CI-IA
Centro de Inovação em Inteligência Artificial para a Saúde

Introdução à Análise de Dados em Saúde com Python

Juliano de Souza Gaspar
Zilma Silveira Nogueira Reis
Isaias José Ramos de Oliveira
Ana Paula Couto da Silva
Cristiane dos Santos Dias

Realização
CI-IA
Centro de Inovação em Inteligência Artificial para a Saúde

Apoio
UFMG

Belo Horizonte. MG
2023

CENTRO DE INOVAÇÃO EM INTELIGÊNCIA ARTIFICIAL PARA A SAÚDE

CI-IA
Centro de Inovação em Inteligência Artificial para a Saúde

Introdução à Análise de Dados em Saúde com Python

Juliano de Souza Gaspar
Zilma Silveira Nogueira Reis
Isaias José Ramos de Oliveira
Ana Paula Couto da Silva
Cristiane dos Santos Dias

Belo Horizonte
2023

Autores
Juliano de Souza Gaspar
Zilma Silveira Nogueira Reis
Isaias José Ramos de Oliveira
Ana Paula Couto da Silva
Cristiane dos Santos Dias

Equipe de Tecnologia da Informação, Ambiente Virtual de Aprendizagem (Moodle) e Site do Curso
Isaias José Ramos de Oliveira
Joabe Dias Salgueiro
Marco Antunes Assis Costa
Wagner Bento de Magalhães

Apoio Técnico
Marina Nogueira Ferraz

Alunos de Iniciação Científica
André Soares da Silva
Guilherme Alexandre Silva Maia
Yago Jean de Almeida Nogueira
Teresa Vitória Carvalho Rocha

DOI: 10.5281/zenodo.7865448

ISBN: 978-65-86593-18-1

Universidade Federal de Minas Gerais

Reitora
Sandra Regina Goulart Almeida

Vice-Reitor
Alessandro Fernandes Moreira

Centro de Inovação em Inteligência Artificial para a Saúde - CI-IA Saúde

Diretoria
Virgílio Augusto Fernandes Almeida
Wagner Meira Junior
Antônio Luiz Pinho Ribeiro

Coordenadora de educação e difusão do conhecimento
Zilma Silveira Nogueira Reis

Coordenador de transferência de tecnologia
Gilberto Medeiros Ribeiro

Gerência de projetos
Fabiana Costa Pereira Peixoto
Letícia Santos Neto

Site: https://ciia-saude.medicina.ufmg.br
Email: cursosciiasaude@medicina.ufmg.br

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Introdução à análise de dados em saúde com Python [livro eletrônico] / Juliano de Souza Gaspar...[et al.]. -- Belo Horizonte, MG : Biblioteca J. Baeta Vianna : Universidade Federal de Minas Gerais, 2023.
PDF

Outros autores: Zilma Silveira Nogueira Reis, Isaias José Ramos de Oliveira, Ana Paula Couto da Silva, Cristiane dos Santos Dias.
Bibliografia.
ISBN 978-65-86593-18-1

1. Dados - Análise 2. Inteligência artificial 3. Serviços de saúde 4. Python (Linguagem de programação para computadores) I. Gaspar, Juliano de Souza. II. Reis, Zilma Silveira Nogueira. III. Oliveira, Isaias José Ramos de. IV. Silva, Ana Paula Couto da. V. Dias, Cristiane dos Santos.

23-161480 CDD-005.73

Índices para catálogo sistemático:

1. Dados : Estruturas : Processamento de dados 005.73

Aline Graziele Benitez - Bibliotecária - CRB-1/3129

Sobre os Direitos Autorais

Sumário

1. Introdução

Este eBook foi elaborado no contexto do curso de capacitação Introdução à Análise de Dados em Saúde com Python ofertado pelo Centro de Inovação em Inteligência Artificial para Saúde. O curso tem como objetivo introduzir o estudo exploratório de bases de dados de saúde, com a utilização do Python. Neste eBook, procura-se apresentar uma abordagem preliminar à Ciência de Dados, que explora e descreve um conjunto de dados com técnicas da estatística descritiva e inferencial por meio da linguagem de programação Python.

O público alvo que pretende-se atingir caracteriza-se por profissionais de saúde, alunos de graduação e pós-graduação, docentes e pesquisadores da área das ciências da saúde, exatas ou demais interessados em utilizar os recursos computacionais para análise de bases de dados em saúde.

A linguagem Python tem se destacado como uma ferramenta poderosa para análise de dados em saúde, possuindo uma ampla gama de bibliotecas e recursos, o Python pode ser usado para limpar, processar, analisar e visualizar dados de saúde. Além disso, a comunidade de utilizadores da linguagem Python é muito colaborativa, com muitos recursos disponíveis, incluindo documentação, tutoriais e fóruns de suporte.

O conteúdo foi agrupado em conceitos iniciais sobre a utilização dos dados em saúde, introdução ao Python para utilização de dados, conceitos de limpeza e tratamento de dados, aplicação da estatística descritiva com os sumários estatísticos e gráficos, técnicas de amostragens, aplicação da estatística inferencial com os testes de hipótese, de associação, de médias, de medianas e correlações, além de explorar a estilização de gráficos.

1.1 Os dados em saúde

Os dados em saúde são extremamente importantes para a tomada de decisões e para a melhoria da qualidade dos cuidados de saúde. Eles podem incluir informações sobre pacientes, tratamentos, resultados clínicos, custos de saúde e outros aspectos relacionados à saúde. Com a crescente adoção de registros eletrônicos de saúde, sensores e dispositivos médicos, a quantidade de dados em saúde disponíveis tem aumentado rapidamente (1).

A análise de dados em saúde pode ajudar a identificar padrões, prever resultados e é fundamental para a melhoria da qualidade e eficiência dos serviços de saúde. No entanto, é importante garantir que esses dados sejam de alta qualidade, confiáveis e seguros. A privacidade dos pacientes também deve ser protegida, e os dados devem ser armazenados de maneira segura e em conformidade com as leis e regulamentações aplicáveis (2). Além disso, é necessário contar com profissionais capacitados para coletar, gerenciar e analisar esses dados de maneira adequada, considerando que a interpretação dos dados requer

conhecimento especializado em análise de dados e em saúde, para evitar conclusões equivocadas e garantir a tomada de decisões informadas e eficazes.

As bases de dados em saúde envolvem diversos tipos de dados e armazenam grandes quantidades de informação. Com o desenvolvimento das tecnologias da informação e com um consequente crescimento das bases de dados, em dimensão e em complexidade, surgem novas necessidades de análise de grande quantidade de dados (3). Entre os serviços de saúde, a área hospitalar ganha destaque por reunir os procedimentos de maior complexidade e custo, o que tem feito com que as bases de dados sobre as hospitalizações sejam o principal foco das mais diversas análises (4).

1.3 Dados, informação, conhecimento e sabedoria

Dados, informação e conhecimento são conceitos interrelacionados que desempenham papéis fundamentais em diversas áreas do conhecimento, incluindo a saúde (Figura 1). Embora muitas vezes sejam usados de forma intercambiável, cada um desses termos tem um significado específico e importante (5).

- **Dados:** os dados são observações brutas e objetivas, que geralmente são expressos em forma numérica, texto, som ou imagem. Eles podem ser coletados de diversas fontes, incluindo prontuário eletrônico, sensores, registros administrativos, pesquisas e questionários. Os dados, por si só, não possuem significado ou valor, mas podem ser organizados, analisados e interpretados para produzir informações.

- **Informação:** a informação é o resultado do processamento e organização dos dados de maneira a produzir significado e contexto. A informação é um conjunto de dados que são analisados, estruturados e contextualizados de forma a tornar-se úteis e compreensíveis. A informação tem um propósito definido, ou seja, fornece respostas para questões específicas.

- **Conhecimento:** o conhecimento é o resultado da interpretação da informação e de sua aplicação em determinado contexto, de modo que é gerado um entendimento que permite tomar decisões. O conhecimento está diretamente relacionado à capacidade de compreender, relacionar e contextualizar informações em uma perspectiva mais ampla, possibilitando a sua utilização para a resolução de problemas e tomada de decisão.
- **Sabedoria:** a sabedoria é o uso do conhecimento e da compreensão para tomada de decisão por meio de escolhas éticas e sábias, considerando as implicações de longo prazo e o impacto em outras pessoas e no mundo.

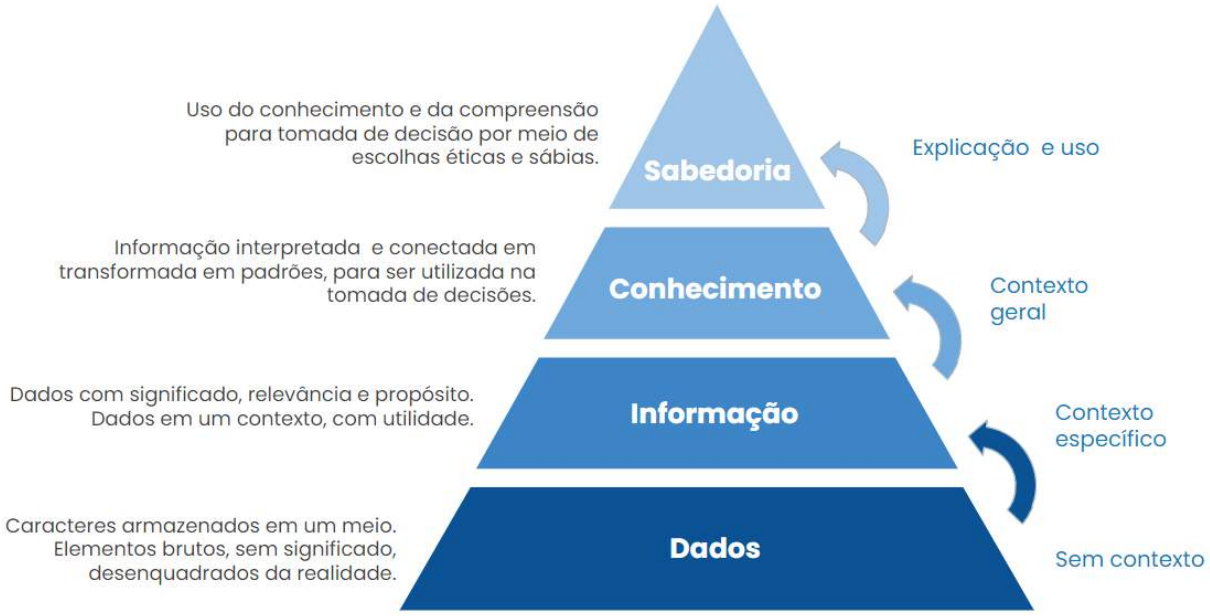


Figura 1: Pirâmide DIKW, adaptado de Shortliffe et al. 2006 (4).

Em resumo, os dados são a matéria-prima, a informação é o resultado do processamento dos dados, o conhecimento é a compreensão da informação e a sabedoria é o uso consciente do conhecimento para tomar decisões prudentes e responsáveis. Todos estes conceitos são essenciais para a tomada de decisão em diversas áreas, incluindo a saúde (4).

1.4 Qualidade de dados

Em todas as áreas, mas em particular na saúde, é importante contar com sistemas de informação robustos e eficientes, que sejam capazes de lidar com grandes volumes de dados, integrar diferentes fontes de informação e garantir a qualidade e veracidade dos dados. Além disso, é fundamental investir em treinamento e capacitação dos profissionais de saúde, para garantir que os dados sejam registrados de forma precisa e completa, e que sejam utilizados de forma apropriada para a tomada de decisão em saúde (1).

Se os dados utilizados em análises não tiverem qualidade, quer durante a prática clínica quanto na pesquisa científica, pode haver graves consequências para a sua saúde em geral. Dados de baixa qualidade podem levar a erros na tomada de decisão, que podem afetar a segurança e eficácia do tratamento, e até mesmo colocar a vida do paciente em risco. Por isso, é fundamental que os dados coletados e registrados nos prontuários sejam precisos, atualizados e confiáveis. O processo de validação e verificação dos dados também são etapas importantes para garantir sua qualidade e confiabilidade, além de ser importante investigar as causas dos erros, como corrigi-los e definir processos para evitá-los no futuro.

Vale a pena destacar que a qualidade dos dados, segundo alguns autores, é um conceito relativo, e está diretamente relacionada com o objetivo da informação desejada (6).

Quando se fala de qualidade dos dados, é comum encontrar na literatura tópicos relacionados aos desafios dos 4Vs na coleta de dados (Figura 2). Referem-se aos quatro aspectos principais que afetam a coleta, armazenamento, processamento e análise de grandes conjuntos de dados: volume, variedade, velocidade e veracidade. Enfrentar esses desafios requer o uso de tecnologias avançadas de coleta, armazenamento, processamento e análise de dados, bem como habilidades e técnicas adequadas de gerenciamento e análise de dados.

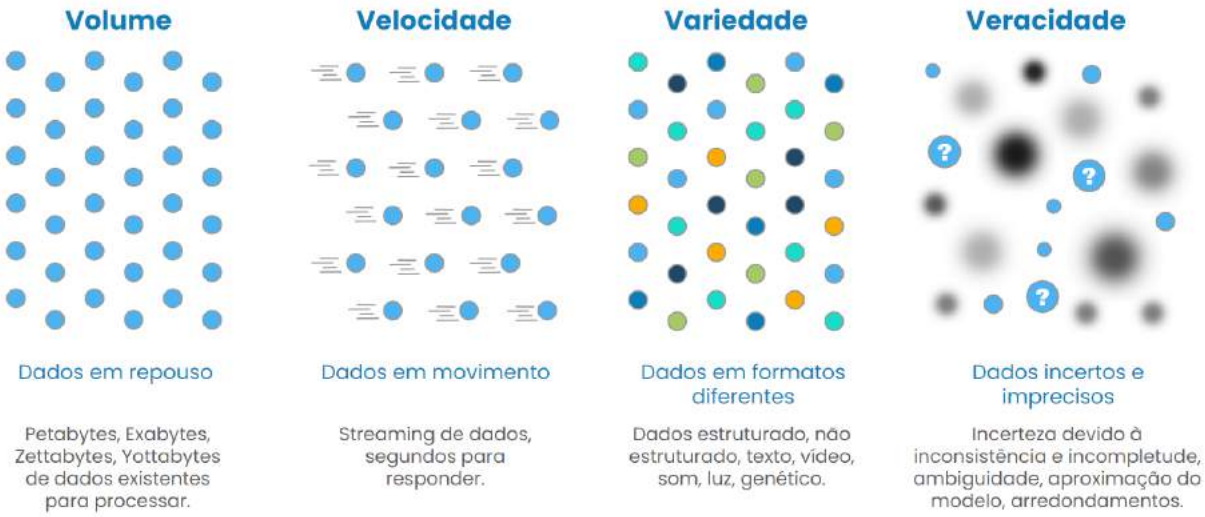


Figura 2: Desafios dos 4Vs da coleta e análise de dados. Fonte: adaptado de Peek, 2013 (7)

- **Volume:** o grande volume de dados gerados pode dificultar a coleta, armazenamento, processamento e análise dos dados. É necessário ter sistemas capazes de lidar com grandes volumes de dados para garantir que as informações sejam registradas de forma completa e precisa.
- **Velocidade:** a velocidade com que os dados são gerados também é um desafio. Em alguns casos, como em monitoramento de pacientes em tempo real, é necessário que os dados sejam registrados e processados imediatamente para garantir um atendimento eficiente.
- **Variedade:** a variedade de fontes de dados, como informações clínicas, imagens, dados de sensores e informações de redes sociais, também pode dificultar a coleta e análise dos dados. É importante ter sistemas que sejam capazes de lidar com diferentes tipos de dados e integrá-los de forma apropriada.
- **Veracidade:** a veracidade dos dados é fundamental para garantir a sua qualidade e confiabilidade. A coleta de dados deve ser feita de forma precisa e confiável, evitando erros e garantindo a validade dos dados.

Em resumo, a qualidade dos dados é um fator crítico para a tomada de decisão em saúde, e é importante garantir que os dados coletados e registrados nos prontuários sejam precisos, atualizados e confiáveis, para proporcionar um atendimento seguro e eficaz aos pacientes. Assim, os conceitos de relevância, consistência, reprodutibilidade, acurácia, completude e disponibilidade são especialmente relevantes para os dados em saúde (Figura 3), pois podem ter impactos significativos na tomada de decisões clínicas, na pesquisa médica e na saúde pública em geral.

Figura 3: Dimensões de qualidade de dados

- **Relevância:** a relevância se refere à importância e utilidade dos dados para o problema ou questão que está sendo investigado, dados que fornecem informações significativas e úteis para a tomada de decisões.
- **Consistência:** a consistência se refere à uniformidade dos dados e à ausência de contradições ou inconsistências entre eles.
- **Reprodutibilidade:** a reprodutibilidade se refere à capacidade de replicar ou reproduzir os resultados e análises a partir dos mesmos dados.
- **Acurácia:** a acurácia se refere à precisão e confiabilidade dos dados, ou seja, quão bem eles representam a realidade ou o fenômeno estudado.
- **Completude:** a completude se refere ao grau em que todos os dados relevantes para a análise estão disponíveis e foram coletados.
- **Disponibilidade:** a disponibilidade se refere à facilidade de acesso aos dados, garantindo que sejam acessíveis e utilizáveis por outros profissionais e pesquisadores, quando necessário e com autorização ética para fazê-lo.

Todos esses aspectos são importantes para garantir a qualidade e utilidade dos dados em pesquisas e tomada de decisões baseadas em dados.

1.5 Planejamento do instrumento de coleta de dados

Para planejar e construir um bom instrumento de coleta de dados para pesquisas em saúde, é importante considerar os conceitos abordados nos tópicos anteriores e seguir alguns passos. Para isso, deixamos aqui algumas orientações que podem ajudar neste planejamento:

Definir o objetivo da coleta de dados: Antes de começar a construir o instrumento, é importante definir claramente o objetivo da coleta de dados. Por exemplo, o objetivo pode ser avaliar a eficácia de um tratamento ou compreender melhor um grupo populacional específico.

Selecionar a população-alvo: Definir a população-alvo para a coleta de dados é fundamental para garantir que as informações coletadas sejam relevantes e úteis para um determinado perfil populacional em questão.

Selecionar as variáveis a serem coletadas: As variáveis são os dados específicos que serão coletados para atender ao objetivo da coleta de dados. É importante selecionar as variáveis com base em sua relevância e utilidade para a análise dos dados, também é importante especificar a granularidade das variáveis, dando preferência pela coleta do dado primário. Recomendamos que você crie um dicionário contendo todas as variáveis e suas especificações de tipos, categorias e limites.

Escolher o método de coleta de dados: Existem várias formas de coletar dados em saúde, como questionários, entrevistas e registros médicos. É importante escolher o método de coleta de dados mais adequado para a população-alvo e as variáveis selecionadas.

Desenvolver o instrumento de coleta de dados: Depois de selecionar as variáveis e o método de coleta de dados, é possível desenvolver o instrumento de coleta de dados. O instrumento deve ser claro, conciso e fácil de usar para garantir que as informações coletadas sejam precisas e consistentes.

Testar o instrumento de coleta de dados: É importante testar o instrumento de coleta de dados antes de usá-lo para coletar dados em grande escala. O teste pode ajudar a identificar problemas e garantir que o instrumento esteja funcionando corretamente.

Validar o instrumento de coleta de dados: A validação do instrumento de coleta de dados é um processo de avaliação para garantir que o instrumento esteja medindo o que se propõe a medir. A validação pode incluir a análise estatística dos dados coletados e a comparação com outras fontes de informação para garantir que as informações coletadas sejam precisas e confiáveis.

Considerar os aspectos éticos: A ética é um aspecto crucial na coleta de dados em saúde, pois envolve questões de privacidade, consentimento informado e uso adequado dos dados.

É importante que os dados sejam coletados com o consentimento informado dos pacientes ou indivíduos, garantindo que eles compreendam o propósito da coleta de dados, como os dados serão utilizados e quem terá acesso a eles. As normas éticas são estabelecidas por organizações como o Comitê de Ética em Pesquisa (CEP) e a Comissão Nacional de Ética em Pesquisa (CONEP) no Brasil

Garantir a segurança dos dados: A segurança dos dados é importante para proteger a privacidade e confidencialidade dos pacientes e indivíduos, garantindo que suas informações não sejam divulgadas sem autorização ou comprometidas por violações de segurança, além disso, é importante garantir que os dados sejam usados apenas para fins legítimos. No caso de sistemas eletrônicos podem ser utilizados diversos procedimentos de segurança, como a criptografia de dados, sistemas de autenticação de usuários e backups regulares, entre outros. No caso de coleta de dados para pesquisa em formulários ou questionários, em papel ou digital, recomenda-se não coletar dados de identificação pessoal (nome, contatos e endereços), em casos que essas informações sejam importantes para a pesquisa, recomenda-se que sejam coletadas em um formulário separado dos dados clínicos, e ligados por um identificador (código) único.

2. Introdução ao Python para análise de dados

2.1 Introdução ao Python

Python é uma linguagem de programação de alto nível e de propósito geral, criada com o objetivo de ser uma linguagem fácil para implementação de algoritmos, possuindo uma sintaxe simples e intuitiva. Python é amplamente utilizado em diversas áreas, incluindo desenvolvimento web, ciência de dados e inteligência artificial (8). Além de ser uma linguagem de programação popular entre iniciantes devido à sua facilidade de aprendizado e ampla comunidade on-line, ela também possui uma comunidade muito ativa, fornecendo uma grande quantidade de recursos, como documentação, tutoriais e fóruns de discussão.

