Implementação de uma Biblioteca Estatística Nativa em Python para Análise Exploratória de Dados

Andrey Mota de Oliveira[1], Claudio dos Santos Junior[1], Gabriel da Silva e Silva[1], Louise de Souza Santino[1]

[1]Curso de Sistemas de Informação – Centro Universitário de Excelência (UNEX)
Feira de Santana - BA - Brasil

deymotta686@gmail.com, claudiosjrel@gmail.com,
gabrielsilva82374055@gmail.com, louisesantino@hotmail.com

Resumo. Este trabalho apresenta a implementação de uma biblioteca estatística em Python utilizando apenas recursos nativos da linguagem. A classe Statistics foi desenvolvida para operar sobre datasets representados como dicionários de listas, com validações estruturais e validação de tipos por métrica. Foram implementadas métricas de tendência central, dispersão, associação, frequência e distribuição, além de testes unitários com unittest. Também foi realizada uma análise exploratória de dados em uma base real de músicas do Spotify, aplicando exclusivamente as operações implementadas na classe. Os resultados mostram consistência matemática das métricas e utilidade prática da biblioteca para análise exploratória.

Abstract. This work presents the implementation of a statistical library in Python using only native language resources. The Statistics class was designed to operate on datasets represented as dictionaries of lists, with structural validations and type validation per metric. Metrics for central tendency, dispersion, association, frequency, and distribution were implemented, along with unit tests using unittest. An exploratory data analysis was also carried out on a real Spotify songs dataset using only the operations implemented in the class. The results show mathematical consistency and practical usefulness of the library for exploratory analysis.

1. Introdução

A estatística é uma base essencial para a área de Dados, pois permite resumir, interpretar e comunicar padrões em conjuntos de dados. Em aplicações digitais, esse conhecimento é importante para entender comportamento de usuários, distribuição de conteúdos e relações entre variáveis.

Neste trabalho, foi desenvolvida uma biblioteca estatística própria em Python para apoiar atividades de Análise Exploratória de Dados (AED). A proposta foi implementar, manualmente, métricas estatísticas sem utilizar bibliotecas como numpy, pandas ou statistics, fortalecendo a compreensão das fórmulas e da lógica de programação.

O objetivo foi construir a classe Statistics, validar seu funcionamento com testes unitários e aplicá-la em uma análise exploratória real sobre um dataset de músicas do Spotify. A Seção 2 apresenta a fundamentação teórica; a Seção 3 descreve a

metodologia; a Seção 4 mostra resultados e discussões; e a Seção 5 traz as considerações finais.

2. Fundamentação Teórica

2.1 Validação do dataset

Antes dos cálculos, a estrutura do dataset deve ser válida: dataset no tipo dict; cada coluna no tipo list; todas as listas com o mesmo tamanho; e valores homogêneos por coluna.

Exemplo: impedir uma coluna com mistura de int e str, como participants = [10, 20, "30"].

2.2 Métricas de tendência central

Média (mean) - numérica.

$\bar{x} = (\Sigma\, xi) / N$

Exemplo: média de participants para estimar público médio.

Mediana (median) - numérica e ordinal. É o valor central do conjunto ordenado. Para a coluna priority, foi adotada ordem manual: baixa < media < alta.

Exemplo: mediana de priority para identificar prioridade central.

Moda (mode) - categórica/numérica.

$Moda = arg\ max\ f(x)$

Exemplo: moda de category para encontrar o tipo de evento mais frequente.

2.3 Métricas de dispersão

Variância (variance) - numérica. Foi adotada a variância populacional.

$\sigma^2 = [\Sigma\ (xi - \bar{x})^2] / N$

Exemplo: variância de ticket_price para medir dispersão dos preços.

Desvio padrão (stdev) - numérica.

$\sigma = \sqrt{\sigma^2}$

Exemplo: desvio padrão de ticket_price para interpretar a dispersão na mesma unidade.

2.4 Métricas de associação

Covariância (covariance) - numérica bivariada.

$Cov(X,Y) = [\Sigma\ (xi - \bar{x})(yi - \bar{y})] / N$

Exemplo: relação entre participants e ticket_price.

