. Medidas de Posição (Quartis)

- São medidas de posição que dividem um conjunto de dados ordenado em quatro partes iguais, cada uma contendo 25% da amostra. O Primeiro Quartil (Q1) delimita os 25% menores valores, o Segundo Quartil (Q2) corresponde à mediana (o ponto central de 50%), e o Terceiro Quartil (Q3) indica o limite abaixo do qual estão 75% dos dados.

$$IQR = Q3 - Q1 \ (9)$$

```python
def quartiles(self, column):  1 usage  João Mira +1 *
    # Recebendo os valores do dataset
    values = sorted(self.dataset[column])
    n = len(values)

    # Caso a quantidade de valores for igual a zero
    if n == 0:
        return {"Q1": 0, "Q2": 0, "Q3": 0}

    # Cálculo do Q2 (Mediana)
    mid = n // 2
    if n % 2 == 0:
        q2 = (values[mid - 1] + values[mid]) / 2
        # O fatiamento começa do início até o mid
        lower_half = values[:mid]
        # O fatiamento começa do mid até o final
        upper_half = values[mid:]
    else:
        q2 = values[mid]
        # Para n ímpar, a mediana (Q2) é excluída de ambas as metades
        lower_half = values[:mid]
        upper_half = values[mid + 1:]

    # Cálculo do Q1 (Mediana da Metade Inferior)
    n_lower = len(lower_half)
    mid_l = n_lower // 2
    if n_lower % 2 == 0:
        q1 = (lower_half[mid_l - 1] + lower_half[mid_l]) / 2
    else:
        q1 = (lower_half[mid_l])

    # Cálculo do Q3 (Mediana da Metade Superior
    n_upper = len(upper_half)
    mid_u = n_upper // 2
    if n_upper % 2 == 0:
        q3 = (upper_half[mid_u - 1] + upper_half[mid_u]) / 2
    else:
        q3 = (upper_half[mid_u])

    return {"Q1": q1, "Q2": q2, "Q3": q3}
```

6. Análise de Probabilidade

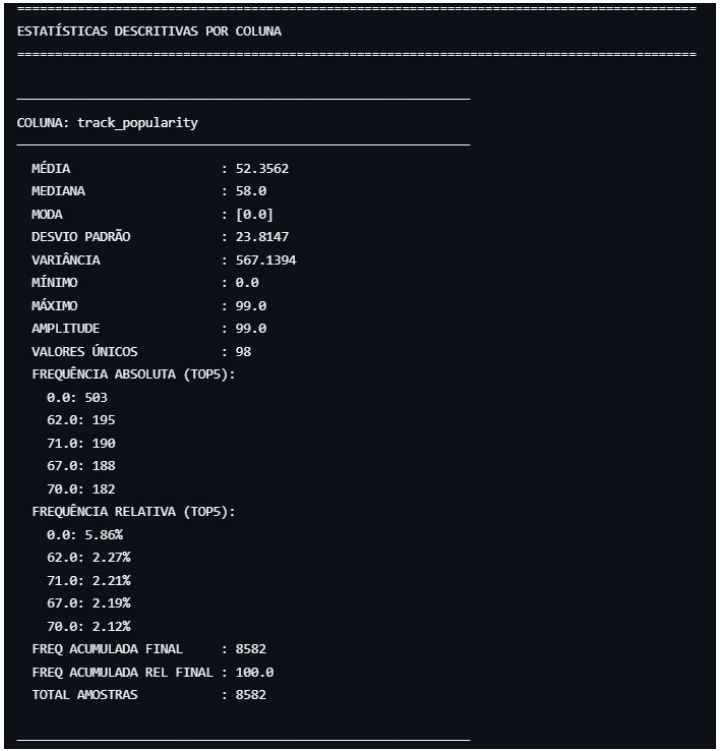
- **Probabilidade Condicional:** Cálculo da chance de um evento ocorrer dado que outro evento já aconteceu, aplicada em análises sequenciais de estados.

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

```python
def conditional_probability(self, column, value1, value2):  1 usage  João Mira +1 *
    # Recebe os valores do dataset
    values = self.dataset[column]

    # Contadores para a fórmula
    total_b = 0  # Quantas vezes o valor condicionante aparece (divisor)
    sucessos_ba = 0  # Quantas vezes a sequência (B -> A) ocorre (numerador)

    # Percorre até o penúltimo elemento
    for i in range(len(values) - 1):
        # Verifica se o elemento atual é o condicionante (B)
        if values[i] == value2:
            total_b += 1

            # Verifica se o consequente (A) aparece imediatamente após o condicionante (B)
            if values[i + 1] == value1:
                sucessos_ba += 1
    # Proteção contra divisão por zero (caso value2 não exista ou seja o último)
    if total_b == 0:
        return 0.0

    return sucessos_ba / total_b
```

7. Análise exploratória - Spotify Data

Exemplo da análise exploratória pelo relatório gerado em .txt:



```
================================================================================
ESTATÍSTICAS DESCRITIVAS POR COLUNA
================================================================================

--------------------------------------------------

COLUNA: track_popularity
--------------------------------------------------
  MÉDIA                  : 52.3562
  MEDIANA                : 58.0
  MODA                   : [0.0]
  DESVIO PADRÃO          : 23.8147
  VARIÂNCIA              : 567.1394
  MÍNIMO                 : 0.0
  MÁXIMO                 : 99.0
  AMPLITUDE              : 99.0
  VALORES ÚNICOS         : 98
  FREQUÊNCIA ABSOLUTA (TOP5):
    0.0: 503
    62.0: 195
    71.0: 190
    67.0: 188
    70.0: 182
  FREQUÊNCIA RELATIVA (TOP5):
    0.0: 5.86%
    62.0: 2.27%
    71.0: 2.21%
    67.0: 2.19%
    70.0: 2.12%
  FREQ ACUMULADA FINAL   : 8582
  FREQ ACUMULADA REL FINAL : 100.0
  TOTAL AMOSTRAS         : 8582

--------------------------------------------------
```

Referências

- WALPOLE, R. E.; MYERS, R. H.; MYERS, S. L.; YE, K. Probability & Statistics for Engineers & Scientists. 9. ed. Boston: Pearson, 2012.
- TRIOLA, M. F. Introdução à Estatística. 12. ed. Rio de Janeiro: LTC, 2017.