Ao utilizar a linguagem Python pela primeira vez é fundamental ter uma boa compreensão da sintaxe e do funcionamento da linguagem antes de começar a escrever códigos complexos. Em geral, a sintaxe Python é bastante clara e intuitiva, seguindo algumas convenções comuns a muitas outras linguagens de programação, como:

- **Indentação:** é chamado de indentação o recuo à direita no início das linhas do código, serve para indicar o início e o final de um bloco de código. O Python usa indentação, em vez de usar chaves ou outros delimitadores. Isso significa que a indentação deve ser consistente em todo o código, geralmente com quatro espaços.
- **Comentários:** os comentários são iniciados por "#" e continuam até o fim da linha. Não existe um método padrão para comentar várias linhas de código em Python.
- **Variáveis:** as variáveis são criadas assim que são atribuídas um valor, não precisam ser declaradas explícitas com tipos e sua atribuição é feita com o sinal "=". Tipos explícitos: int, float, complex, object (strings), bool, list, tuple e dict.
- **Operadores:** Python suporta os operadores matemáticos comuns, incluindo +, -, *, /, % (módulo), ** (potência) e muitos outros.
- **Controle de fluxo:** a linguagem suporta as estruturas de controle de fluxo comuns, incluindo if, elif, else, for e while.
- **Funções:** as funções são definidas usando a palavra-chave "def", seguida pelo nome da função e parênteses.

Essas são apenas algumas das convenções de sintaxe do Python, mas há muito mais para aprender. A melhor maneira de aprender a sintaxe é experimentando com exemplos simples e assim, construindo os seus conhecimentos. A documentação oficial do Python (9) também é uma ótima fonte para consulta quando tiver dúvidas sobre a sintaxe de algum comando ou construtores. Há muitas informações on-line e gratuitas disponíveis, incluindo documentação oficial, tutoriais e exemplos, que podem ajudá-lo no aprendizado.

2.1.1 Regras para nomes das variáveis

Em Python, os nomes de variáveis devem obedecer algumas regras simples:

- Devem sempre começar com uma letra minúscula ou underline;
- Deve ser uma sequência de letras (a → z, A → Z) e números (0 → 9);
- Apenas letras comuns são permitidas;
- Não são permitidos acentuação, cedilhas, espaços, caracteres especiais
    - Por exemplo: $, #, @, !, %, entre outros;
- Pode ser usado o caractere _ (sublinhado/underline);
- Palavras reservadas da linguagem não são permitidas
    - Por exemplo: print, True, False, if, else, min, max, entre outras);

2.1.2 Variáveis e comentários

```
# Criar variáveis
x = 10
# Mostrar o conteúdo
print(x)
# Somar variáveis e números
y = x + 30
# Criar listas de números, variáveis e textos
lista = [20, 18, 48, 62]
# Mostrar o conteúdo da 1ª posição da lista
lista[0]
# Mostrar o conteúdo da 3ª posição da lista
lista[2]
# Somar o 1º com o 2º elemento da lista
soma = lista[0] + lista[1]
# Mostrar o resultado
print(soma)
```

2.1.3 Operações e funções aritméticas

```
# Adição
valor = 50 + 30
# Subtração
valor = 50 - 30
# Divisão
valor = 50 / 10
# Multiplicação
valor = 3 * 2
# Exponenciação
valor = 3 ** 2
```

```
# Função para calcular o valor absoluto
abs(-15)
# Função para exponenciação
x = pow(2, 3)
# Função para radiciação
x = sqrt(9)
# Função para valor mínimo de uma lista
minimo = min(4, 18, 6)
# Função para valor máximo de uma lista
maximo = max(45, 67, 23)
# As funções também podem ser usadas combinada com as listas
max(lista)
```

2.1.4 Operadores relacionais

```
# Operador de igualdade - comparando variáveis
x == y
# Operador de igualdade - comparando uma variável com um número
x == 20
# Operador de diferença
x != y
# Operador de maior
x > y
# Operador de menor
x < y
# Operador de maior ou igual
x >= y
# Operador de menor ou igual
x <= y
```

2.1.5 Variáveis do tipo texto

```
# Criar variáveis do tipo texto (string)
nome = 'Juliano'
# Mostrar
print(nome)
# Variáveis de texto
sobrenome = 'Gaspar'
# Mostrar duas variáveis
print(nome, sobrenome)
# Mostrar combinações de textos e variáveis de textos
print('Nome do professor:', nome, sobrenome)
# Mostrar texto e variáveis numéricas
print('Idade:', y)
```

2.2 Introdução ao Google Colaboratory

O Google Colaboratory (10), ou simplesmente Colab, é uma ferramenta de desenvolvimento de código gratuita que roda no navegador (nuvem) e permite escrever, executar e compartilhar códigos em Python. Ele é baseado no Jupyter Notebook e fornece uma interface de usuário amigável para escrever e executar códigos, bem como incluir texto, imagens, gráficos e equações matemáticas.

O Colab é especialmente útil para trabalhar com dados, análises estatísticas, aprendizado de máquina, permitindo que os usuários carreguem arquivos de sua máquina local ou usem arquivos armazenados em outros serviços, como o Google Drive. O Colab é uma ótima opção para quem está começando a trabalhar com Python e ciência de dados, pois não requer configuração local e possui uma vasta documentação on-line para consulta.

Para acessar o Google Colab: https://colab.research.google.com

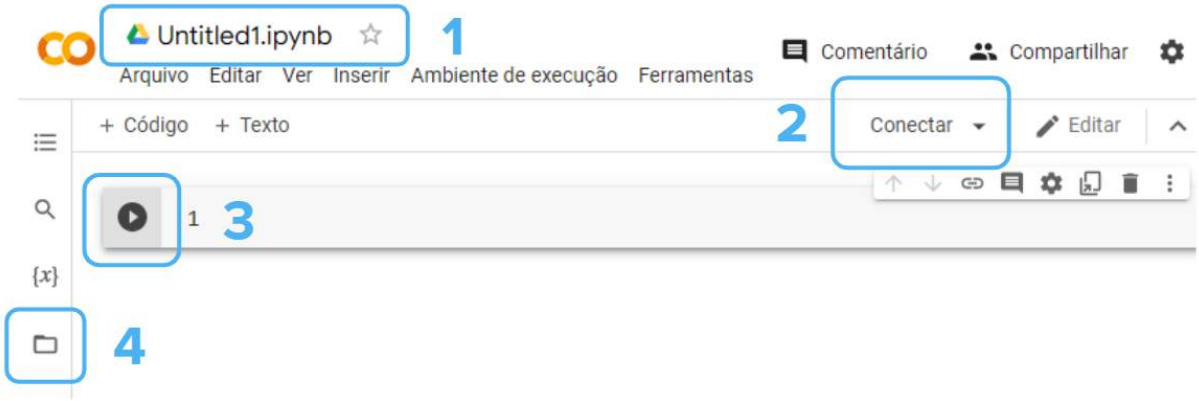


Figura 4: Página inicial do Google Colab

Na figura 4, pode ser visto o nome do arquivo (1) que será salvo automaticamente na conta do Google Drive associada, o botão e indicador de conexão ao servidor (2), o botão de execução do bloco de código digitado (3) e o gerenciador de arquivos (4) que permite entre outras coisas acessar o Google Drive do usuário.

2.3 Bibliotecas em Python

As bibliotecas em Python são conjuntos de módulos, classes e funções pré-definidas que permitem acesso a diversos tipos de recursos, desde o processamento de dados, a criação de gráficos, a integração com outras aplicações, entre outras. Elas são projetadas para tornar os códigos mais eficientes e flexíveis, permitindo que novas funcionalidades sejam rapidamente adicionadas com pouco ou nenhum código.

2.3.1 Biblioteca Pandas

A biblioteca Pandas (11) é uma biblioteca de software de código aberto e gratuito para manipulação e análise de dados em Python. É projetada para fornecer uma estrutura de dados flexível e de alto desempenho para trabalhar dados numéricos e de texto em Python. Ela é considerada uma das bibliotecas mais poderosas e populares para análise de dados em Python.

As principais características da biblioteca Pandas são:

- **Tipos de dados:** permite trabalhar com dados estruturados e não estruturados;
- **Estruturas de dados:** possui estruturas de dados flexíveis, como séries de dados, dataframes e matrizes:
    - **Séries:** São arrays unidimensionais rotulados capazes de armazenar qualquer tipo de dado.
    - **DataFrames:** São estruturas de dados bidimensionais semelhantes a tabelas de banco de dados ou planilhas do Excel.
- **Leitura e escrita de dados:** Pandas fornece funcionalidades para ler e escrever dados de diferentes fontes, incluindo arquivos CSV, Excel, SQL, JSON e HTML;
- **Manipulação de dados:** fornece ferramentas para filtrar, selecionar, renomear, agregar, ordenar e organizar os dados de acordo com os requisitos do usuário;
- **Limpeza de dados:** a biblioteca possui diversas funções para tratar dados faltantes e remover duplicatas;
- **Análise de dados:** permite que os usuários explorem os dados rapidamente para descobrir padrões e realizar análises estatísticas sofisticadas;
- **Visualização de dados:** oferece funcionalidades para criar tabelas, gráficos e diagramas para visualizar os dados de forma clara e intuitiva.

Exemplo: importando a biblioteca

```
# Importar a biblioteca
import pandas as pd
```

	DT_INTERNACAO	DT_ALTA	DURACAO_INT	GESTACOES	PARTOS	IG_OBSTETRA	IG_PEDIATRA
795	2014-05-05	2014-05-07	2.040278	2	0.0	41.0	41.0
729	2014-01-07	2014-01-09	1.995833	2	1.0	27.0	27.0
1514	2014-05-08	2014-05-12	4.088889	6	4.0	39.0	39.0
1226	2013-10-02	2013-10-05	2.695139	2	1.0	38.0	38.0

Figura 5: Exemplo de um DataSet com dados importados utilizando a biblioteca Pandas

2.3.2 Biblioteca Matplotlib

A biblioteca Matplotlib (12) é uma das principais bibliotecas de visualização de dados em Python. Ela permite criar gráficos e visualizações de alta qualidade de forma simples e eficiente, permitindo a criação de gráficos de linha, gráficos de barra, gráficos de dispersão, histogramas, entre outros tipos de gráficos. Além disso, a Matplotlib permite personalizar os gráficos com cores, legendas, títulos e outros elementos, tornando possível criar visualizações que sejam adequadas para cada tipo de dado e análise (Figura 6).

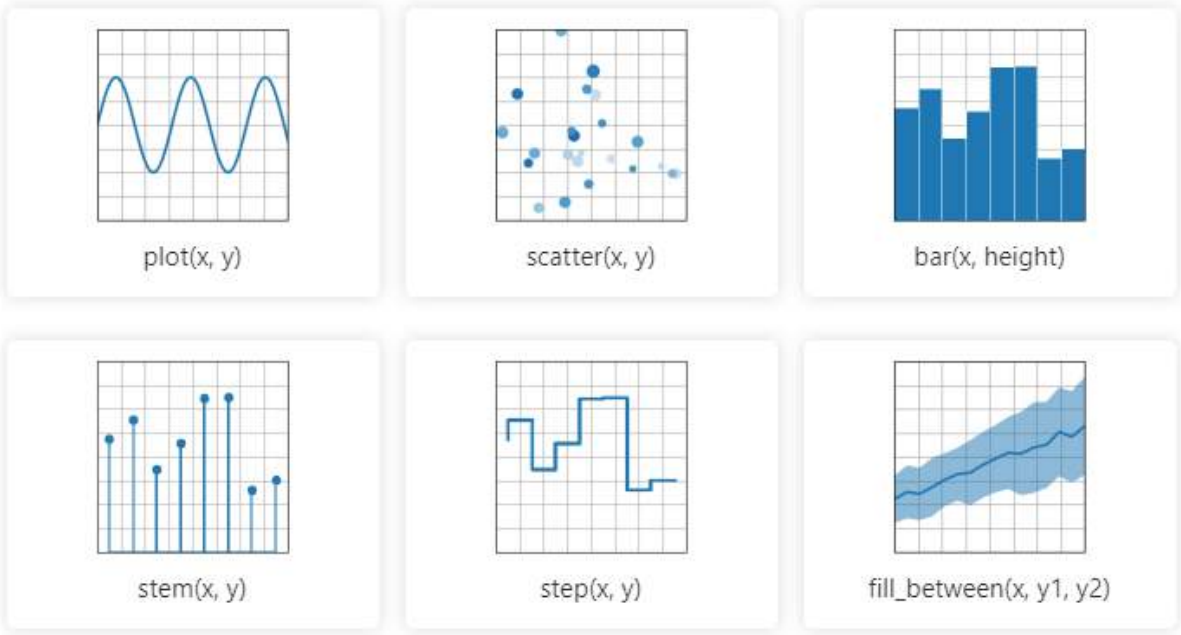


Figura 6: Exemplos de gráficos que podem ser feitos com a Matplotlib
Fonte: Matplotlib (2023). Disponível em: https://matplotlib.org

A biblioteca Matplotlib possui as seguintes características:

- Possibilidade de customização completa de cada elemento do gráfico, incluindo cores, tamanhos, fontes e posicionamento;
- Suporte para múltiplos subplots em um único gráfico, permitindo a visualização de diferentes dados lado a lado;
- Compatibilidade com diversos formatos de arquivo para salvar as visualizações, como PNG, PDF, SVG, entre outros.

Exemplo: importando a biblioteca

```
# Importar a biblioteca
import matplotlib.pyplot as plt
```

2.3.3 Biblioteca NumPy

A biblioteca NumPy (13) é uma biblioteca de software de código aberto para a computação científica, para realizar cálculos numéricos, como cálculos matriciais e álgebra linear. Ela é útil para trabalhar com dados estruturados, como tabelas, oferecendo suporte para trabalhar com estruturas de dados flexíveis e multidimensionais, como matrizes e arrays, permitindo acesso rápido e eficiente aos dados (Figura 7).

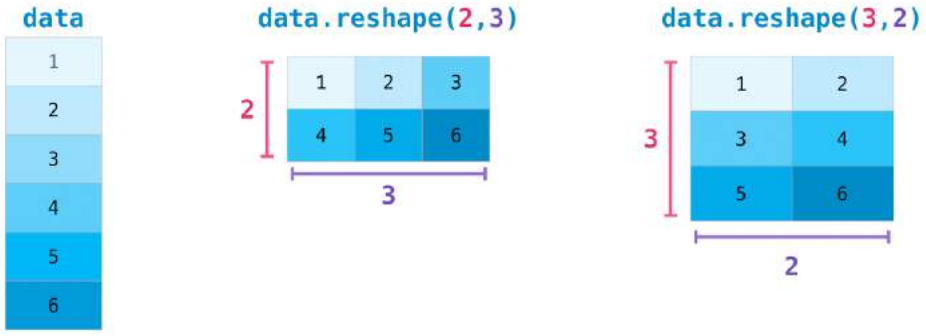


Figura 7: Exemplos de operações em matrizes com a NumPy
Fonte: NumPy(2023). Disponível em: https://numpy.org

As principais características da biblioteca Numpy são:

- Suporte para estruturas de dados flexíveis e multidimensionais (matrizes e arrays);
- Funções de álgebra linear para realizar cálculos numéricos;
- Funções para manipular dados estruturados, como tabelas;
- Funções de plotagem para visualizar os dados.

Exemplo: importando a biblioteca

```
# Importar a biblioteca
import numpy as np
```

2.3.4 Biblioteca Seaborn

A biblioteca Seaborn (14) é uma biblioteca de visualização de dados de alto nível para Python. Ela oferece uma série de gráficos estatísticos e de visualização que permitem que os usuários explorem e descubram padrões nos dados de forma rápida e intuitiva. A biblioteca também oferece algumas funcionalidades avançadas, como maneiras de personalizar os gráficos para torná-los mais intuitivos (Figura 8).

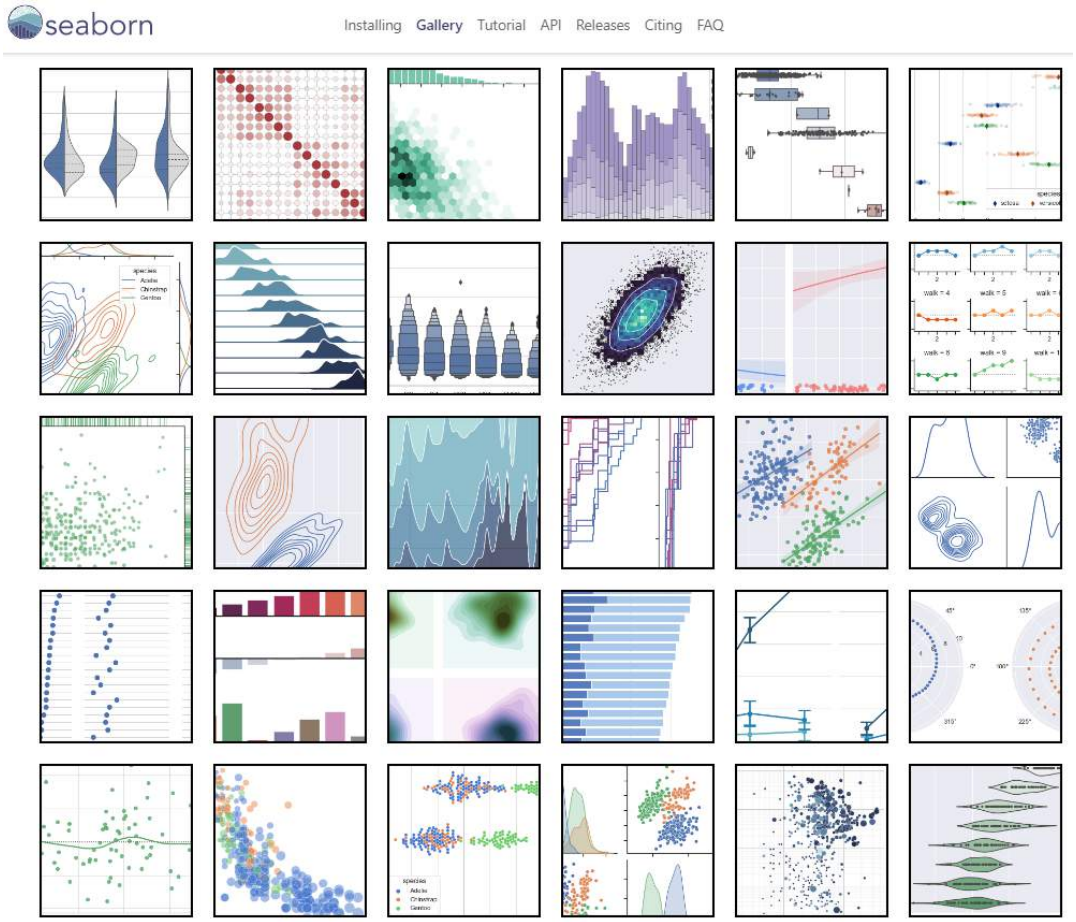


Figura 8: Exemplos de gráficos que podem ser feitos com a Seaborn
Fonte: Seaborn (2023). Disponível em: https://seaborn.pydata.org

A biblioteca Seaborn possui as seguintes características:

- Gráficos estatísticos e de visualização para explorar e descobrir padrões nos dados;
- Funcionalidades avançadas de personalização dos gráficos para torná-los mais intuitivos;
- Funcionalidades para conectar gráficos e compor imagens complexas.

Exemplo: importando a biblioteca

```
# Importar a biblioteca
import seaborn as sns
```

3. Manipulação, limpeza e tratamento de dados

3.1 Manipulação de dados

Manipulação de dados é o processo de transformar, organizar, limpar e re-estruturar os dados para que possam ser usados para análise, envolvendo tarefas como filtragem, seleção, agregação, ordenação, renomeação e junção de dados. O objetivo da manipulação de dados é preparar os dados para que seja realizada quer uma análise estatística como também para aplicação modelos de aprendizado para descoberta de padrões ocultos, não revelados pela análise descritiva (15).

Uma das formas de se manipular dados em Python é através da biblioteca Pandas, que oferece um conjunto de recursos para auxiliar nas mais diversas necessidades de manipulação de dados. A estrutura de dados primária da biblioteca Pandas é o objeto DataFrame, que por sua vez, é composto por um conjunto de Series (caracterizado por uma coluna de um DataFrame).

3.1.1 Importando dados para o DataFrames

Um DataFrame da biblioteca Pandas é uma estrutura de dados tabular bidimensional (matriz) que armazena dados e é projetado para ajudar os usuários a trabalhar com dados de forma prática e eficiente. Os DataFrames permitem que os usuários adicionem, modifiquem, filtrem, agreguem e realizem outras operações em dados estruturados. Uma das características principais é permitir que os usuários importem dados de diversos arquivos e os convertam em DataFrames facilitando e uniformizando a manipulação de diversas fontes de dados.