Probabilidade condicional (conditional_probability) - sequencial.

$P(A|B) = contagem(B \rightarrow A) / contagem(B)$

Exemplo: P(alta|media) na coluna priority.

2.5 Métricas de conjunto e frequência

Itemset (itemset).

Itemset(X) = {x em X}

Exemplo: valores únicos de priority.

Frequência absoluta (absolute_frequency).

f(x) = contagem de x

Exemplo: contagem de TRUE e FALSE em explicit.

Frequência relativa (relative_frequency).

fr(x) = f(x) / N

Exemplo: proporção de eventos com prioridade alta.

Frequência acumulada (cumulative_frequency): soma progressiva das frequências em ordem definida.

Exemplo: frequência acumulada de priority em ordem ordinal.

2.6 Métricas de posição e distribuição

Quartis (quartiles): Q1, Q2 e Q3 dividem os dados ordenados em quatro partes. Foi usado o método de interpolação linear.

*posição = (n - 1) * p*

Exemplo: quartis de participants para resumir a distribuição.

Histograma (histogram): agrupa valores em intervalos (bins) e conta ocorrências.

Exemplo: histograma de track_duration_min.

3. Metodologia

3.1 Implementação das métricas

O projeto foi organizado em duas classes: Statistics, com a implementação das métricas, e TestStatistics, com testes unitários em unittest. Foram usados apenas recursos nativos do Python, como dict, list, set, laços, condicionais, sorted, sum, min e max.

Houve reaproveitamento entre métodos, como variance() usando mean(), stdev() usando variance() e relative_frequency() usando absolute_frequency().

3.2 Validação das restrições do dataset

No método __init__, foram implementadas as validações de estrutura do dataset (dict), estrutura das colunas (list), consistência do número de linhas e homogeneidade de tipos por coluna. Essas verificações garantem consistência estrutural para os cálculos.

3.3 Validação do tipo para execução das métricas

Cada método valida a compatibilidade da coluna com a operação. As métricas mean, variance, stdev, covariance, quartiles e histogram exigem colunas numéricas. As métricas mode, itemset, absolute_frequency, relative_frequency e

cumulative_frequency aceitam colunas categóricas. A mediana também trata o caso ordinal da coluna priority.

Também há validações para coluna inexistente e coluna vazia, com erro ou retorno específico, conforme a métrica implementada.

3.4 Ajustes nos testes unitários

Durante a implementação, alguns valores esperados nos testes foram revisados para manter consistência com as fórmulas adotadas.

- Variância (ticket_price): ajustada de 507.25 para 525.25 (variância populacional).
- Desvio padrão (ticket_price): ajustado de 22.527756 para 22.918333.
- Covariância (participants, ticket_price): ajustada para 1212.25, com base na fórmula populacional.
- Quartis (participants): Q1 e Q3 ajustados para 65.0 e 157.5 pelo método de interpolação linear.

3.5 Análise exploratória no dataset Spotify

A análise exploratória foi realizada com scripts do projeto (carregar_dados.py, analise_popularidade.py, analise_duracao.py, analise_explicit.py e analise_artistas.py), com leitura do CSV por csv.DictReader, conversão manual de colunas numéricas, criação do objeto Statistics e aplicação das métricas implementadas.

4. Resultados e Discussões

4.1 Código da métrica de frequência relativa

```
    def relative_frequency(self, column):
        frequencia_absoluta = self.absolute_frequency(column)
        total_elementos = len(self.dataset[column])


        frequencias_relativas = {}
        for item, contagem in frequencia_absoluta.items():
            frequencias_relativas[item] = contagem / total_elementos


        return frequencias_relativas
```

4.2 Resultado de teste unitário

No teste de frequência relativa da coluna priority, os valores obtidos foram baixa = 0.3, media = 0.2 e alta = 0.5. Os resultados coincidiram com os valores esperados, validados com assertAlmostEqual.

4.3 Aplicação de duas métricas na base de eventos

1) Média de participantes: mean(participants) = 113.5. Interpretação: os eventos possuem, em média, 113,5 participantes.

2) Covariância entre participantes e preço: cov(participants, ticket_price) = 1212.25. Interpretação: a covariância positiva indica tendência de eventos com maior público terem preços maiores.