Exemplo: importação de dados para DataFrames

```
# Importar a biblioteca Pandas
import pandas as pd

# Ler os dados do arquivo e colocar os dados no objeto DataFrame

# Para ler os dados de um arquivo CSV
dados = pd.read_csv('pacientes.csv')
# Para ler os dados de um arquivo JSON
dados = pd.read_json('pacientes.json')
# Para ler os dados de um arquivo XML
dados = pd.read_xml('pacientes.XML')
# Para ler os dados de um arquivo Excel
dados = pd.read_excel('pacientes.xlsx')
```

Exemplos: ajustes ao importar arquivos CSV (com separadores)

```
# Importar o arquivo e forçar usar um separador específico (;)
dados = pd.read_csv(arquivo, sep=';')

# Importar o arquivo e forçar que pule na importação a primeira
linha (ou de um conjunto de linhas)
dados = pd.read_csv(arquivo, sep=';', skiprows=1)

# Importar o arquivo e forçar usar um outro tipo de codificação
# Codificações comuns = latin1 / utf8 / cp1252
dados = pd.read_csv(arquivo, sep=';', skiprows=1, encoding='latin1')

# Importar o arquivo e ler apenas algumas colunas
dados = pd.read_csv(arquivo, usecols=[0, 1, 2])

# Para excluir uma coluna da tabela, considere:
# axis = 0 para indicar uma linha
# axis = 1 para indicar uma coluna
dados = dados.drop('UBS', axis=1)
```

Exemplo: mostrar o conteúdo de DataFrame

```
# Mostrar o conteúdo do DataFrame "dados"
dados
```

	DT_INTERNACAO	DT_ALTA	DURACAO_INT	GESTACOES	PARTOS	IG_OBSTETRA	IG_PEDIATRA	ALTO_RISCO
0	2014-01-20	2014-01-21	0.768750	2	1.0	38.0	38.0	sim
1	2014-05-21	2014-05-22	0.773611	1	0.0	36.0	36.0	sim
2	2014-04-13	2014-04-14	0.798611	2	1.0	39.0	39.0	não
3	2013-12-04	2013-12-05	0.807639	2	1.0	41.0	41.0	não
4	2013-12-05	2013-12-06	0.815972	1	0.0	36.0	36.0	não
...	...	...	...	...	...	...	...	...

Figura 9: Exemplo de dados importados para o DataFrame

3.1.2 Acessando os dados de uma Series

Um objeto do tipo Series da biblioteca Pandas é uma estrutura de dados unidimensional que armazena dados de forma similar a uma lista (pode ser chamada de vetores). Pode-se comparar uma Series à uma coluna de dados em uma tabela, porém contém intrinsecamente

diversas funcionalidades que podem ser aplicadas diretamente ao conjunto de dados da respectiva variável.

Exemplo de acesso aos dados de uma Series

```
# Acessar os dados de uma Series (coluna)

# Se o título da coluna não tiver espaços ou caracteres especiais
# Pode-se usar a forma simples:
dados.TIPO_PARTO

# Caso contrário, use a forma padrão:
dados['TIPO_PARTO']
```

3.2 Limpeza e tratamento de dados

Ao realizar a limpeza e o tratamento de dados, garantimos que eles estejam em um formato adequado para análise ou modelagem, reduzindo as chances de erro e aumentando as chances de conclusões mais confiáveis. O processo de limpeza de dados envolve tarefas como remover dados ausentes, substituir dados ausentes ou inconsistentes, aplicar transformações em dados numéricos, converter dados entre diferentes tipos de dados e até mesmo normalizar dados (16). O objetivo da limpeza de dados é tornar os dados mais confiáveis e consistentes para que possam ser usados para análise, ou seja com mais qualidade. Cabe destacar que este processo demanda um tempo importante em toda a análise de dados, recomenda-se destinar de 20% a 40% do tempo do projeto de análise para esta etapa.

3.2.1 Dicionário de dados

Um dicionário de dados é um documento que descreve os dados contidos em um banco de dados ou em outro conjunto de dados. Ele contém informações sobre os nomes dos campos, o tipo de dado de cada campo, a definição de cada campo, os intervalos possíveis e/ou categorias das variáveis e outras informações relevantes. O dicionário de dados é uma ferramenta importante para garantir a qualidade dos dados e a consistência das informações, além de ser útil para a documentação do banco de dados.

Variável	Tipo	Limites e descrição das categorias
DT_INTERNACAO	Data	Data da internação
DT_ALTA	Data	Data da alta
GESTACOES	Numérica	Número de gestações
PARTOS	Numérica	Número de partos
IG_OBSTETRA	Numérica	Idade gestacional dada pelo obstetra

ALTO_RISCO	Categórica	0 = Não, 1 = Sim
TIPO_PARTO	Categórica	1 = parto normal, 2 parto cesáreo
HIPERTENSAO	Categórica	0 = Não, 1 = Sim
CESAREAS_PREVIAS	Numérica	Número de cesáreas prévias

Tabela 1: Exemplo de um dicionário de dados

Ao planejar uma coleta de dados, procure antes criar o dicionário de dados do conjunto que planeja realizar a coleta. Em casos de utilização de dados secundários, já coletadas por algum sistema ou disponibilizadas em fontes de dados públicas como o DataSUS, procure pelos arquivos de dicionário de dados, que podem ser descritos como: catálogo de dados, dicionário de metadados, especificação de dados, glossário de dados, dicionário de informações, de campos, de variáveis, de elementos de dados, de estrutura de dados ou ainda de dicionário técnico de dados.

3.2.2 Identificar e compreender os dados

Antes de iniciar o processo de tratamento e limpeza dos dados, é importante se familiarizar com os dados e entender o que eles representam, ou seja, entender o contexto dos dados, como eles foram coletados, se existem documentação ou protocolos a respeito das variáveis, limites, faixas de valores, identificar categorias, identificar como os valores ausentes (omissos, null, NaN) são registrados, entre outros.

Exemplo: mostrar as colunas do DataFrame

```
# Mostrar uma lista com os nomes das colunas
dados.columns
```

```
Index(['DT_INTERNACAO', 'DT_ALTA', 'DURACAO_INT', 'GESTACOES', 'PARTOS',
       'IG_OBSTETRA', 'IG_PEDIATRA', 'ALTO_RISCO', 'TIPO_PARTO', 'HIPERTENSAO',
       'GEMELAR', 'CESAREAS_PREVIAS', 'EPISIOTOMIA', 'ANALGESIA', 'FORCEPS',
       'CM_LACERACAO_CANAL', 'CM_UTI', 'CM_INFECCAO', 'CM_NEARMISS',
       'CM_TRANSFUSAO', 'VIVO', 'SEXO', 'PESO_NASCER', 'APGAR1', 'APGAR5',
       'UTI_RN', 'LIGADURACORDAO', 'LC_MOTIVO', 'EQUIPAMENTO_A',
       'EQUIPAMENTO_B', 'UBS', 'BAIRRO', 'LATITUDE', 'LONGITUDE'],
      dtype='object')
```

Figura 10: Lista de colunas do DataFrame

Exemplo: mostrar as informações básicas sobre os dados importados

```
# Mostrar as informações das variáveis importadas
dados.info()
```

```
RangeIndex: 1709 entries, 0 to 1708
Data columns (total 34 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   DT_INTERNACAO       1709 non-null   datetime64[ns]
 1   DT_ALTA             1709 non-null   datetime64[ns]
 2   DURACAO_INT         1709 non-null   float64
 3   GESTACOES           1709 non-null   int64
 4   PARTOS              1708 non-null   float64
 5   IG_OBSTETRA         1689 non-null   float64
 6   IG_PEDIATRA         1706 non-null   float64
 7   ALTO_RISCO          1709 non-null   object
 8   TIPO_PARTO          1709 non-null   int64
```

Figura 11: Exemplo de dados importados para o DataFrame

3.2.3 Inspeção dos dados

Após compreender os dados, é importante inspecionar os dados mais de perto, e assim identificar dados ausentes, inconsistências ou até mesmo erros. Pode-se visualizar os dados importados, analisar as primeiras e últimas linhas do arquivo, verificar as estatísticas descritivas dos valores numéricos, conferir formatos de datas, campos de textos livres ou categóricos, em caso de campos de textos se existem caracteres especiais não formatados ou espaços antes ou no final dos textos.

Exemplos: visualizando os dados

```
# Mostrar as 5 primeiras linhas e 5 últimas linhas
dados

# Mostrar uma amostra aleatória de 10 linhas
dados.sample(10)

# Mostrar o início (topo) da lista de dados
dados.head()

# Mostrar o fim (cauda) da lista de dados
dados.tail()
```

Exemplos: visualizando os dados duplicados

```
# Verificar se existem dados duplicados

# keep = False ==> mostra os duplicados
# keep = 'first' ==> mostra o 1° duplicado
# keep = 'last' ==> mostra o último duplicado

# Fazer uma busca por duplicados em alguma coluna específica
dados[dados.duplicated(['DT_INTERNACAO'], keep=False)]

# Fazer uma busca por duplicados em um conjunto de colunas
```

```
dados[dados.duplicated(['DT_INTERNACAO', 'DURACAO_INT',
'IG_OBSTETRA', 'PESO_NASCER'], keep=False)]

# Para apagar linhas duplicadas

# Pode-se apagar as linhas onde todas as colunas forem iguais
dados.drop_duplicates()

# Pode-se apagar as linhas onde um conjunto de colunas são iguais
dados.drop_duplicates(subset=['DT_INTERNACAO', 'DURACAO_INT',
'BAIRRO', 'PESO_NASCER'], keep='first', inplace=True)
```

Exemplos: visualizando e contando as categorias de uma coluna

```
# Contando a quantidade de casos por categoria em uma variável
dados.TIPO_PARTO.value_counts()
```

```
Parto Normal     1056
Parto Cesáreo     652
Name: TIPO_PARTO, dtype: int64
```

Figura 12: Contagem de registros por tipo de categoria

3.2.4 Tratamento de dados ausentes

Os dados ausentes devem ser tratados de alguma forma e o tratamento pode variar conforme o contexto dos dados e tipo de análise que se deseja fazer. No Python, os dados ausentes são representados com **NaN** (*Not a Number*) e o conceito de dados omissos é comumente chamado de ***null*** (nulo). Por exemplo, os dados ausentes podem ser substituídos por médias, medianas, moda, definidos como nulos ou removidos da análise. São decisões que cabem ao analista de dados decidir para cada uma das variáveis com dados ausentes, por exemplo, podem ser utilizadas estratégias diferentes para cada variável da base de dados.

Exemplos: tratamentos de valores ausentes

```
# Contar a quantidade de casos ausentes em cada variável
dados.isnull().sum()

# Estratégia 1
# Apagar as linhas com qualquer valor nulo em qualquer variável
dados_sem_nulos = dados.dropna(how = 'any')

# Estratégia 2
# Apagar as linhas com todas as variáveis com valores nulos
dados_sem_linhas_vazias = dados.dropna(how = 'all')

# Estratégia 3
```

```
# Apagar as linhas que tenham variáveis específicas com nulos
dados_validos = dados.dropna(how = 'any', subset=['IG_OBSTETRA',
'PESO_NASCER'])

# Estratégia 4
# Preencher os valores ausentes com a MEDIANA da variável
# Calcular a mediana
mediana = dados.APGAR5.median()
# Preenche os nulos com a mediana
dados.APGAR5.fillna(mediana, inplace=True)

# Estratégia 5
# Preencher os valores ausentes (NULL) com a MÉDIA da variável
# Calcular a média
media = np.mean(dados.EQUIPAMENTO_B)
# Preenche os nulos com a média
dados.EQUIPAMENTO_B.fillna(media, inplace=True)
```

Em algumas análises, pode ser necessário converter um valor numérico específico para um valor omisso, para que não seja incluído em análises futuras. São estratégias muito específicas e devem estar coerentes com as decisões tomadas durante toda a análise de dados. Vejamos dois exemplos onde este tipo de estratégia pode ser adotada:

Exemplo 1: Recodificar valores para que sejam considerados ausentes

```
# Neste exemplo a variável SEXO apresenta alguns casos com o valor
ZERO (0), que no dicionário de dados significa valor não informado.
Caso registros com ZERO, não sejam convertidos em ausentes, o valor
ZERO será considerado como uma categoria válida em análises futuras.

# Preencher com nulo (nan) o valor 0
dados.SEXO.replace(0, np.nan, inplace=True)
```

Exemplo 2: Recodificar valores que podem ser considerados ausentes

```
# Neste exemplo a variável SEXO apresenta alguns casos com o valor
TRÊS (3), que no dicionário de dados significa recém-nascidos que
nasceram com sexo indeterminado. Ao avaliar a quantidade de
registros com valor 3 observou-se que são eventos raros. Assim, para
esta análise os analistas de dados decidiram considerar os registros
com valor SEXO igual a 3 como valores ausentes para a análise
subsequentes.

# Preencher com nulo (nan) o valor 3
dados.SEXO.replace(3, np.nan, inplace=True)
```

3.2.5 Identificar e lidar com outliers

Os *outliers* são dados que estão muito distantes do padrão geral. Eles podem ser causados por erros de entrada de dados ou podem representar dados reais, por isso, é importante identificar e lidar com outliers para evitar problemas com a análise. O tratamento dos *outliers* podem seguir (ou não) as mesmas decisões tomadas para tratar os dados ausentes.

A identificação de outliers é uma tarefa complexa e exige compreensão do contexto das variáveis. É importante escolher métodos adequados para realizar a identificação dos casos extremos em cada contexto. Uma forma muito básica é observar os valores extremos mínimos e máximos das variáveis numéricas e confrontá-los com a mediana ou média, procurando analisar se estão dentro do esperado para cada contexto.

Exemplo: Para extrair os valores descritivos para as variáveis numéricas

```
# Resumo descritivo das colunas numéricas
dados.describe()
```

	DURACAO_INT	GESTACOES	PARTOS	IG_OBSTETRA	IG_PEDIATRA	CESAREAS_PREVIAS	PESO_NASCER
count	1708.000000	1708.000000	1707.00000	1686.000000	1704.000000	1701.000000	1680.000000
mean	3.013466	2.175059	0.95372	37.901542	37.957746	0.262199	2977.958929
std	4.350403	1.449129	1.22674	3.181517	3.113337	0.600519	691.185194
min	1.000000	1.000000	0.00000	19.000000	19.000000	0.000000	270.000000
25%	2.000000	1.000000	0.00000	37.000000	37.000000	0.000000	2710.000000
50%	2.000000	2.000000	1.00000	39.000000	39.000000	0.000000	3065.000000
75%	3.000000	3.000000	1.00000	40.000000	40.000000	0.000000	3390.000000
max	63.000000	17.000000	10.00000	42.000000	42.000000	5.000000	5625.000000

Figura 13: Descritivo das variáveis numéricas

3.2.6 Formatação dos dados

Os dados devem estar no formato correto para o tipo de análise que você deseja realizar. Muitas vezes é necessário criar novas variáveis com diferentes tipos de recodificação de variáveis numéricas e/ou categóricas, ou simplesmente ajustar os rótulos das categorias.

Exemplos: Formatação e substituição de dados

```
# Trocar números por conteúdo das categorias
# Por exemplo trocar "1" para "partos normais"
dados.TIPO_PARTO.replace(1, 'Parto Normal', inplace=True)
dados.TIPO_PARTO.replace(2, 'Parto Cesáreo', inplace=True)
```

```
# Arredondar números
dados.DURACAO_INT = dados.DURACAO_INT.round(0)

# No caso de variáveis com datas é possível explorar medidas descritivas, desde que especifique
dados.DT_INTERNACAO.describe(datetime_is_numeric=True)

# Converter o tipo de uma coluna OBJECT para DATETIME
dados.DATA_ENTREGA = pd.to_datetime(dados.DATA_ENTREGA)

# Se a coluna for do tipo DATETIME, pode-se trabalhar com funções específicas para datas, por exemplo: contar a quantidade por ano
dados.DATA_ENTREGA.dt.year.value_counts()

# Remover possíveis espaços em branco em variáveis de texto, no início e fim de cada texto
dados.DESTINATARIO = dados.DESTINATARIO.str.strip()
```

Exemplos: Seleção e filtro de dados

```
# Filtrar os 5 maiores valores de uma coluna
dados.nlargest(5, 'IDADE')
# Filtrar os 5 menores  valores de uma coluna
dados.nsmallest(5, 'IDADE')

# Selecionar um subconjunto (intervalo de dados)
adultos = dados[(dados.IDADE >= 18)]

# Selecionar um subconjunto
adultosF = dados[(dados.IDADE >= 18) & (dados.GENERO == 'Feminino')]

# Criar um subconjunto que contenham parte do texto em uma variável
dados2 = dados[dados.ESTADO.str.contains('R')]
```

Exemplos: Criação de novas variáveis e preenchimento com valores

```
# Localizar e visualizar um subconjunto de dados
# criar uma nova variável e atribuir um valor 'qualquer'
dados.loc[ (dados.PESO_NASCER <  2500 ), 'BAIXO_PESO' ] = 'Sim'
dados.loc[ (dados.PESO_NASCER >= 2500 ), 'BAIXO_PESO' ] = 'Não'

# Criando uma nova variável (PESO_US)
dados = dados.assign(PESO_US = (dados.PESO_VIAVEIS - 1000))

# A partir da variável criada, pode-se recalcular valores
# Exemplo: para bebês abaixo de 36 semanas, reduzir 30% o peso
dados.loc[(dados.IG_OBSTETRA < 36), 'PESO_US'] = dados.PESO_US * 0.7

# Pode-se ajustar um conjunto de valores para nulo
dados.loc[ (dados.PESO_US <=  300 ), 'PESO_US' ] = np.nan
```

3.2.7 Verificação final e salvar

Após realizar todo o tratamento e limpeza dos dados é importante realizar uma verificação final para garantir que não haja nenhum erro e que os dados estejam prontos para a análise.

Exemplo: Salvando os dados tratados

```
# Salvar em excel, um novo arquivo, com os dados tratados
dados.to_excel("BD_PARTOS_corrigida.xlsx")
```

3.3 Bibliotecas e funções utilizadas

Funções da biblioteca Pandas

```
# Importar a biblioteca Pandas
import pandas as pd
# Para ler os dados de um arquivo CSV
dados = pd.read_csv('pacientes.csv')
# Para excluir uma coluna da tabela
dados = dados.drop('UBS', axis=1)
# Mostrar uma lista com os nomes das colunas
dados.columns
# Mostrar as informações das variáveis importadas
dados.info()
# Mostrar uma amostra aleatória de 10 linhas
dados.sample(10)
# Mostrar o início (topo) da lista de dados
dados.head()
# Mostrar o fim (cauda) da lista de dados
dados.tail()
# Pode-se apagar as linhas onde todas as colunas forem iguais
dados.drop_duplicates()
# Contando a quantidade de casos por categoria em uma variável
dados.TIPO_PARTO.value_counts()
# Apagar as linhas com qualquer valor nulo em qualquer variável
dados.dropna(how = 'any')
# Resumo descritivo das colunas numéricas
dados.describe()
# Por exemplo trocar "1" para "partos normais"
dados.TIPO_PARTO.replace(1, 'Parto Normal', inplace=True)
# Filtrar os 5 maiores valores de uma coluna
dados.nlargest(5, 'IDADE')
# Filtrar os 5 menores  valores de uma coluna
dados.nsmallest(5, 'IDADE')
# Salvar em excel, um novo arquivo, com os dados tratados
dados.to_excel("BD_PARTOS_corrigida.xlsx")
```

Saiba mais sobre as bibliotecas e funções utilizadas

Biblioteca	Função	Link de acesso	Referência
Pandas	read_excel()	Acesse o site	(17)
Pandas	value_counts()	Acesse o site	(18)
Pandas	describe()	Acesse o site	(19)

4. Estatística descritiva

As estatísticas descritivas são usadas para descrever e resumir as características de um conjunto de dados. Isso inclui medidas de frequência, as medidas de tendência central, como média, mediana e moda, as medidas de dispersão, como amplitude, variância e desvio padrão, além das medidas de posição como percentis e quartis (20). As estatísticas descritivas costumam ser usadas para fornecer uma visão geral de um conjunto de dados e podem ajudar a identificar padrões e relacionamentos nos dados. Elas são uma das ferramentas da análise exploratória de dados e são frequentemente usadas em conjunto com estatísticas inferenciais, que envolvem fazer previsões ou inferências sobre uma população maior com base em uma amostra de dados (15).

4.1 A natureza das variáveis

Na estatística e na pesquisa em geral, as variáveis são características ou propriedades que variam de indivíduo para indivíduo, objeto para objeto ou evento para evento. A natureza das variáveis pode ser classificada em duas categorias principais: variáveis qualitativas e quantitativas (Figura 14).

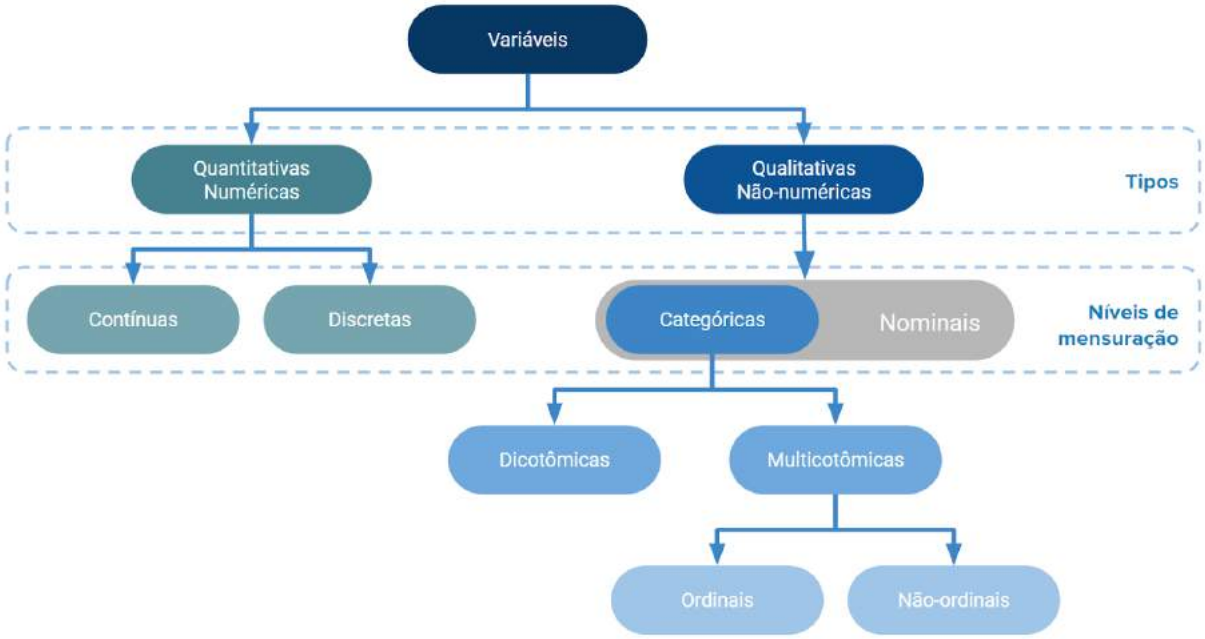


Figura 14: Esquema da classificação dos tipos de variáveis

As variáveis qualitativas são aquelas que descrevem uma característica que não pode ser medida numericamente, como gênero, cor dos olhos, se tem ou não uma doença, entre outras. Elas são geralmente divididas em duas sub-categorias: nominais e ordinais. Variáveis nominais são aquelas que não possuem uma ordem específica, como cor dos olhos, enquanto variáveis ordinais têm uma ordem específica, como nível de educação.

As variáveis quantitativas são aquelas que podem ser medidas numericamente e geralmente são divididas em duas sub-categorias: discretas e contínuas. As variáveis discretas são aquelas que assumem um conjunto finito ou contável de valores, como o número de irmãos de uma pessoa. As variáveis contínuas são aquelas que podem assumir uma infinidade de valores em um intervalo contínuo, como a altura ou o peso de uma pessoa.

A compreensão da natureza das variáveis é fundamental na análise estatística de dados e na interpretação dos resultados. A escolha das técnicas estatísticas apropriadas depende da natureza das variáveis, bem como dos objetivos da pesquisa.

4.2 Distribuição de frequência

Uma distribuição de frequência é um gráfico ou tabela que mostra a frequência com que cada valor ou faixa de valores ocorre em um conjunto de dados. Existem dois principais tipos de distribuição de frequência: a simples (valores absolutos) e relativa (valores percentuais). Uma distribuição de frequência pode ser representada de várias maneiras, por exemplo por meio de uma tabela de frequência, por gráfico de barras ou gráfico de colunas.

Para facilitar a construção da tabela de frequência, pode-se organizar os dados em ordem crescente ou decrescente. Em seguida, basta contar o número de valores em cada categoria ou faixa de valores. A tabela pode ou não conter os valores ausentes bem como o respectivo percentual de cada categoria ou faixa de valores.

Exemplo: Suponha que você deseja saber qual é a frequência absoluta e relativa das categorias da variável sexo.

```
# Frequência absoluta
fa = dados.SEXO.value_counts()
print('Frequência absoluta')
print(fa)

# Frequência relativa
fr = dados.SEXO.value_counts(normalize = True)
print('Frequência relativa')
print(fr)

# Tabela de frequência (incluindo os valores nulos)
fan = dados.SEXO.value_counts(dropna=False)
print('Frequência absoluta considerando os nulos')
print(fan)

# Tabela de frequência relativa (incluindo os valores nulos)
frn = dados.SEXO.value_counts(normalize = True, dropna=False)
print('Frequência relativa considerando os nulos')
```

```
print(frn)
```

```
Frequência absoluta
Masculino    853
Feminino     797
Name: SEXO, dtype: int64

Frequência relativa
Masculino    0.51697
Feminino     0.48303
Name: SEXO, dtype: float64

Frequência absoluta considerando os nulos
Masculino    853
Feminino     797
NaN           58
Name: SEXO, dtype: int64

Frequência relativa considerando os nulos
Masculino    0.499415
Feminino     0.466628
NaN          0.033958
Name: SEXO, dtype: float64
```

Figura 15: Frequências absolutas e relativas para a variável sexo

Exemplo: Suponha que você deseja criar 5 faixas de valores para o peso ao nascer e calcular a frequência de casos em cada uma das faixas. O parâmetro **bins** da função **value_counts()** pode ajudar, ao criar faixas de valores para a quantidade de **bins** especificados.

```
# Criar uma tabela temporária com a contagem das 5 faixas
tabela = dados.PESO_NASCER.value_counts(bins=5)

# Mostrar a tabela já ordenada pela sequência dos valores das faixas
tabela.sort_index()
```

```
(264.644, 1341.0]      62
(1341.0, 2412.0]      184
(2412.0, 3483.0]     1104
(3483.0, 4554.0]      324
(4554.0, 5625.0]        6
Name: PESO_NASCER, dtype: int64
```

Figura 16: Frequências de 5 faixas de valores para a variável peso ao nascer

4.3 Sumário estatístico

O sumário estatístico é uma descrição numérica dos dados que resume as principais medidas de tendência central (média, mediana e moda) e de dispersão (desvio padrão, variância,

mínimo e máximo) (20). Ele também pode incluir informações sobre a distribuição dos dados, como o coeficiente de assimetria e curtose. O sumário estatístico é útil para uma análise mais aprofundada dos dados e pode ser usado para identificar valores extremos, distribuição dos dados e outras informações importantes para a análise.

4.3.1 Sumário estatístico simplificado

A função *describe()* da biblioteca Pandas, retorna a contagem, média, desvio padrão, valores mínimos e máximos e percentis de todas as variáveis numéricas se aplicada a todo o *DataFrame* ou referente a uma variável se aplicada a uma coluna. Ela excluirá os valores ausentes por padrão.

Exemplo: Suponha que você precise extrair algumas medidas do sumário estatístico de forma automática de todas as variáveis que o Python considera como numérica.

```
# Sumário estatístico simplificado - para variáveis numéricas
dados.describe()
```

	DURACAO_INT	GESTACOES	PARTOS	IG_OBSTETRA	IG_PEDIATRA	CESAREAS_PREVIAS	PESO_NASCER
count	1708.000000	1708.000000	1707.00000	1686.000000	1704.000000	1701.000000	1680.000000
mean	3.013466	2.175059	0.95372	37.901542	37.957746	0.262199	2977.958929
std	4.350403	1.449129	1.22674	3.181517	3.113337	0.600519	691.185194
min	1.000000	1.000000	0.00000	19.000000	19.000000	0.000000	270.000000
25%	2.000000	1.000000	0.00000	37.000000	37.000000	0.000000	2710.000000
50%	2.000000	2.000000	1.00000	39.000000	39.000000	0.000000	3065.000000
75%	3.000000	3.000000	1.00000	40.000000	40.000000	0.000000	3390.000000
max	63.000000	17.000000	10.00000	42.000000	42.000000	5.000000	5625.000000

Figura 17: Sumário estatístico simplificado para todas variáveis numéricas

Exemplo: Suponha que você precise do sumário estatístico simplificado apenas da variável idade gestacional informada pelo obstetra.

```
# Sumário estatístico simplificado da variável IG_OBSTETRA
dados.IG_OBSTETRA.describe()
```

```
count    1686.000000
mean       37.901542
std         3.181517
min        19.000000
25%        37.000000
50%        39.000000
75%        40.000000
max        42.000000
Name: IG_OBSTETRA, dtype: float64
```

Figura 18: Sumário estatístico simplificado para a variável idade gestacional

4.3.2 Medidas de tendência central

As medidas de tendência central são estatísticas utilizadas para resumir a localização central dos dados em um conjunto de observações. Elas fornecem uma ideia da posição central dos dados e incluem a média, a mediana e a moda:

- **Média *(mean)*:** A média aritmética de um conjunto de valores. É calculada somando todos os valores no conjunto de dados e dividindo pelo número total de valores.
- **Mediana *(median)*:** O valor médio em um conjunto de dados quando os valores são ordenados do menor para o maior. Se houver um número ímpar de valores, a mediana é o valor do meio. Se houver um número par de valores, a mediana é a média dos dois valores do meio.
- **Moda *(mode)*:** O valor que ocorre com mais frequência em um conjunto de dados.

Exemplo: Suponha que você deseje calcular a média, mediana e moda para a variável peso ao nascer.

```
# Calcular as medidas de tendência central para o Peso ao Nascer

# Média
x = dados.PESO_NASCER.mean()
print('Média:', round(x, 2))

# Mediana
y = dados.PESO_NASCER.median()
print('Mediana:', y)

# Moda
z = dados.PESO_NASCER.mode()
print('Moda:', z)
```

Resultado:

Média: 2977.96
Mediana: 3065.0
Moda: 3200.0

4.3.3 Medidas de dispersão

Medidas de dispersão incluem:

- **Intervalo:** os valores mínimo e máximo dos dados.
- **Variância *(var)*:** uma medida de quão espalhados estão os valores em um conjunto de dados. É calculada tomando a média das diferenças quadradas entre cada valor e a média do conjunto de dados.
- **Desvio padrão *(std)*:** uma medida de quão espalhados são os valores em um conjunto de dados, calculado como a raiz quadrada da variação. É uma medida de dispersão mais interpretável do que a variância, pois é expressa nas mesmas unidades que os dados originais.

Exemplo: Suponha que você deseje calcular o intervalo, a variância e o desvio padrão para a variável peso ao nascer.

```
# Calcular as medidas de dispersão para o Peso ao Nascer

# Mínimo
min = dados.PESO_NASCER.min()
# Máximo
max = dados.PESO_NASCER.max()
# Intervalo
print('Intervalo:', min, ' - ', max)

# Variância
var = dados.PESO_NASCER.var()
print('Variância:', round(var, 2))

# Desvio padrão
std = dados.PESO_NASCER.std()
print('Desvio Padrão:', round(std, 2))
```

Resultado:

Intervalo: 270.0 - 5625.0
Variância: 477736.97
Desvio Padrão: 691.19

4.3.4 Medidas de posição

Os quartis são medidas que dividem um conjunto de dados em quatro partes iguais (quartos). O primeiro quartil (Q1) é o valor no 25º percentil do conjunto de dados, o segundo quartil (Q2) é o valor no 50º percentil (também conhecido como mediana) e o terceiro quartil (Q3) é o valor no percentil 75.

A diferença entre o primeiro e o terceiro quartis, Q3 - Q1, é conhecida como intervalo interquartil (IQR). O IQR é uma medida de dispersão usada para identificar outliers em um conjunto de dados. Os pontos de dados que estão fora do intervalo Q1 - 1,5 * IQR a Q3 + 1,5 * IQR são considerados outliers em potencial.

Exemplo: Suponha que você deseje calcular o 1º quartil, o 2º quartil (mediana), o 3º quartil, o intervalo interquartil, os outliers inferiores e os outliers superiores para a variável peso ao nascer.

```
# 1° quartil - percentil 25
q1 = dados.PESO_NASCER.quantile(0.25)
print('1° quartil:', q1)

# 2° quartil - percentil 50
q2 = dados.PESO_NASCER.quantile(0.5)
print('2° quartil:', q2)

# 3° quartil - percentil 75
q3 = dados.PESO_NASCER.quantile(0.75)
print('3° quartil:', q3)

# Intervalo interquartil
IQR = q3 - q1
print('Intervalo interquartil:', IQR)

# Outliers inferior
outliersInf = q1 - 1.5 * IQR
print('Limite dos outliers inferiores:', outliersInf)

# Outliers superior
outliersSup = q3 + 1.5 * IQR
print('Limite dos outliers superiores:', outliersSup)
```

Resultado:

1º quartil: 2710.0

2º quartil: 3065.0

3º quartil: 3390.0
Intervalo interquartil: 680.0
Limite dos outliers inferiores: 1690.0
Limite dos outliers superiores: 4410.0

4.4 Sumário Gráfico

O sumário gráfico é uma representação visual dos dados que pode ajudar a identificar padrões, tendências e outliers nos dados (20). Ele pode incluir diferentes tipos de gráficos, como histogramas, gráficos de barras, gráficos de dispersão e gráficos de linha. O sumário gráfico é útil para uma rápida visualização das informações e pode ser usado para identificar padrões e tendências nos dados.

4.4.1 Gráficos para distribuição de frequência

As distribuições de frequência podem ser representadas de várias maneiras graficamente, por exemplo por meio de um gráfico de colunas, gráfico de barras, gráfico de pizza (setores), entre outros. Cada tipo de gráfico tem suas próprias vantagens e desvantagens, e o tipo mais adequado depende do conjunto de dados e do objetivo da análise.

Exemplo: Suponha que você deseje criar um gráfico de colunas para a variável alto risco.

```
# Tabela de frequência da variável
tabelaFreq = dados.ALTO_RISCO.value_counts()

# Fazer o gráfico de colunas
tabelaFreq.plot.bar()
```

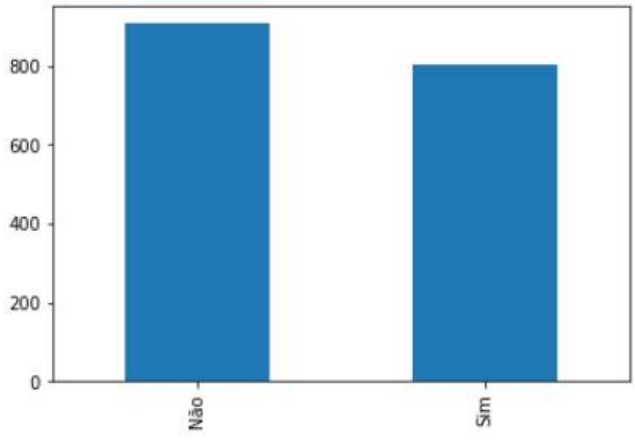


Figura 19: Gráfico de colunas para a variável alto risco

Exemplo: Suponha que você deseje criar um gráfico de barras para a variável sexo.

```
# Tabela de frequência da variável
tabelaFreq = dados.SEXO.value_counts()

# Fazer o gráfico de barras
tabelaFreq.plot.barh().set_title('Frequência por sexo do recém-nascido')
```

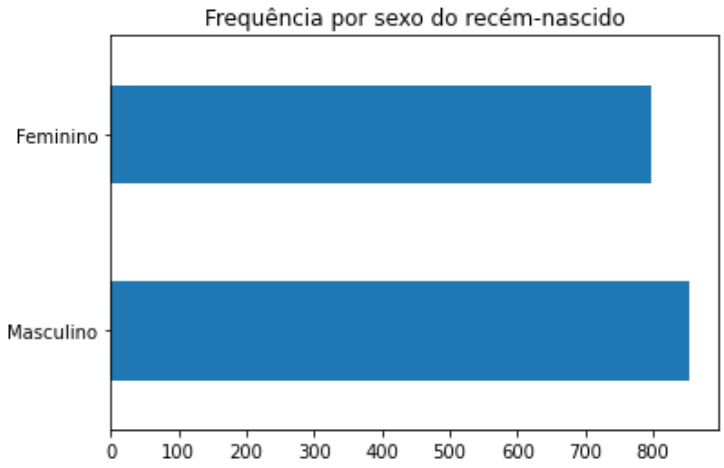


Figura 20: Gráfico de barras para a variável sexo do recém-nascido

Exemplo: Suponha que você deseje criar um gráfico de setores (pizza) para a variável hipertensão.

```
# Tabela de frequência da variável
tabelaFreq = dados.HIPERTENSAO.value_counts()

# Fazer o gráfico de pizza
tabelaFreq.plot.pie(legend=True).set_title('Frequência das gestantes
com hipertensão')
```

Figura 21: Gráfico de pizza para a variável hipertensão

Exemplo: Suponha que você deseje incluir no gráfico anterior os valores percentuais.

```
# Fazer o gráfico de pizza
tabelaFreq.plot.pie(legend=True, autopct='%1.1f%%')
```

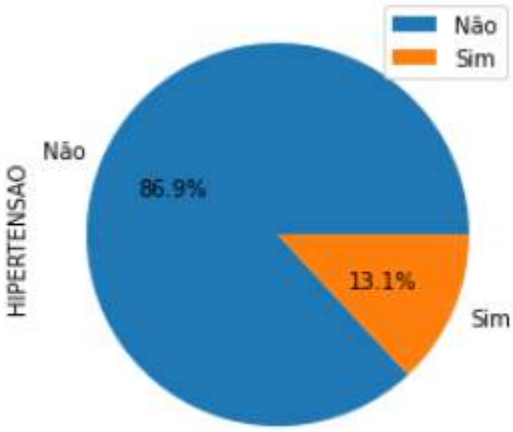


Figura 22: Gráfico de pizza para a variável hipertensão com valores dos rótulos

4.4.2 Gráfico de Histogramas

Um histograma é um tipo de gráfico que é usado para exibir a distribuição de frequência de dados quantitativos. Ele é formado por barras verticais que são desenhadas ao lado de cada outra, formando um gráfico de barras que representa a quantidade de dados em cada intervalo de valores. As barras do histograma são geralmente largas o suficiente para que elas possam ser tocadas sem deixar espaços entre elas, o que torna o gráfico fácil de ler e interpretar. O eixo horizontal do histograma representa os valores dos dados, enquanto o eixo vertical representa a frequência dos dados nesse intervalo de valores. O histograma é útil para visualizar a distribuição de dados e identificar tendências e padrões.

Exemplo: Suponha que você queira fazer um histograma da variável peso ao nascer para saber como esta variável numérica está distribuída.

```
# Importar a biblioteca Seaborn
import seaborn as sns

# Histograma
sns.histplot(dados.PESO_NASCER)
```

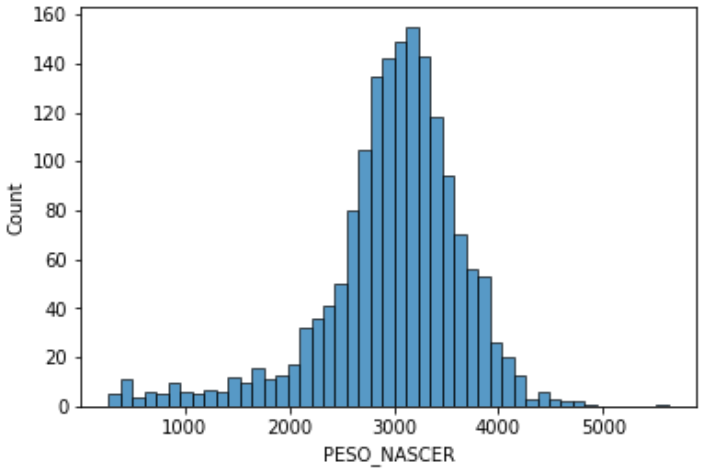


Figura 23: Histograma da variável peso ao nascer

Exemplo: Suponha que você queira formatar o histograma acima, incluindo a linha de densidade, formatação do rótulo da variável e do título do gráfico.

```
# Histograma com linha de densidade (kde)
grafico = sns.histplot(dados.PESO_NASCER, kde=True)

# Formatar o título do gráfico
grafico.set_title('Histograma do peso ao nascer')

# Formatar os rótulos dos eixos
grafico.set_xlabel("Peso ao Nascer")
grafico.set_ylabel("Quantidade")
```

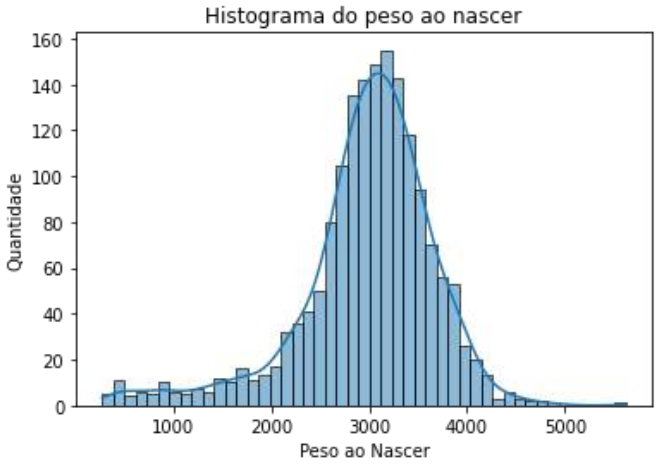


Figura 24: Histograma da variável peso ao nascer

Exemplo: Suponha que você queira fazer um histograma do peso ao nascer agrupado por tipo de parto, incluindo a linha de densidade, formatação do rótulo da variável e do título do gráfico.

```
# Histograma com linha de densidade (kde)
grafico = sns.histplot(data=dados, x='PESO_NASCER',
hue='TIPO_PARTO', linewidth=0, kde=True)

# Formatar o título do gráfico
grafico.set_title('Histograma do peso ao nascer por tipo de parto')
# Formatar os rótulos dos eixos
grafico.set_xlabel("Peso ao Nascer")
grafico.set_ylabel("Quantidade")
```

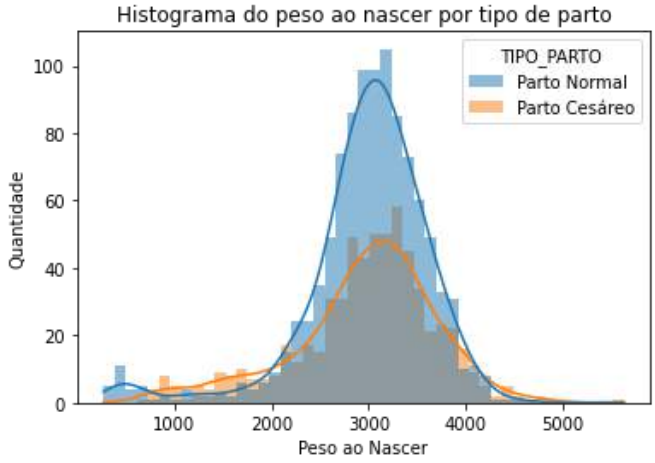


Figura 25: Histograma separado por categoria

4.4.3 Gráfico Boxplot

O gráfico *Boxplot*, também conhecido como gráfico de caixa, é um tipo de gráfico que representa a distribuição de um conjunto de dados. Ele é composto por uma caixa, dois limites e dois conjuntos de traços. A caixa mostra o intervalo interquartil (IQR) dos dados, ou seja, o intervalo entre o primeiro quartil (Q1) e o terceiro quartil (Q3). Os limites representam os valores máximo e mínimo dos dados, exceto os valores atípicos, que são representados por pontos ou símbolos. Um *boxplot* é útil para entender a distribuição dos dados e para comparar vários conjuntos de dados ao mesmo tempo. Ele é especialmente útil para detectar valores atípicos e para comparar conjuntos de dados com distribuições assimétricas ou não-normais (21).