4.4 Análise exploratória no dataset Spotify

A análise foi aplicada ao arquivo spotify_data_clean.csv (8.582 músicas), com variáveis como popularidade, duração, conteúdo explícito e artista.

4.4.1 Popularidade (track_popularity)

Métricas aplicadas: média, mediana, moda, variância, desvio padrão e quartis.

- Média: 52,36
- Mediana: 58,0
- Moda: 0
- Variância: 567,14
- Desvio padrão: 23,814688
- Quartis: Q1 = 39,0; Q2 = 58,0; Q3 = 71,0

Discussão: a mediana acima da média sugere assimetria; a dispersão é alta para uma escala de 0 a 100.

4.4.2 Duração (track_duration_min)

Métricas aplicadas: média, mediana, variância, desvio padrão, quartis e histograma.

- Média: 3,49 min
- Mediana: 3,445 min
- Variância: 1,1192
- Desvio padrão: 1,0579 min
- Quartis: Q1 = 2,88; Q2 = 3,445; Q3 = 3,99

Discussão: as músicas se concentram entre 3 e 4 minutos, com dispersão menor que a popularidade.

4.4.3 Conteúdo explícito (explicit)

Métricas aplicadas: frequência absoluta, frequência relativa e comparação de média de popularidade.

- TRUE: 2148 músicas (25,0%)
- FALSE: 6434 músicas (75,0%)
- Popularidade média (explícitas): 57,82
- Popularidade média (não explícitas): 50,53

Discussão: músicas explícitas são minoria, mas apresentam maior popularidade média.

4.4.4 Artistas (artist_name)

Métricas aplicadas: itemset, frequências e análise de concentração.

- Artistas únicos: 2548
- Média de músicas por artista: 3,37
- Top 10 artistas: 1131 músicas (13,2%)
- Top 20 artistas: 1703 músicas (19,8%)
- Artistas com 1 música: 1581

Discussão: há concentração em poucos artistas e uma cauda longa de artistas com poucas faixas.

4.5 Comparação conceitual com bibliotecas

A implementação adotou variância populacional (divisor N), alinhada ao comportamento padrão de numpy.var (ddof = 0). Em pandas, DataFrame.var() usa ddof = 1 por padrão, o que pode produzir valores diferentes. Essa diferença de convenção explica divergências em comparações externas.

5. Considerações Finais

Em síntese, o objetivo de aprendizagem foi cumprido, pois o trabalho exigiu compreensão das métricas estatísticas e sua implementação manual em Python, além da aplicação em análise exploratória de uma base real.

Os principais desafios foram validar a estrutura do dataset e os tipos por coluna, tratar corretamente a coluna ordinal priority, implementar quartis com uma convenção estatística explícita e revisar testes unitários com valores divergentes.

Alguns pontos de melhoria essenciais para os próximos trabalhos seriam generalizar o tratamento de colunas ordinais, padronizar o comportamento para colunas vazias, adicionar suporte a valores ausentes, ampliar testes para cenários de erro e incluir novas métricas, como correlação, percentis e assimetria.

A aplicação no dataset Spotify mostrou que a classe Statistics atende bem a tarefas de AED em dados reais, com resultados coerentes e interpretáveis.

6. Referências

KAGGLE. Spotify Songs for ML & Analysis (8700+ tracks). Dataset de AlyAhmedTS13. Disponível em: https://www.kaggle.com/datasets/alyahmedts13/spotify-songs-for-ml-and-analysis-over-8700-tracks. Acesso em: 13 fev. 2026.

PYTHON SOFTWARE FOUNDATION. unittest - Unit testing framework. Disponível em: https://docs.python.org/3/library/unittest.html. Acesso em: 14 fev. 2026.

NUMPY DEVELOPERS. numpy.var - NumPy documentation. Disponível em: https://numpy.org/devdocs/reference/generated/numpy.var.html. Acesso em: 18 fev. 2026.

PANDAS DEVELOPMENT TEAM. pandas.DataFrame.var - pandas documentation. Disponível em: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.var.html. Acesso em: 18 fev. 2026.