Exemplo: Suponha que você queira fazer um gráfico boxplot para a variável peso ao nascer para saber como esta variável numérica está distribuída.

```
# Importar a biblioteca Seaborn
import seaborn as sns

# Boxplot simples
grafico = sns.boxplot(y=dados.PESO_NASCER)

# Formatar o título do gráfico
grafico.set_title('Boxplot do peso ao nascer')
# Formatar os rótulos dos eixos
grafico.set_ylabel("Peso ao Nascer")
```

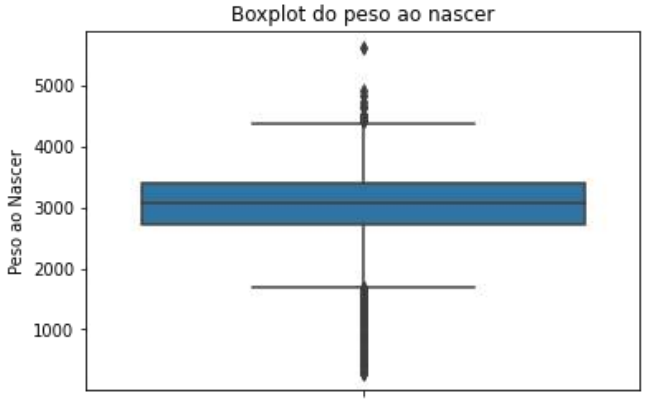


Figura 26: Boxplot do peso ao nascer

Exemplo: Suponha que você queira fazer um gráfico Bloxpot para a variável peso ao nascer agrupado por tipo de parto.

```
# Importar a biblioteca Seaborn
import seaborn as sns

# Boxplot simples
grafico = sns.boxplot(y=dados.PESO_NASCER, x=dados.TIPO_PARTO,
linewidth=1)

# Formatar o título do gráfico
grafico.set_title('Boxplot do peso ao nascer')
# Formatar os rótulos dos eixos
grafico.set_ylabel("Peso ao Nascer")
grafico.set_xlabel("Tipo de Parto")
```

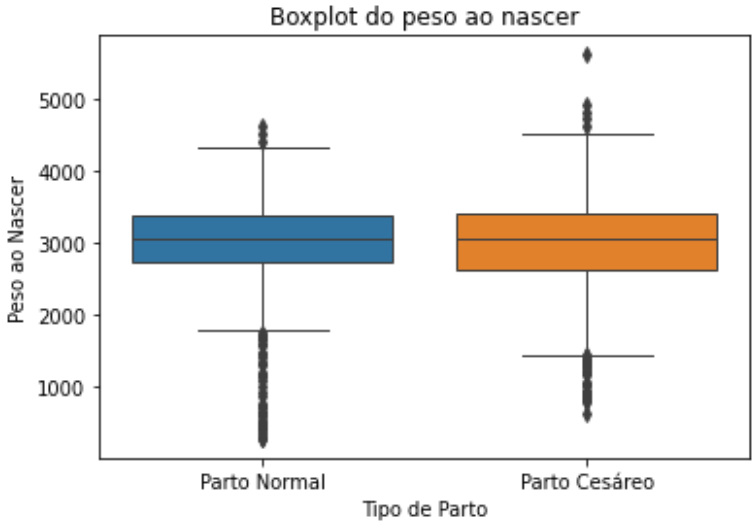


Figura 27: Boxplot do peso ao nascer agrupado por tipo de parto

Exemplo: Suponha que você queira fazer um gráfico Bloxpot para a variável peso ao nascer agrupado por tipo de parto e separado em recém-nascidos prematuros ou não.

```
# Importar a biblioteca Seaborn
import seaborn as sns

# Boxplot simples
grafico = sns.boxplot(y=dados.PESO_NASCER, x=dados.TIPO_PARTO,
hue=dados.PREMATURO_OBS, linewidth=1)

# Formatar o título do gráfico
grafico.set_title('Boxplot do peso ao nascer')
# Formatar os rótulos dos eixos
grafico.set_ylabel("Peso ao Nascer")
grafico.set_xlabel("Tipo de Parto")
```

```
# Colocar a legenda ao lado do gráfico
grafico.legend(bbox_to_anchor=(1,1), title='Prematuro')
```

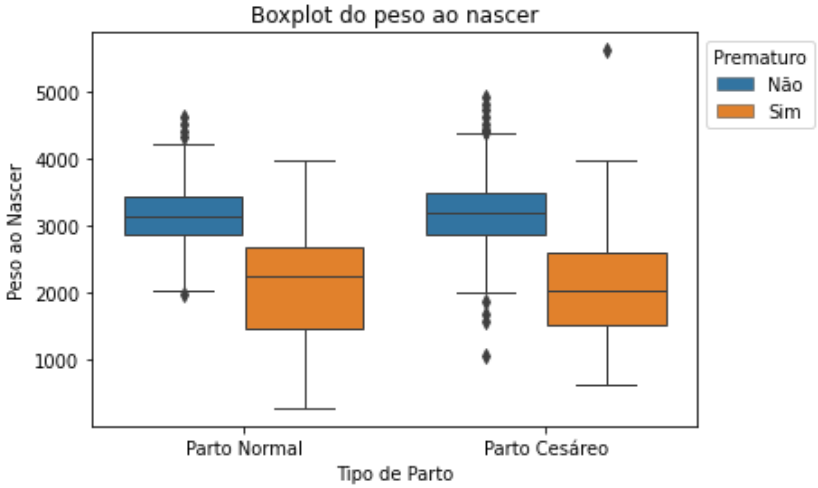


Figura 28: Boxplot do peso ao nascer agrupado por tipo de parto e prematuridade

4.4.4 Gráfico de Dispersão

Um gráfico de dispersão é uma forma de visualizar a relação entre duas variáveis numéricas. Ele é construído desenhando cada par de valores (x, y) em um eixo cartesiano, onde x e y representam as duas variáveis em questão. O resultado é um conjunto de pontos que mostram como as duas variáveis estão correlacionadas. Ele pode ser usado para verificar se há uma relação linear entre as variáveis, tendências, se há outliers (pontos fora da curva) e para avaliar a força e a direção da relação.

Exemplo: Suponha que você queira fazer um de dispersão para ver a correlação do peso ao nascer e a idade gestacional.

```
# Gráfico de Dispersão entre x="IG_OBSTETRA", y="PESO_NASCER"
grafico = sns.scatterplot(data=dados, x="IG_OBSTETRA",
y="PESO_NASCER")

# Formatar o título do gráfico
grafico.set_title('Gráfico de dispersão')

# Formatar os rótulos dos eixos
grafico.set_xlabel('IG Obstetra', fontsize=12)
grafico.set_ylabel('Peso ao Nascer', fontsize=12)
```

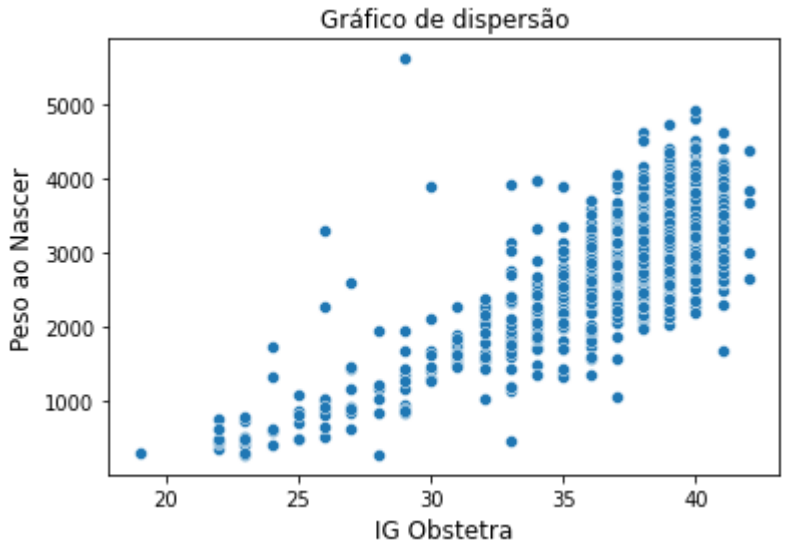


Figura 29: Gráfico de dispersão do peso ao nascer e idade gestacional

4.5 Distribuições

As distribuições são um conceito fundamental na estatística que se refere à forma como os dados estão distribuídos em uma população ou conjunto de dados. Existem vários tipos de distribuições, cada um com suas próprias características e aplicações. Alguns exemplos de distribuições comuns incluem distribuição normal, binomial, t-Student, chi-quadrado, entre outras.

4.5.1 Assimetria e Curtose

Assimetria é a medida da simetria de uma distribuição de dados em torno de sua média. Uma distribuição de dados é considerada assimétrica se a maioria dos dados estiver concentrada em um lado da média, enquanto os dados restantes estiverem espalhados pelo outro lado. para interpretar o índice de assimetria, deve-se extrair o módulo (remover o sinal positivo ou negativo) e interpretar da seguinte forma:

- Índice de assimetria <0,15 = distribuição simétrica
- Índice de assimetria entre 0,15 e 1,0 = distribuição assimétrica moderada
- Índice de assimetria > 1 = distribuição assimétrica forte

Curtose é a medida do achatamento ou alongamento de uma distribuição de dados em relação a uma distribuição padrão, denominada de curva normal. A distribuição normal possui uma curtose de 0,263.

Exemplo: Suponha que você deseje calcular os valores de assimetria e de curtose para a variável idade gestacional e da duração da internação.

```
print('Idade Gestacional')
# Assimetria
assimetria = dados.IG_OBSTETRA.skew()
print('Assimetria: ', round(assimetria, 3))
# Curtose
curtose = dados.IG_OBSTETRA.kurtosis()
print('Curtose: ', round(curtose, 3))

print('Duração da internação')
# Assimetria
assimetria = dados.DURACAO_INT.skew()
print('Assimetria: ', round(assimetria, 3))
# Curtose
curtose = dados.DURACAO_INT.kurtosis()
print('Curtose: ', round(curtose, 3))
```

Resultado:

Idade Gestacional

Assimetria: -2.54

Curtose: 7.967

Duração da internação

Assimetria: 7.165

Curtose: 64.511

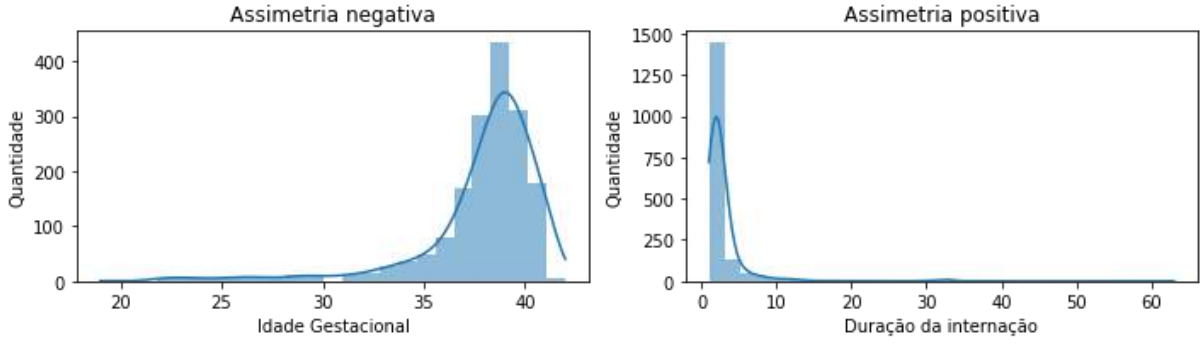


Figura 30: Exemplos de distribuições assimétricas

4.5.2 Distribuição Normal

A distribuição normal é uma distribuição de probabilidade contínua que é usada para modelar e entender como os dados estão distribuídos em uma população. A importância da identificação deste tipo de distribuição dá-se porque muitos testes estatísticos pressupõem que os dados são normalmente distribuídos.

A distribuição normal é definida por dois parâmetros: a média (μ) e o desvio padrão (σ). A média representa o valor central dos dados e o desvio padrão representa a dispersão dos dados em torno da média. É uma distribuição simétrica em torno da média e tem uma curtose média.

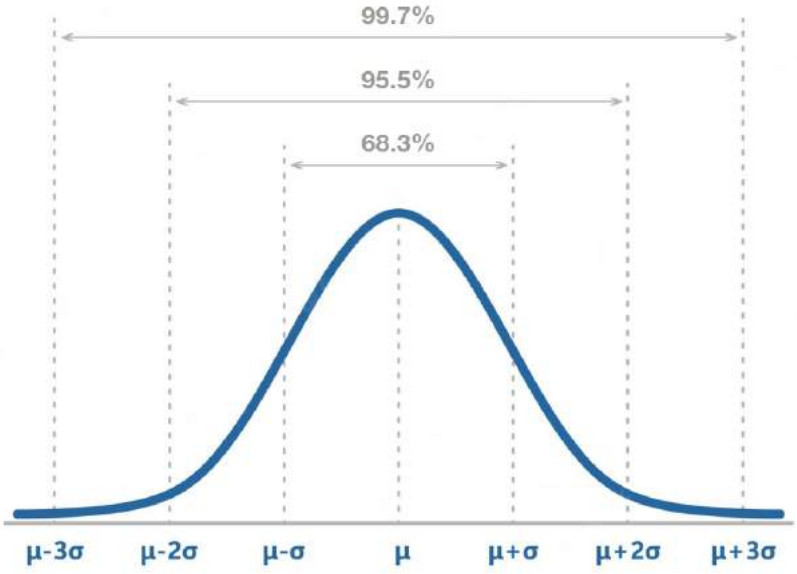


Figura 31: Distribuição Normal

Escore-z

A estatística z ou *score-z* é uma medida de distância em relação à média de uma distribuição normal padrão, ou seja, uma distribuição normal com média 0 e desvio padrão 1. Ela é dada pela equação $z = (x - \mu) / \sigma$. Onde x é o valor da variável, μ é a média da população e σ é o desvio padrão da população.

A estatística z é usada para comparar os valores de uma variável com a média da população e para avaliar a significância estatística de uma diferença entre dois grupos. Ela é especialmente útil quando os dados são normalmente distribuídos e quando o tamanho da amostra é grande. Determina a distância do valor em relação à média em unidades de desvio padrão.

Exemplo: Neste exemplo, vamos calcular o Escore-Z para a variável peso ao nascer e armazenar o resultado em uma nova coluna da base de dados.

```
# Importar a biblioteca Scipy.Stats
import scipy.stats as stats

# Z-Score do PESO_NASCER e armazenar em uma nova coluna no DataFrame
dados['PESO_NASCER_zscore'] = stats.zscore(dados.PESO_NASCER,
nan_policy='omit')
```

```
# Mostrar as duas colunas lado a lado para conferência
colunas = ['PESO_NASCER', 'PESO_NASCER_zscore']
dados[colunas].sample(5)
```

	PESO_NASCER	PESO_NASCER_zscore
165	2960.0	-0.025991
652	3665.0	0.994300
1001	830.0	-3.108571
1200	3583.0	0.875628
1102	2800.0	-0.257546

Figura 32: Visualização de uma amostra de 5 registros com as duas colunas do peso

4.6 Bibliotecas e funções utilizadas

Funções da biblioteca Pandas

```
# Bibliotecas Pandas
import pandas as pd
import matplotlib.pyplot as plt
# Importar dados de um arquivo excel
dados = pd.read_excel(arquivo)
# Frequência absoluta
dados.COLUNA.value_counts()
# Frequência relativa
dados.COLUNA.value_counts(normalize = True)
# Estatísticas descritivas das variáveis numéricas
dados.describe()
# Média
dados.COLUNA.mean()
# Mediana
dados.COLUNA.median()
# Moda
dados.COLUNA.mode()
# Mínimo
dados.COLUNA.min()
# Máximo
dados.COLUNA.max()
# Variância
dados.COLUNA.var()
# Desvio padrão
dados.COLUNA.std()
# Quartis
dados.COLUNA.quantile(0.25)
```

```
# Curtose
dados.COLUNA.kurtosis()
# Assimetria
dados.COLUNA.skew()
# Gráfico de colunas
dados.COLUNA.value_counts().plot.bar()
# Gráfico de barras horizontais
dados.COLUNA.value_counts().plot.barh()
# Gráfico de pizza
dados.COLUNA.value_counts().plot.pie()
```

Funções da biblioteca Seaborn

```
# Bibliotecas Seaborn
import seaborn as sns
# Histograma
sns.histplot(dados.COLUNA)
# Boxplot
sns.boxplot(y=dados.COLUNA)
```

Funções da biblioteca Scipy

```
# Importar a biblioteca Scipy.stats
import scipy.stats as stats
# Z-Score
stats.zscore(dados.PESO_NASCER, nan_policy='omit')
```

Saiba mais sobre as bibliotecas e funções utilizadas

Biblioteca	Função	Link de acesso	Referência
Pandas	describe()	Acesse o site	(19)
Pandas	plot.bar()	Acesse o site	(22)
Seaborn	histplot()	Acesse o site	(23)
Seaborn	boxplot()	Acesse o site	(24)
Scipy.stats	zscore()	Acesse o site	(25)

5. Estatística Inferencial

A estatística inferencial é a parte da estatística que se concentra em fazer inferências sobre uma população com base em uma amostra (20). Ela é usada para responder a perguntas sobre a população, como "Qual é a média da população?", "Qual é a proporção de indivíduos com uma determinada característica na população?" ou "Existe uma diferença significativa entre as médias das duas populações?".

A estatística inferencial usa técnicas estatísticas para estimar parâmetros da população com base em dados da amostra e para avaliar a precisão dessas estimativas. Isso inclui o cálculo de intervalos de confiança e a realização de testes de hipótese. É uma ferramenta amplamente utilizada em várias áreas, como ciência de dados, pesquisa médica, pesquisa de mercado e muito mais. Ela permite que os pesquisadores façam inferências sobre a população com base em dados coletados em uma amostra, o que é muitas vezes mais prático e eficiente do que coletar dados da população completa.

5.1 População e amostra

Na estatística, a população é o conjunto completo de indivíduos ou objetos de interesse. A amostra é uma parte da população que é selecionada para representar a população inteira. A escolha da amostra é importante, pois os resultados da análise da amostra são usados para fazer inferências sobre a população inteira (16). É importante que a amostra seja representativa da população, ou seja, que inclua indivíduos ou objetos que sejam semelhantes à população inteira em termos de características relevantes.

Existem várias técnicas de seleção de amostra, como amostragem aleatória simples, amostragem estratificada e amostragem sistemática. O tamanho da amostra também é importante, pois quanto maior for a amostra, mais precisas serão as estimativas da população. No entanto, o tamanho da amostra também pode afetar o custo e o esforço necessários para coletar e analisar os dados.

5.1.1 Amostragem aleatória simples

A amostragem aleatória simples é um método de seleção de amostra em que cada elemento da população tem a mesma chance de ser selecionado. Isso é feito sorteando aleatoriamente os elementos da população para a amostra. É um método simples e barato de seleção, mas requer que a população esteja listada e que todos os elementos da população tenham a mesma chance de ser selecionados, ou seja, para populações homogêneas. Ela também pode ser ineficiente se a população for muito grande e a amostra for muito pequena.

Exemplo: Suponha que você queira sortear aleatoriamente uma amostra com 150 pacientes de um banco de dados.

```
# Extrair uma amostra aleatória de 150 pacientes
amostra = dados.sample(150)

# Mostrar a amostra
amostra
```

Resultado: Note que os índices da tabela (coluna inicial) indicam os números das linhas sorteadas da base de dados completa.

	DT_ALTA	GESTACOES	PARTOS	IG_OBSTETRA	ALTO_RISCO	TIPO_PARTO	HIPERTENSAO
1217	2013-11-04	2	1.0	37.0	Sim	Parto Cesáreo	Não
1354	2013-12-18	2	1.0	41.0	Não	Parto Normal	Não
472	2013-10-23	1	0.0	40.0	Não	Parto Normal	Sim
1145	2013-10-20	2	1.0	39.0	Não	Parto Cesáreo	Não
196	2013-12-07	2	1.0	38.0	Não	Parto Normal	Não
...	...	...	...	...	...	...	...

Figura 33: Tabela com sorteio da amostra aleatória

5.1.2 Amostragem sistemática

A amostragem sistemática é um método de seleção de amostra em que os elementos da população são selecionados de forma sistemática com base em um intervalo fixo. Para selecionar a amostra usando amostragem sistemática, é preciso escolher um ponto inicial aleatório na lista da população e selecionar os elementos a partir daí de acordo com o intervalo fixo. Isso garante que todos os elementos da população tenham a mesma chance de ser selecionados para a amostra. A amostragem sistemática é mais fácil de implementar do que a amostragem aleatória simples, pois não requer que a população esteja listada (20). Ela pode ser afetada por padrões na população que ocorrem a intervalos regulares e não representar subgrupos da população.

Exemplo: Suponha que você queira sortear sistematicamente uma amostra a cada 10 casos a partir do 3º caso no banco de dados.

```
# Importar as bibliotecas
import numpy as np
# Definir o tamanho do passo, no exemplo de 10 em 10 casos
passo = 10
# Criar uma lista com números de 0 a tamanho da base de dados
indices = np.arange(3, len(dados), step=passo)
# Caso queira conferir a lista de índices criados
print(indices)
# Selecionar na base de dados as linhas com os índices criados
amostra_sistematica = dados.iloc[indices]
# Mostrar a amostra
amostra_sistematica
```

Resultado: No exemplo, é mostrado uma lista com os índices sorteados e a tabela com os registros sorteados sistematicamente. Note que os índices da tabela (coluna inicial) indicam os números das linhas sorteadas da base de dados completa.

```
[   3   13   23   33   43   53   63   73   83   93  103  113  123  133
  143  153  163  173  183  193  203  213  223  233  243  253  263  273
  283  293  303  313  323  333  343  353  363  373  383  393  403  413
  423  433  443  453  463  473  483  493  503  513  523  533  543  553
  563  573  583  593  603  613  623  633  643  653  663  673  683  693
  703  713  723  733  743  753  763  773  783  793  803  813  823  833
  843  853  863  873  883  893  903  913  923  933  943  953  963  973
  983  993 1003 1013 1023 1033 1043 1053 1063 1073 1083 1093 1103 1113
 1123 1133 1143 1153 1163 1173 1183 1193 1203 1213 1223 1233 1243 1253
 1263 1273 1283 1293 1303 1313 1323 1333 1343 1353 1363 1373 1383 1393
 1403 1413 1423 1433 1443 1453 1463 1473 1483 1493 1503 1513 1523 1533
 1543 1553 1563 1573 1583 1593 1603 1613 1623 1633 1643 1653 1663 1673
 1683 1693 1703]
```

	DT_ALTA	GESTACOES	PARTOS	IG_OBSTETRA	ALTO_RISCO	TIPO_PARTO	HIPERTENSAO
3	2013-12-05	2	1.0	41.0	Não	Parto Normal	Não
13	2014-04-09	1	0.0	39.0	Não	Parto Normal	Não
23	2014-08-18	2	1.0	39.0	Não	Parto Normal	Não
33	2014-03-17	2	1.0	38.0	Não	Parto Normal	Não
43	2014-04-26	2	0.0	38.0	Sim	Parto Normal	Não
...	...	...	...	...	...	...	...

Figura 34: Tabela com sorteio da amostra sistemática

5.1.3 Amostragem estratificada

A amostragem estratificada é um método de seleção de amostra em que a população é dividida em grupos (estratos) com base em uma ou mais características e uma amostra é selecionada de cada estrato (15). Isso é feito para garantir que a amostra seja representativa de toda a população e que todas as subpopulações estejam representadas na amostra. A amostragem estratificada é mais precisa do que a amostragem aleatória simples para

populações heterogêneas e é útil quando é importante garantir que todas as subpopulações estejam representadas na amostra.

Exemplo: Suponha que você queira sortear uma amostra contendo 10% da base de dados completa, de forma aleatória e que esteja estratificada pela variável sexo.

```
# Importar as bibliotecas
from sklearn.model_selection import StratifiedShuffleSplit

# Filtrar registros removendo os nulos das duas colunas
dados2 = dados.dropna(how = 'any', subset=['SEXO'])

# Nos dados totais temos 54% de masculino e 46% de Feminino
teste = dados2.SEXO.value_counts(normalize=True)
print('Porcentagem por sexo nos dados completos:')
print(teste)

# Tamanho da amostra de 10% do total da base de dados
tamanho_amostra = 0.1
split = StratifiedShuffleSplit(test_size=0.1)

for x, y in split.split(dados2, dados2.SEXO):
  df_x = dados2.iloc[x]
  df_y = dados2.iloc[y]

# df_y é a amostra com 10% dos dados balanceados por sexo
# df_x é a amostra com 90% dos dados (mas não está balanceado)

# Para conferir se a estratificação ficou aproximada à da população,
# vamos contar quantos casos por sexo têm na amostra gerada (10%)
print()
testeAmostra = df_y.SEXO.value_counts(normalize=True)
print('Porcentagem por sexo na amostra:')
print(testeAmostra)
```

Resultado: Observe que as frequências relativas da amostra são similares às frequências da distribuição na base de dados completa.

```
Porcentagem por sexo nos dados completos:
Masculino    0.51697
Feminino     0.48303
Name: SEXO, dtype: float64

Porcentagem por sexo na amostra:
Masculino    0.515152
Feminino     0.484848
Name: SEXO, dtype: float64
```

Figura 35: Frequências relativas da base de dados e da amostra por sexo

5.1.4 Tamanho da amostra

O tamanho da amostra é o número de elementos da amostra selecionados para representar a população. O cálculo do tamanho da amostra é importante porque quanto maior for a amostra, mais precisas serão as estimativas da população. No entanto, o tamanho da amostra também pode afetar o custo e o esforço necessários para coletar e analisar os dados.

Existem vários fatores que devem ser considerados ao calcular o tamanho da amostra, como:

- **Tamanho da população:** O tamanho da amostra pode ser proporcional ao tamanho da população. Quanto maior for a população, maior pode ser o tamanho da amostra.
- **Nível de precisão:** O tamanho da amostra pode ser ajustado para atingir um nível desejado de precisão nas estimativas da população. Por exemplo, você pode querer uma amostra maior se desejar uma estimativa mais precisa da média da população.
- **Nível de confiança:** O tamanho da amostra também pode ser ajustado para atingir um nível desejado de confiança nos resultados. Por exemplo, você pode querer uma amostra maior se desejar um intervalo de confiança mais estreito.

Existem várias fórmulas e tabelas que podem ser usadas para calcular o tamanho da amostra para diferentes situações (26). Por exemplo, uma das fórmulas é a proposta por Slovin que pode ser utilizada como uma equação geral para estimar uma população mas não temos referência do comportamento dela (27). No entanto, é importante lembrar que o cálculo do tamanho da amostra é apenas uma estimativa e que o tamanho real da amostra pode precisar ser ajustado de acordo com os resultados esperados para a pesquisa. Outros fatores, como a qualidade dos dados e a validade da medida, também são importantes ao fazer estimativas.

Exemplo: Suponha que você queira calcular o tamanho de uma amostra padrão para uma população heterogênea, na qual não temos referências do comportamento dela. Iremos assumir um erro amostral de 5% e considerar que a população total é composta por aproximadamente 5 mil pessoas.

```
# Função para cálculo amostral padrão para uma população heterogênea
# Fórmula de Slovin
def calculoAmostral(e, N):
    n = (N / (1 + (N*(e**2))))
    return (n)

# Erro amostral definido para o estudo
erro = 0.05

# População estimada para o estudo
populacao = 5000
```

```
# Cálculo da amostra
amostra = calculoAmostral(erro, populacao)
# arredondar o resultado para valores inteiros
round(amostra)
```

Resultado: 370 → A partir da fórmula de Slovin, foi estimado 370 como um tamanho amostral.

5.2 Testes de hipótese

O teste de hipótese é uma ferramenta estatística usada para avaliar se os resultados observados em uma amostra são consistentes com uma determinada hipótese sobre a população (20). Ele consiste em formular duas hipóteses opostas, a hipótese nula (H0) e a hipótese alternativa (H1). A hipótese nula é uma afirmação que se assume verdadeira até que seja refutada pelos dados. A hipótese alternativa é a afirmação que se quer testar.

Para realizar um teste de hipótese, primeiro é necessário definir o nível de significância (α). Esse é o limite aceitável de probabilidade para rejeitar a hipótese nula quando ela é verdadeira. O nível de significância é geralmente definido como 5%. Em seguida, é necessário coletar/extrair uma amostra e calcular o valor do teste estatístico. O valor do teste estatístico é comparado com o nível de significância para determinar se os resultados da amostra são consistentes com a hipótese nula. Se o valor do teste estatístico for menor que o nível de significância, a hipótese nula é rejeitada. Se o valor do teste estatístico for maior ou igual ao nível de significância, a hipótese nula é aceita. Assim, para interpretar o teste estatístico verifica-se se a hipótese nula foi rejeitada ou não. Sendo:

- H0: a hipótese nula
- H1: a hipótese alternativa
- p-value: 0,05, assim:
    - se p-value < 0,05, rejeita-se H0 (conclui-se que a H1 é verdadeira)
    - se p-value >= 0,05, aceita-se H0 (conclui-se que a H0 é verdadeira)

Os testes de hipóteses são úteis porque permitem avaliar a validade de uma determinada afirmação ou hipótese com base em dados empíricos. Estes testes são amplamente utilizados em diversas áreas, incluindo ciência, pesquisa e negócios, para tomar decisões com base em evidências. No entanto, é importante lembrar que os testes de hipóteses não são infalíveis e podem ser limitados pelo tamanho da amostra, a qualidade dos dados e outros fatores. Além disso, é importante lembrar que os resultados dos testes de hipóteses só são válidos para a população da qual a amostra foi coletada, e não necessariamente para outras populações ou contextos.

5.3 Intervalos de confiança

Os intervalos de confiança consistem em um intervalo de valores que inclui o valor verdadeiro da população com uma determinada probabilidade (16). Por exemplo, um intervalo de confiança de 95% inclui o valor verdadeiro da população com 95% de probabilidade.

Os intervalos de confiança são úteis porque permitem que os pesquisadores façam estimativas da população com uma determinada precisão. Por exemplo, imagine que você queira estimar a média de idade de uma determinada população. Você pode coletar uma amostra da população e calcular um intervalo de confiança para a média de idade. Isso lhe daria um intervalo de valores que inclui o valor verdadeiro da média da população com uma determinada probabilidade.

Para calcular um intervalo de confiança, é necessário definir o nível de confiança (por exemplo, 95%) e o tamanho da amostra. O tamanho da amostra afeta a precisão da estimativa, portanto é importante coletar uma amostra suficientemente grande. Destaca-se que os intervalos de confiança são apenas estimativas e que o valor verdadeiro da população pode não estar incluído no intervalo com a probabilidade especificada. Assim como a amostra, os intervalos de confiança só são válidos para a população da qual a amostra foi coletada, e não necessariamente para outras populações ou contextos.

Para exemplificar, vamos supor que precisamos calcular a média da idade gestacional de uma amostra de dados obstétricos e o respectivo intervalo de confiança. O nível de confiança desejado é de 95%, o que significa que o intervalo deve conter a verdadeira média da população com uma probabilidade de 95%. Para isso, precisamos obter o valor crítico da distribuição t de Student, que pode ser calculado utilizando a função **"t.ppf()"** presente na biblioteca **"scipy.stats"** e os graus de liberdade do intervalo de confiança. Em seguida, calculamos os limites do intervalo de confiança a partir da média, do desvio padrão, do número de observações e do valor crítico da distribuição t de Student. Por fim, mostramos os valores obtidos.

Exemplo: Suponha que você queira calcular a média do peso ao nascer e o respectivo intervalo de confiança populacional.

```
import numpy as np
from scipy.stats import t

# Nível de confiança desejado (95%)
conf_level = 0.95

# Calcular a média e o desvio padrão dos dados
media = dados.PESO_NASCER.mean()
```

```
std_dev = dados.PESO_NASCER.std()

# Número de observações
n = len(dados.PESO_NASCER)

# Graus de liberdade para o intervalo de confiança
grausLiberdade = n - 1

# Valor crítico da distribuição t de Student
t_value = t.ppf((1 + conf_level) / 2, grausLiberdade)

# Limites do intervalo de confiança
inf = media - t_value * std_dev / np.sqrt(n)
sup = media + t_value * std_dev / np.sqrt(n)

# Imprime a média e o desvio padrão
print('Média:', round(media),' Desvio padrão:', round(std_dev))

# Imprime o intervalo de confiança para a média
print('Intervalo de confiança:', round(inf),'-', round(sup))
```

Resultado:

Média: **2978** Desvio padrão: **691**

Intervalo de confiança: **2945 - 3011**

A média do peso ao nascer na amostra é de 2.978 gramas com desvio padrão de 691 gramas e o intervalo de confiança (95%) estima que o verdadeiro valor populacional está entre 2.945 e 3.011 gramas.

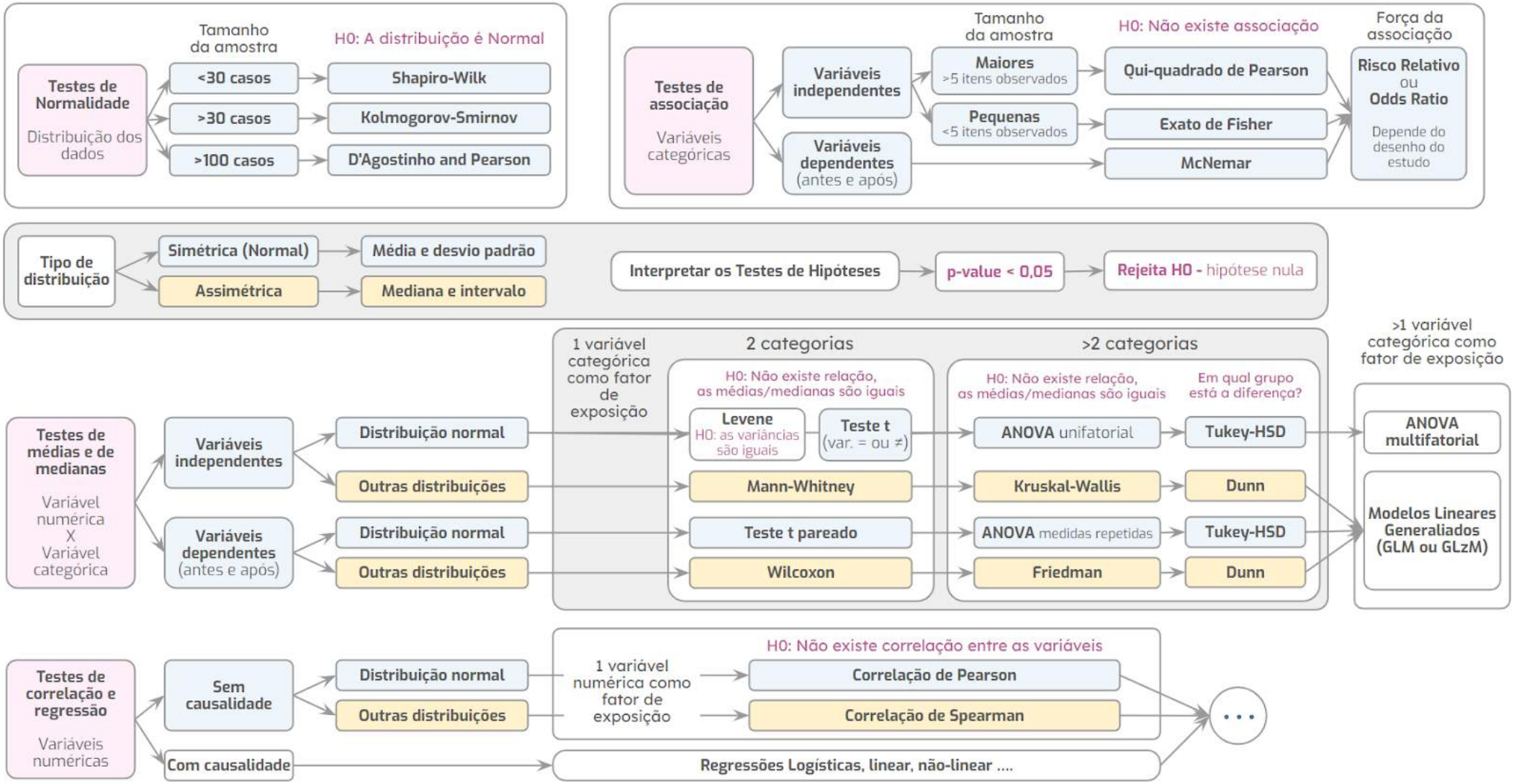
5.4 Como definir qual teste de hipótese utilizar?

Nos próximos capítulos, veremos como aplicar diversos testes de hipóteses. Entretanto, antes de utilizá-los precisamos definir qual teste aplicar em cada situação. Para isso, é importante identificar além da hipótese nula e a hipótese alternativa, é preciso identificar os tipos de variáveis (quantitativas ou qualitativas), o tipo de amostra (independentes ou pareadas/emparelhadas), o número de grupos (dois ou mais grupos), se as distribuições das variáveis são normais ou assimétricas. Com base nessas informações, é possível escolher o teste de hipótese mais apropriado em função das características específicas do problema em questão. Lembre-se que é importante consultar um especialista em estatística ou buscar referências específicas para cada situação. A seguir podemos observar um fluxograma que procura resumir e orientar a escolha de alguns testes de hipóteses comumente utilizados na área da saúde.

Como descrever meus dados e quais testes estatísticos devo usar?

Choosing the Right Statistical Test? Análise introdutória de dados em Saúde

Como citar:
Gaspar, Juliano de Souza, 2023. Como descrever meus dados e quais testes estatísticos devo usar? Adaptado de Marco Mello e Jutta Schmid.
DOI: https://doi.org/10.5281/zenodo.7855276



5.5 Testes de normalidade

Os testes de distribuição normal são usados para verificar se um conjunto de dados segue uma distribuição normal. Identificar se a variável da amostra que se deseja analisar possui uma distribuição normal, é importante porque muitos testes estatísticos pressupõem que os dados sejam normalmente distribuídos. Existem vários testes de distribuição normal, cada um com suas próprias características e aplicações. Para mais informações sobre essas diferenças e características, recomendamos a leitura do artigo "Avaliação da normalidade dos dados em estudo clínicos e experimentais." de Miot (2017) (28).

Teste de Normalidade	Resumo das principais características
D'Agostino and Pearson's (Normal Test)	• Foi desenvolvido para lidar com amostras numerosas (n > 100) • Apresenta desempenho próximo ao do Shapiro-Wilk
Shapiro-Wilk (SW)	• Recomendado para amostras pequenas (n < 30 casos)
Kolmogorov-Smirnov (KS)	• Para amostra com mais de 30 casos • Geralmente apresenta desempenho abaixo de outros testes • Recomenda-se o KS com a correção de Lilliefors • Indicado quando a distribuição contiver muitos dados extremos

Tabela 2: Resumo das principais características dos testes de normalidade

É importante lembrar que nenhum teste é perfeito e que é preciso considerar vários fatores ao escolher um teste de distribuição normal, como o tamanho da amostra, a presença de outliers e a distribuição dos dados.

Ao aplicar os testes de normalidade, a hipótese nula é de que a distribuição é normal e o nível de significância é geralmente fixado em 0,05. Assim, para interpretar o teste estatístico verifica-se se a hipótese nula foi rejeitada ou não. Sendo:

- H0: a distribuição é normal
- H1: a distribuição não é normal
- p-value: 0,05, assim:
    - se p-value < 0,05, rejeita-se H0 (conclui-se que a distribuição não é normal)
    - se p-value >= 0,05, aceita-se H0 (conclui-se que a distribuição é normal)

5.5.1 Teste de D'Agostino e Pearson's

O teste de D'Agostino e Pearson's é um teste de hipótese que verifica se um conjunto de dados é normalmente distribuído. Ele é baseado nas estatísticas de assimetria e curtose, que medem a simetria e a forma da distribuição dos dados em relação à distribuição normal. Ele é

mais adequado para conjuntos de dados com tamanhos médios e grandes e é robusto a outliers.

Exemplo: Suponha que você queira saber se a variável peso ao nascer é normalmente distribuída utilizando o teste de D'Agostino e Pearson's.

```
# Selecionar os dados da variável que se deseja testar sem nulos
valores = dados.PESO_NASCER.notnull()

# Executar o teste de Normal Test - D'Agostino and Pearson's
resultado = stats.normaltest(valores)
pvalue = resultado.pvalue

# Mostrar o resultado
print('Normal Test p-value: ', pvalue)
```

Resultado: Normal Test p-value: <0,001 → Como o resultado é menor que nível de significância (0,05), rejeita-se a hipótese nula (H0). Ou seja, existem evidências estatísticas para afirmar que a distribuição do peso ao nascer não segue uma distribuição normal.

5.5.2 Teste de Shapiro-Wilk

O teste de Shapiro-Wilk é um teste de hipótese que verifica se os dados são normalmente distribuídos com base na ordenação dos valores dos dados. Ele é um dos testes mais precisos para verificar a normalidade, mas é menos eficiente para conjuntos de dados grandes.

Exemplo: Suponha que você queira saber se a variável peso ao nascer é normalmente distribuída utilizando o teste de Shapiro-Wilk.

```
# Selecionar os dados da variável que se deseja testar
valores = dados.PESO_NASCER.notnull()

# Executar o teste de Shapiro-Wilk
resultado = stats.shapiro(valores)
pvalue = resultado.pvalue

# Mostrar o resultado
print('Shapiro-Wilk p-value: ', pvalue)
```

Resultado: Shapiro-Wilk p-value: <0,001 → Como o resultado é menor que nível de significância (0,05), rejeita-se a hipótese nula (H0). Ou seja, existem evidências estatísticas para afirmar que a distribuição do peso ao nascer não segue uma distribuição normal.

5.5.3 Teste de Kolmogorov-Smirnov

O teste de Kolmogorov-Smirnov é um teste de hipótese que verifica se um conjunto de dados é normalmente distribuído. Ele é baseado na comparação dos quantis teóricos da distribuição normal com os quantis observados nos dados. É um dos testes mais amplamente utilizados para verificar a normalidade dos dados, mas é menos preciso do que outros testes. Ele é mais adequado para conjuntos de dados com tamanhos grandes e é robusto a outliers.

Exemplo: Suponha que você queira saber se a variável peso ao nascer é normalmente distribuída utilizando o teste de Kolmogorov-Smirnov.

```
# Selecionar os dados da variável que se deseja testar
valores = dados.PESO_NASCER.notnull()

# Executar o teste de Kolmogorov-Smirnov
resultado = stats.kstest(valores, 'norm')
pvalue = resultado.pvalue

# Mostrar o resultado
print('Kolmogorov-Smirnov p-value: ', pvalue)
```

Resultado: Kolmogorov-Smirnov p-value: <0,001 → Como o resultado é menor que nível de significância (0,05), rejeita-se a hipótese nula (H0). Ou seja, existem evidências estatísticas para afirmar que a distribuição do peso ao nascer não segue uma distribuição normal.

5.6 Bibliotecas e funções utilizadas

Funções da biblioteca Pandas

```
# Importar a biblioteca
import pandas as pd
# Extrair uma amostra aleatória
dados.sample(150)
# Selecionar na base de dados uma linha com o índice informado
dados.iloc[10]
# Selecionar as linhas iguais aos índices de uma lista
dados.iloc[lista]
```

Funções da biblioteca Numpy

```
# Importar a biblioteca
import numpy as np
# criar uma lista com números múltiplos de 10 a partir de 0 a tamanho da base de dados
lista = np.arange(3, len(dados), step=10)
```

Funções da biblioteca Sklearn

```
# Importar as bibliotecas
from sklearn.model_selection import StratifiedShuffleSplit
# Dividir a amostra aleatoriamente em dois sub-conjuntos com tamanhos específicos. Neste exemplo subgrupos de 10%/90%
split = StratifiedShuffleSplit(test_size=0.1)
```

Funções da biblioteca Scipy

```
# Importar a biblioteca Scipy.stats
import scipy.stats as stats
# Teste de Normal Test - D'Agostino and Pearson's
resultado = stats.normaltest(dados.COLUNA)
pvalue = resultado.pvalue
# Executar o teste de Shapiro-Wilk
resultado = stats.shapiro(dados.COLUNA)
pvalue = resultado.pvalue
# Executar o teste de Kolmogorov-Smirnov
resultado = stats.kstest(dados.COLUNA, 'norm')
pvalue = resultado.pvalue
```

Saiba mais sobre as bibliotecas e funções utilizadas

Biblioteca	Função	Link de acesso	Referência
Pandas	sampe()	Acesse o site	(29)
Pandas	iloc()	Acesse o site	(30)
NumPy	arange()	Acesse o site	(31)
Scikit-learn	StratifiedShuffleSplit()	Acesse o site	(32)
Scipy.stats	normaltest()	Acesse o site	(33)
Scipy.stats	shapiro()	Acesse o site	(34)
Scipy.stats	kstest()	Acesse o site	(35)

6. Estatística Inferencial sobre variáveis categóricas

6.1 Tabelas de contingência

As tabelas de contingência são uma ferramenta estatística usada para examinar a relação entre duas variáveis categóricas. Elas mostram a frequência ou o percentual de ocorrências em cada combinação de categorias das duas variáveis. As tabelas de contingência são úteis porque permitem que você visualize facilmente a relação entre as duas variáveis e faça inferências sobre essa relação. No entanto, é importante lembrar que as tabelas de contingência não podem ser usadas para estabelecer causalidade, ou seja, para afirmar que uma variável causa outra (20).

		Variável de desfecho		
		Presente	Ausente	Total
Variável preditora	Presente	A	B	A+B
	Ausente	C	D	C+D
	Total	A+C	B+D	A+B+C+D

Tabela 3: Tabela de contingência

Exemplo: Suponha que você queira saber quantos recém-nascidos por categoria de sexo nasceram por via de parto normal e por cesárea.

```
# Remover os registros ausentes de SEXO, TIPO_PARTO
df = dados.dropna(how = 'any', subset=['SEXO', 'TIPO_PARTO'])
# Tabela cruzada - valor absoluto
tabAbsoluto = pd.crosstab(df.SEXO, df.TIPO_PARTO)
print(tabAbsoluto)
print()
# Tabela cruzada - valor relativo (%)
tabRelativo = pd.crosstab(df.SEXO, df.TIPO_PARTO, normalize=True)
print(tabRelativo)
```

Valores absolutos

TIPO_PARTO	Parto Cesáreo	Parto Normal
SEXO		
Feminino	274	523
Masculino	356	497

Valores relativos

TIPO_PARTO	Parto Cesáreo	Parto Normal
SEXO		
Feminino	0.166061	0.316970
Masculino	0.215758	0.301212

Figura 37: Exemplos de tabelas de contingência

6.2 Testes de independência

Os testes de independência são uma ferramenta estatística usada para avaliar se duas variáveis são independentes, ou seja, se uma variável não afeta a outra. Eles são geralmente realizados em uma tabela de contingência, que é uma ferramenta estatística usada para examinar a associação entre duas variáveis categóricas. Ao aplicar e interpretar um teste de independência é preciso identificar:

- H0: Não existe associação entre as variáveis selecionadas
- H1: Existe associação entre as variáveis selecionadas
- Escolher o teste: Qui-quadrado, Exato de Fisher, McNemar
- p-value: 0,05, assim:
    - se p-value < 0,05, rejeita-se H0 (conclui-se que existe associação)
    - se p-value >= 0,05, aceita-se H0 (conclui-se que não existe associação)

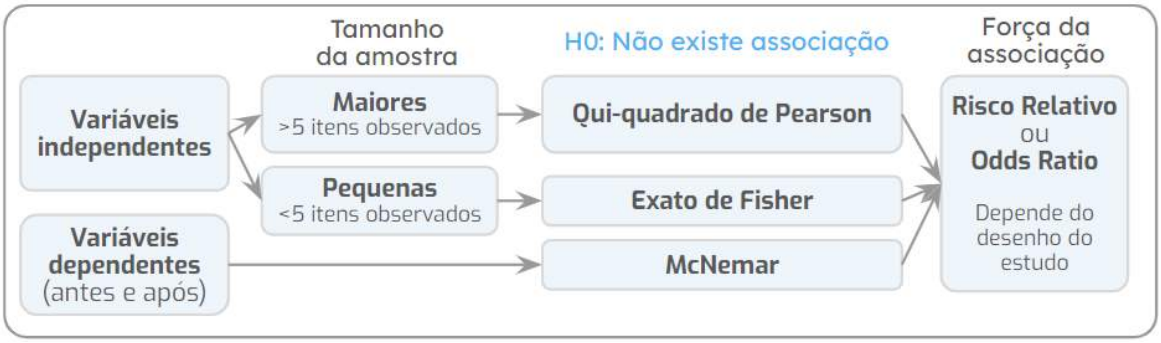


Figura 38: Fluxograma resumo para escolha dos testes de independência

6.2.1 Testes de independência: Qui-quadrado de Pearson

O qui-quadrado de Pearson é um teste estatístico usado para avaliar se a distribuição de frequências observadas em uma tabela de contingência é consistente com a hipótese de independência entre duas variáveis, ou seja, de que uma variável não afeta a outra. O qui-quadrado de Pearson é amplamente utilizado em diversas áreas para avaliar a relação entre duas variáveis categóricas, mas pode ser limitado pelo tamanho da amostra e não ser apropriado para todas as situações. Para aplicar o qui-quadrado de Pearson, são consideradas as seguintes premissas:

- As observações devem ser independentes (variáveis diferentes)
- Cada participante só entra uma vez na amostra
- Permite comparação de N fatores de exposição e desfecho
- Frequência esperada deve ser maior a 5 em 80% das caselas
- Todas as frequências devem ser maior ou igual a 1 em todas as caselas

Se um ou mais destes pressupostos não forem cumpridos, o qui-quadrado de Pearson pode não ser apropriado para avaliar a relação entre as duas variáveis. Nesses casos, pode ser necessário usar outro teste estatístico ou modificar a coleta de dados para atender aos pressupostos do qui-quadrado de Pearson.

Exemplo: Suponha que você queira saber se existe associação estatisticamente significativa entre o sexo dos recém-nascidos e a via de parto (normal ou cesárea).

```
# Remover os registros ausentes de SEXO, TIPO_PARTO
dados2 = dados.dropna(how = 'any', subset=['SEXO', 'TIPO_PARTO'])

# Criar a tabela contingência
tbContigencia = pd.crosstab(dados2.SEXO, dados2.TIPO_PARTO)

# Calculando o qui-quadrado
stat, p, dof, expected = stats.chi2_contingency(tbContigencia)

# Mostrar o valor de 'p' formatado com 3 casas decimais
print(f'{p:.3f}')
```

Resultado: 0,003 → Como o resultado é menor que nível de significância (0,05), rejeita-se a hipótese nula (H0). Ou seja, existe evidências estatísticas que existe associação entre o sexo do recém-nascido e a via de parto.

6.2.2 Testes de independência: Exato de Fisher

O teste exato de Fisher é um teste estatístico usado para avaliar se a distribuição de frequências observadas em uma tabela de contingência é consistente com a hipótese de independência entre as duas variáveis. Ele é geralmente usado quando as frequências observadas são pequenas ou quando a distribuição de frequências esperadas é muito desigual. Para aplicar o teste Exato de Fisher, é necessário cumprir alguns pressupostos:

- As observações devem ser independentes (variáveis diferentes)
- Cada participante só entra uma vez na amostra
- Permite comparação de um fator de exposição e um desfecho
- Frequência esperada pode ser menor que 5
- Frequência observada permite zero

Exemplo: Suponha que você queira saber se existe associação estatisticamente significativa entre a utilização de fórceps e o quadro de *'near miss'* materno.

```
# Filtrar registros removendo os nulos das duas colunas
dados2 = dados.dropna(how = 'any', subset=['FORCEPS', 'CM_NEARMISS'])

# Criar a tabela contingência
tbContigencia = pd.crosstab(dados2.FORCEPS, dados2.CM_NEARMISS)
print(tbContigencia)

# Calculando o Fisher
oddsr, p =stats.fisher_exact(tbContigencia, alternative='two-sided')
# Mostrar o valor de 'p' formatado com 3 casas decimais
print(f'{p:.3f}')
```

Resultado: 1,000 → Como o resultado é maior que nível de significância (0,05), aceita-se a hipótese nula (H0). Ou seja, não existem evidências estatísticas que haja associação entre a utilização de fórceps e o quadro de *'near miss'* materno.

6.2.3 Testes de independência: McNemar

O teste de McNemar é geralmente usado quando as observações são pareadas ou quando cada observação é dividida em duas categorias. Para aplicar o teste McNemar, é necessário cumprir alguns pressupostos:

- As observações devem ser dependentes (a mesma natureza da variável)
- Cada participante só entra uma vez na amostra
- Relacionar o fator de exposição (antes) com o fator de desfecho (após)
- As categorias devem ser idênticas (ter os mesmos valores)

Exemplo: Suponha que você queira saber se existe associação estatisticamente significativa entre os recém-nascidos diagnosticados com baixo Apgar ao 1º minuto de vida em comparação ao 5º minuto de vida.

```
# Importar a biblioteca
import statsmodels.stats.contingency_tables as stsmct

# Filtrar registros removendo os nulos das duas colunas
dados2 = dados.dropna(how = 'any', subset=['BAIXO_APGAR1',
'BAIXO_APGAR5'])

# Criar a tabela contingência
Contingência = pd.crosstab(dados2.BAIXO_APGAR1, dados2.BAIXO_APGAR5)
print(tbContigencia)

# Calculando o McNemar
result = stsmct.mcnemar(tbContigencia, exact=True, correction=True)
```

```
# Mostrar o valor de 'p' formatado com 3 casas decimais
print(f'p-value: {result.pvalue:.3f}')
```

Resultado: <0,001 → Como o resultado é menor que nível de significância (0,05), rejeita-se a hipótese nula (H0). Ou seja, existem evidências estatísticas que haja associação entre as duas variáveis.

6.3 Magnitude ou força da associação

Como vimos, os testes de hipóteses de independência indicam se duas variáveis estão ou não associadas, para determinar a magnitude ou a força da associação entre a variável de exposição (ou intervenção) e a variável de desfecho (resultados) precisamos utilizar outras medidas. Para variáveis categóricas o Risco Relativo e a Razão de Chances são as mais frequentes.

6.3.1 Risco relativo

O risco relativo é a relação entre o risco de ocorrência de um evento em uma determinada população para o risco de ocorrência do mesmo evento em outra população. Ele é usado para avaliar o quanto o risco de ocorrência de um evento é alterado por um determinado fator de exposição em estudos prospectivos. Por exemplo, o risco relativo de desenvolver uma doença em pessoas expostas a um determinado fator de exposição em relação a pessoas não expostas é a medida da associação entre o fator de exposição e o evento.

$$Risco\ Relativo\ = \frac{(Risco\ de\ ocorrência\ do\ evento\ em\ pessoas\ expostas)}{(Risco\ de\ ocorrência\ do\ evento\ em\ pessoas\ não\ expostas)}$$

A partir da tabela de contingência pode-se aplicar a seguinte fórmula:

$$Risco\ Relativo\ = \frac{A\ /\ (A+B)}{C\ /\ (C+D)}$$

O resultado do risco relativo é interpretado de acordo com o seu valor. Quanto maior o valor do risco relativo, maior é a associação entre as duas variáveis. Alguns exemplos de como interpretar o resultado do risco relativo são:

- **Se o risco relativo é menor que 1:** indica que a variável de exposição está associada a um menor risco de ocorrência do evento, geralmente é chamada de fator de proteção.
- **Se o risco relativo é igual a 1:** indica que a variável de exposição não está associada ao risco de ocorrência do evento, ou seja, significa que o risco de ocorrência do evento é o mesmo entre as pessoas expostas e não expostas.

- **Se o risco relativo é maior que 1:** indica que a variável de exposição está associada a um maior risco de ocorrência do evento. Por exemplo, se o risco relativo é de 2, isso significa que o risco de ocorrência do evento é o dobro do que é entre as pessoas não expostas.

É importante lembrar que o risco relativo é uma medida de associação e não de causalidade. Isso significa que ele não prova que a variável de exposição é a causa do evento, mas sim que há uma associação entre as duas variáveis. Para estabelecer a causalidade, são necessários outros tipos de estudo e análises.

Exemplo: Suponha que você já aplicou o teste de independência e confirmou que existe associação estatisticamente significativa entre gestantes com hipertensão e o recém-nascido ser internado na UTI neonatal. A partir desta observação, você deseja saber qual é o risco relativo de um recém-nascido ir para UTI em gestantes com hipertensão quando comparado a gestantes sem hipertensão?

```
# Filtrar registros removendo os nulos das duas colunas
d2 = dados.dropna(how = 'any', subset=['HIPERTENSAO', 'UTI_RN'])

# Criar a tabela contingência
tbContingencia = pd.crosstab(d2.HIPERTENSAO,
d2.UTI_RN).reindex(["Sim", "Não"])[['Sim', 'Não']]

# Mostrar a tabela de contingência
print(tbContingencia)
print()

# N° de casos que que tem o fator de exposição (presente) e o fator
de desfecho (presente)
dados3 = d2[(d2.HIPERTENSAO=='Sim') & (d2.UTI_RN=='Sim')].count()
A = dados3.HIPERTENSAO

# N° de casos que que tem o fator de exposição (presente)
dados3 = d2[(d2.HIPERTENSAO== 'Sim')].count()
AB = dados3.HIPERTENSAO

# N° de casos que que tem o fator de exposição (ausente) e o fator
de desfecho (presente)
dados3 = d2[(d2.HIPERTENSAO=='Não') & (d2.UTI_RN=='Sim')].count()
C = dados3.HIPERTENSAO

# N° de casos que que tem o fator de exposição (ausente) e o fator
de desfecho (presente)
dados3 = d2[(d2.HIPERTENSAO == 'Não')].count()
CD = dados3.HIPERTENSAO
```

```
# Função do risco relativo
resultado = stats.contingency.relative_risk(A, AB, C, CD)

# Mostrar o resultado do risco relativo
rr = resultado.relative_risk
print(f'Risco relativo: {rr:.3f}')

# mostrar o intervalo de confiança a 95%
ci = resultado.confidence_interval(confidence_level=0.95)
print(f'IC 95% inferior: {ci.low:.3f}')
print(f'IC 95% superior: {ci.high:.3f}')
```

Resultado:

Risco relativo: 1.378

IC 95% inferior: 1.020

IC 95% superior: 1.862

Ou seja, o risco relativo de um recém-nascido ir para UTI em gestantes com hipertensão quando comparado a gestantes sem hipertensão é de 1.378 na amostra e o verdadeiro valor populacional pode variar entre 1.020 e 1.862 com 95% de confiança.

6.3.2 Razão de chance

A razão de chance (*odds ratio,* OR) é uma medida usada em estudos epidemiológicos para avaliar a força da associação entre duas variáveis. Diferente do Risco Relativo ela pode ser usada nos três tipos (quanto ao tempo) de estudos epidemiológicos retrospectivo, transversal e prospectivo. A razão de chance é calculada como razão da probabilidade de ocorrência do evento em pessoas expostas à variável de exposição para a probabilidade de ocorrência do evento em pessoas não expostas à variável de exposição. No entanto, é importante lembrar que a razão de chance não leva em conta o tamanho absoluto do risco de ocorrência de um evento.

A partir da tabela de contingência pode-se aplicar a seguinte fórmula:

$$Razão\ de\ chance\ = \frac{A\ x\ D}{B\ x\ C}$$

O resultado da razão de chance é interpretado de acordo com o seu valor da mesma forma como é feito no risco relativo. Quanto maior o valor da razão de chance, maior é a associação entre as duas variáveis:

- **Se a razão de chance é menor que 1:** indica que a variável de exposição está associada a uma menor chance de ocorrência do evento, geralmente é chamada de fator de proteção.

- **Se a razão de chance é igual a 1:** indica que a variável de exposição não está associada à chance de ocorrência do evento, ou seja, significa que a chance de ocorrência do evento é a mesma entre as pessoas expostas e não expostas.
- **Se a razão de chance é maior que 1:** indica que a variável de exposição está associada a uma maior chance de ocorrência do evento. Por exemplo, se a razão de chance é de 2, isso significa que a chance de ocorrência do evento é o dobro do que é entre as pessoas não expostas.

É importante lembrar que a razão de chance é uma medida de associação e não de causalidade. Isso significa que ele não prova que a variável de exposição é a causa do evento, mas sim que há uma associação entre as duas variáveis. Para estabelecer a causalidade, são necessários outros tipos de estudo e análises.

Exemplo: Suponha que você já aplicou o teste de independência e confirmou que existe associação estatisticamente significativa entre gestantes com hipertensão e o recém-nascido ser internado na UTI neonatal. A partir desta observação, você deseja saber qual é a razão de chance (*odds ratio*) de um recém-nascido ir para UTI em gestantes com hipertensão quando comparado a gestantes sem hipertensão?

```
# Filtrar registros removendo os nulos das duas colunas
d2 = dados.dropna(how = 'any', subset=['HIPERTENSAO', 'UTI_RN'])

# Criar a tabela contingência
tbContingencia = pd.crosstab(d2.HIPERTENSAO,
d2.UTI_RN).reindex(["Sim", "Não"])[['Sim', 'Não']]

# Mostrar a tabela de contingência
print(tbContingencia)
print()

# N° de casos que que tem o fator de exposição (presente) e o fator
de desfecho (presente)
dados3 = d2[(d2.HIPERTENSAO=='Sim') & (d2.UTI_RN=='Sim')].count()
A = dados3.HIPERTENSAO

# N° de casos que que tem o fator de exposição (presente) e o fator
de desfecho (ausente)
dados3 = d2[(d2.HIPERTENSAO== 'Sim') & (d2.UTI_RN=='Não')].count()
B = dados3.HIPERTENSAO

# N° de casos que que tem o fator de exposição (ausente) e o fator
de desfecho (presente)
dados3 = d2[(d2.HIPERTENSAO=='Não') & (d2.UTI_RN=='Sim')].count()
C = dados3.HIPERTENSAO
```

```
# N° de casos que que tem o fator de exposição (ausente) e o fator de desfecho (ausente)
dados3 = d2[(d2.HIPERTENSAO == 'Não') & (d2.UTI_RN=='Não')].count()
D = dados3.HIPERTENSAO

# Calcular o Odds de forma simples
odds = (A*D)/(C*B)

# Calcular o intervalo de Confiança com 95% para o Odds
soma = (1/A + 1/B + 1/C + 1/D)
desvio = 1.96 * np.sqrt(soma)
logOdds = np.log(odds)

lower = np.exp(logOdds - desvio)
upper = np.exp(logOdds + desvio)

# Mostrar os valores
print(f'Odds ratio: {odds:.3f}')
print(f'IC 95% inferior: {lower:.3f}')
print(f'IC 95% superior: {upper:.3f}')
```

Resultado:

Risco relativo: 1.465

IC 95% inferior: 1.015

IC 95% superior: 2.115

Ou seja, a razão de chance de um recém-nascido ir para UTI em gestantes com hipertensão quando comparado a gestantes sem hipertensão é de 1.465 na amostra e o verdadeiro valor populacional pode variar entre 1.015 e 2.115 com 95% de confiança.

6.4 Teste de concordância Kappa de Cohen

O Teste Kappa de Cohen é um teste de concordância usado para medir o grau de concordância entre dois observadores ou classificações. Ele é comumente usado em estudos de validação de medidas, onde é importante avaliar se dois observadores estão classificando os mesmos itens de maneira consistente (36).

No exemplo abaixo, dois médicos classificam a severidade de um diagnóstico de um grupo de pacientes em Leve, Moderado e Grave. Observa-se na matriz de confusão feita com os resultados das classificações, que eles concordam nas células azuis e discordam na célula vermelha. O Índice Kappa serve para quantificar o grau de concordância entre os médicos.

		Avaliador 1		
		Leve	Moderado	Grave
Avaliador 2	Leve	**14**	0	0
	Moderado	0	**12**	0
	Grave	**1**	0	**13**

Tabela 4: Matriz de confusão entre avaliadores

É importante lembrar que o índice kappa não é um teste de significância e não pode ser usado para testar hipóteses. Ele é usado apenas para quantificar o grau de concordância entre os observadores. Entretanto, ao fazer a fórmula também é calculado um p-value que pode ser interpretado desta forma:

- **H0:** A concordância foi ao acaso
- **H1:** A concordância não foi ao acaso.
- **Escolher o teste:** Kappa
- **p-value 0,05:**
    - se p-value < 0,05, rejeita-se H0 (A concordância não foi ao acaso)
    - se p-value >= 0,05, aceita-se H0 (A concordância foi ao acaso)

O índice kappa em si, pode variar de -1 a 1, onde -1 indica total discordância, 1 indica total concordância e valores próximos a 0 indicam concordância aleatória. Diversos autores propõem escalas para interpretar qualitativamente o Índice Kappa, neste exemplo apresentamos a escala de Viera e Garrett (2005) (36):

Kappa	Interpretação
< 0	Concordância menor que o acaso
0.01 - 0.20	Concordância insignificante
0.21 - 0.40	Concordância fraca
0.41 - 0.60	Concordância moderada
0.61 - 0.80	Concordância substancial
0.81 - 0.99	Concordância quase perfeita

Tabela 5: Interpretação do Kappa de Cohen (36)

Exemplo: Suponha que você deseje saber se existe concordância entre os diagnósticos realizados por obstetras e pediatras em relação à prematuridade do recém-nascido. Neste exemplo, vamos abstrair as diferenças nos métodos de diagnóstico feito pelos respectivos profissionais.

```
# Importar a biblioteca
import sklearn.metrics as sklm

# Filtrar registros removendo os nulos nas duas colunas
dados2 = dados.dropna(how = 'any', subset=['PREMATURO_OBS',
'PREMATURO_PED'])

# Seleção dos dados das variáveis
juiz1 = dados2.PREMATURO_OBS.values
juiz2 = dados2.PREMATURO_PED.values

# Calcular o Kappa de Cohen
kappa = sklm.cohen_kappa_score(juiz1, juiz2)
# Mostrar o resultado
print('Kappa de Cohen:', kappa.round(3))

# Se desejar fazer um gráfico com a matriz de confusão
matrizconf = sklm.confusion_matrix(juiz1, juiz2)

# Mostrar a matriz de confusão, com o gráfico mapa de calor
sns.heatmap(matrizconf, annot=True, cmap='Blues', fmt="d")
```

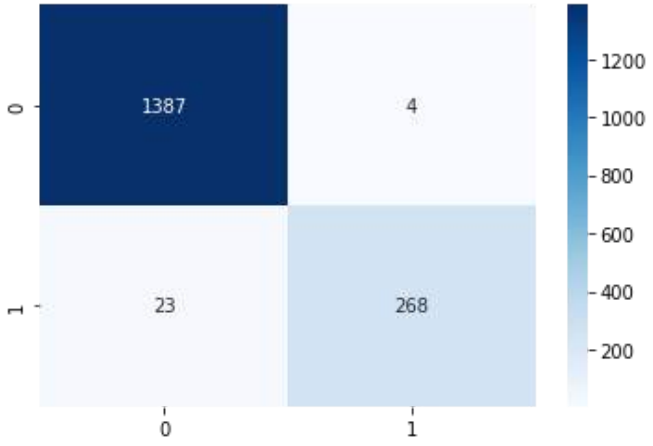


Figura 39: Gráfico de mapa de calor para matriz de confusão

Resultado: Kappa de Cohen: 0,942
Ou seja, existe uma concordância quase perfeita segundo Viera e Garrett (2005) entre os diagnósticos realizados pelos profissionais.

6.5 Bibliotecas e funções utilizadas

Funções da biblioteca Pandas

```
# Importar a biblioteca
import pandas as pd
# Filtrar registros removendo os nulos
dados.dropna()
# Tabela cruzada
pd.crosstab()
```

Funções da biblioteca Scipy

```
# Importar a biblioteca
import scipy.stats as stats
# Qui-quadrado
stats.chi2_contingency()
# Exato de Fisher
stats.fisher_exact()
# Risco relativo
stats.contingency.relative_risk()
# Intervalo de confiança a 95%
resultado.confidence_interval(confidence_level=0.95)
```

Funções da biblioteca Statsmodels

```
# Importar a biblioteca
import statsmodels.stats.contingency_tables as stsmct
# McNemar
result = stsmct.mcnemar()
```

Funções da biblioteca Numpy

```
# Importar a biblioteca
import numpy as np
# Raiz quadrada
np.sqrt()
# Logaritmo
np.log()
# Exponencial
np.exp()
```

Funções da biblioteca Sklearn

```
# Importar as bibliotecas
from sklearn.metrics import sklm
# Kappa de Cohen
sklm.cohen_kappa_score()
```

```
# Matriz de confusão
sklm.confusion_matrix()
```

Funções da biblioteca Seaborn

```
# Bibliotecas Seaborn
import seaborn as sns
# Gráfico Mapa de calor - heatmap
sns.heatmap()
```

Saiba mais sobre as bibliotecas e funções utilizadas

Biblioteca	Função	Link de acesso	Referência
Scipy.stats	chi2_contingency()	Acesse o site	(37)
Scipy.stats	fisher_exact()	Acesse o site	(38)
Scipy.stats	relative_risk()	Acesse o site	(39)
Statsmodels	mcnemar()	Acesse o site	(40)
Scikit-learn	cohen_kappa_score()	Acesse o site	(41)
Scikit-learn	confusion_matrix()	Acesse o site	(42)
Seaborn	heatmap()	Acesse o site	(43)

7. Testes de médias

Os testes de médias são usados para comparar a média de uma amostra com uma média populacional ou com a média de outra amostra. São utilizados quando se tem uma distribuição normal nos dados analisados (20). Eles são utilizados para avaliar se a diferença entre as médias é estatisticamente significativa ou se ela pode ter ocorrido por acaso. Existem vários tipos de testes de médias que são usados em diferentes situações e requerem pressupostos diferentes. O infográfico a seguir pode auxiliar na escolha do teste adequado:

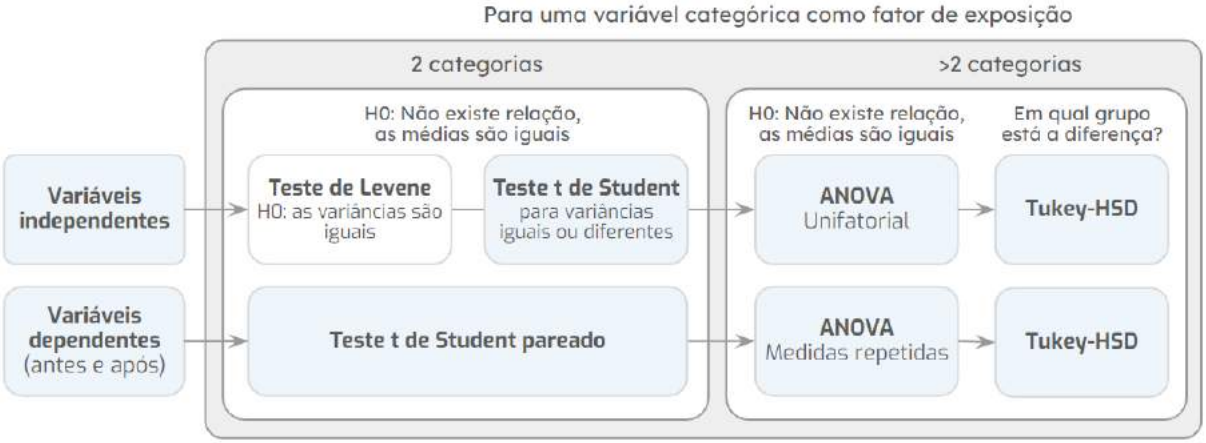


Figura 40: Fluxograma resumo para escolha dos testes de médias (distribuições normais)

7.1 Variável independente com duas categorias

7.1.1 Teste de Levene

O teste de Levene é um teste estatístico usado para avaliar se as variâncias de duas ou mais amostras são iguais. Ele é usado como um pré-teste antes de realizar o teste-t de Student ou o teste ANOVA, que pressupõem que as variâncias das amostras são iguais. Se as variâncias não forem iguais, esses testes podem ter menor poder de detecção e os resultados podem não ser confiáveis. O teste de Levene pressupõe que as amostras são independentes e que seguem uma distribuição normal ou aproximadamente normal.

Ao aplicar e interpretar o teste de Levene é preciso identificar:

- **H0:** Não existe diferença entre as variâncias (as variâncias são iguais)
- **H1:** Existe diferença entre as variâncias (as variâncias são diferentes)
- **Escolher o teste:** Levene
- **p-value 0,05:**
    - se p-value < 0,05, rejeita-se H0 (conclui-se que as variâncias são diferentes)
    - se p-value >= 0,05, aceita-se H0 (conclui-se que as variâncias são iguais)

Exemplo: Suponha que você deseje saber se existe diferença entre as médias do peso ao nascer dos recém-nascidos do sexo feminino e masculino. Para isso, primeiramente, você precisa saber se as variâncias entre essas duas médias são estatisticamente iguais ou diferentes, use o teste de Levene para isso. Logo após, utilize o teste-t de Student para saber se há diferença entre as médias com base no resultado do teste de Levene.

```
# Filtrar registros removendo os nulos
dados2 = dados.dropna(how = 'any', subset=['PESO_NASCER','SEXO'])

# Separar os dois grupos pela categoria
grupo1 = dados2[(dados2.SEXO == 'Feminino')].PESO_NASCER
grupo2 = dados2[(dados2.SEXO == 'Masculino')].PESO_NASCER

# Teste de Levene
# H0 = as variâncias são iguais
resultado = stats.levene(grupo1, grupo2, center = 'mean')
print(f'p-value: {resultado.pvalue:.3f}')

# Guardar o resultado para usar na fórmula do Teste de t Student
varIguais = (resultado.pvalue >= 0.05)
print('Variâncias iguais?', varIguais)
```

Resultado: 0,619 ➔ Como o resultado é maior que nível de significância (0,05), aceita-se a hipótese nula (H0 - As variâncias são iguais). Ou seja, as variâncias entre as médias do peso dos recém-nascidos do sexo feminino é igual as do sexo masculino.

7.1.2 Teste-t de Student

O teste-t de Student é um teste estatístico usado para comparar a média de uma amostra com uma média populacional conhecida. Ele é usado quando não é possível calcular a variância da população diretamente, mas é possível estimá-la a partir da amostra. Ele é baseado na distribuição t de Student, que é uma distribuição normal com uma determinada quantidade de graus de liberdade.

O teste-t de Student também pode ser usado para comparar as médias entre duas categorias de uma variável. Por exemplo, suponha que você tem um conjunto de dados com uma variável "sexo" (masculino ou feminino) e outra variável "altura", e quer saber se há diferença na altura média entre os homens e as mulheres. Neste caso, você pode usar o teste-t de Student para comparar as médias das alturas entre os dois grupos de sexo.

Ao aplicar e interpretar o teste-t de Student é preciso identificar:

- **H0:** Não existe relação entre as variáveis (as médias são iguais)
- **H1:** Existe relação entre as variáveis (as médias são diferentes)
- **Escolher o teste:** t de Student
- **p-value 0,05:**
    - se p-value < 0,05, rejeita-se H0 (conclui-se que as médias são diferentes)
    - se p-value >= 0,05, aceita-se H0 (conclui-se que as médias são iguais)

Exemplo: Continuando o exemplo anterior, você deseja saber se existe diferença entre as médias do peso ao nascer dos recém-nascidos do sexo feminino e masculino. Vamos utilizar o teste-t de Student para saber se há diferença entre as médias com base no resultado do teste de Levene.

```
# Filtrar registros removendo os nulos
dados2 = dados.dropna(how = 'any', subset=['PESO_NASCER','SEXO'])

# Separar os dois grupos pela categoria
grupo1 = dados2[(dados2.SEXO == 'Feminino')].PESO_NASCER
grupo2 = dados2[(dados2.SEXO == 'Masculino')].PESO_NASCER

# Teste de Levene
# H0 = as variâncias são iguais
resultado = stats.levene(grupo1, grupo2, center = 'mean')
print('Teste de Levene p-value: ', f'{resultado.pvalue:.3f}')

# Guardar o resultado para usar na fórmula do Teste de t Student
varIguais = (resultado.pvalue >= 0.05)

# Teste-t de Student
# H0 = As médias são iguais
resultado = stats.ttest_ind(grupo1, grupo2, equal_var=varIguais)
print('Teste-t de médias p-value: ', f'{resultado.pvalue:.3f}')
```

Resultado:

Teste de Levene p-value: 0.619

Teste-t de médias p-value: 0.061

Assim, como o resultado do p-value do teste de médias é maior que o nível de significância (0,05), aceita-se a hipótese nula (H0 - As médias são iguais). Ou seja, as médias dos pesos dos recém-nascidos do sexo feminino é igual as do sexo masculino.

7.2 Variável independente com mais de duas categorias

7.2.1 ANOVA Unifatorial

ANOVA é a sigla em inglês para Análise de Variância, que é um teste estatístico usado para comparar a média de duas ou mais amostras. Ele é usado quando há apenas uma variável independente (também chamada de variável explicativa ou categórica) e uma variável dependente (variável numérica).

O ANOVA unifatorial é baseado na hipótese de que as médias das amostras são iguais. No entanto, é importante lembrar que o ANOVA unifatorial pressupõe que as amostras são independentes e que seguem uma distribuição normal ou aproximadamente normal. Se esses pressupostos não forem cumpridos, outros testes podem ser mais adequados. Além disso, o ANOVA unifatorial não permite identificar qual das amostras é diferente da outra, apenas que há uma diferença estatisticamente significativa entre elas. Para identificar qual das amostras é diferente, é preciso realizar um teste de comparação múltipla, como o teste de Tukey.

Ao aplicar e interpretar o teste ANOVA é preciso identificar:

- **H0:** As médias são iguais
- **H1:** As médias são diferentes
- **Escolher o teste:** ANOVA
- **p-value 0,05:**
    - se p-value < 0,05, rejeita-se H0 (conclui-se que as médias são diferentes)
    - se p-value >= 0,05, aceita-se H0 (conclui-se que as médias são iguais)

Exemplo: Suponha que você deseje saber se existe diferença entre as médias a partir da medição de um equipamento A entre os três subgrupos de idade gestacional (prematuro, termo-precoce e termo).

```
# Filtrar registros removendo os nulos nas duas colunas
dados2 = dados.dropna(how = 'any', subset=['EQUIPAMENTO_A',
'IG_TERMO'])

# Separar os dois grupos pela categoria
grupo1 = dados2[(dados2.IG_TERMO == 'Prematuro')].EQUIPAMENTO_A
grupo2 = dados2[(dados2.IG_TERMO == 'Termo-precoce')].EQUIPAMENTO_A
grupo3 = dados2[(dados2.IG_TERMO == 'Termo')].EQUIPAMENTO_A
```

```
# Teste F - ANOVA
# H0 = As médias são iguais
resultado = stats.f_oneway(grupo1, grupo2, grupo3)
print('Teste-F ANOVA p-value', f': {resultado.pvalue:.3f}')
```

Resultado: Teste-F ANOVA p-value : 0.749
Como o resultado do p-value do teste é maior que o nível de significância (0,05), aceita-se a hipótese nula (H0 - As médias são iguais). Ou seja, as médias das medições do equipamento são estatisticamente iguais entre os três grupos.

7.2.2 Teste post-hoc de Tukey-HSD

O teste de Tukey é um teste de comparação múltipla usado para comparar as médias de duas ou mais amostras que foram previamente agrupadas em um teste ANOVA. Ele é usado para identificar qual das amostras é diferente da outra e é considerado um teste robusto, pois é menos sensível a violações dos pressupostos do ANOVA do que outros testes. Ele pressupõe que as amostras são independentes e que seguem uma distribuição normal ou aproximadamente normal.

Ao aplicar e interpretar o teste Tukey-HSD é preciso identificar:

- **H0:** As médias são iguais (entre os subgrupos analisados)
- **H1:** As médias são diferentes
- **Escolher o teste:** Tukey-HSD
- **p-value 0,05:**
    - se p-value < 0,05, rejeita-se H0 (conclui-se que as médias são diferentes)
    - se p-value >= 0,05, aceita-se H0 (conclui-se que as médias são iguais)

Exemplo: Suponha que você deseje saber se existe diferença entre as médias peso ao ultrassom entre os três subgrupos de idade gestacional (prematuro, termo-precoce e termo). E se existir diferença, precisa-se identificar em quais grupos elas ocorrem.

```
# Importar as bibliotecas
import statsmodels.stats.multicomp as stsmodels
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from statsmodels.stats.anova import AnovaRM

# Filtrar registros removendo os nulos nas duas colunas
dados2 = dados.dropna(how = 'any', subset=['PESO_US', 'IG_TERMO'])

# Separar os dois grupos pela categoria
grupo1 = dados2[(dados2.IG_TERMO == 'Prematuro')].PESO_US
```

```
grupo2 = dados2[(dados2.IG_TERMO == 'Termo-precoce')].PESO_US
grupo3 = dados2[(dados2.IG_TERMO == 'Termo')].PESO_US

# Teste F - ANOVA
# H0 = As médias são iguais
resultado = stats.f_oneway(grupo1, grupo2, grupo3)
print('Teste-F ANOVA p-value', f': {resultado.pvalue:.3f}')

# Teste de Tukey-HSD
# H0: as médias são iguais (entre os subgrupos analisados)

resultado = stsmodels.pairwise_tukeyhsd(endog=dados2.PESO_US,
groups=dados2.IG_TERMO, alpha=0.05)

# Mostrar os resultados
print(resultado)
```

Resultado:
Teste-F ANOVA p-value : <0.001
A partir do resultado rejeita-se a hipótese nula (H0 - As médias são iguais). Ou seja, existe diferença entre as médias dos três grupos. Adicionalmente, a tabela apresentada, mostra cada uma das combinações entre os grupos e o respectivo teste de Tukey-HSD realizado. Na coluna **p-adj** pode ser observado que em todas as combinações possíveis o p-value é significativo, ou seja existem diferenças em todos os subgrupos.

Multiple Comparison of Means - Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
Prematuro	Termo	1158.1471	0.0	1074.217	1242.0772	True
Prematuro	Termo-precoce	829.0515	0.0	737.4748	920.6282	True
Termo	Termo-precoce	-329.0956	0.0	-392.9244	-265.2667	True

Figura 41: Resultados do teste Tukey-HSD para as combinações possíveis

7.3 Variável dependente (avaliação em dois momentos)

7.3.1 Teste-t Pareado

O teste-t pareado, também chamado de teste-t de amostra dependente ou emparelhada, é um teste estatístico usado para comparar a média de duas amostras relacionadas (emparelhadas). Ele é baseado na hipótese de que a diferença entre as médias das amostras é igual a zero. Se a diferença entre as médias foi estatisticamente significativa, conclui-se que a variável independente afeta a variável dependente.

No teste-t de amostra pareada, cada indivíduo é medido/avaliado duas vezes, resultando em pares de observações. São comumente usados em estudos de caso-controle ou desenhos de medidas repetidas. Suponha que você esteja interessado em avaliar a eficácia de uma dieta em um grupo de pacientes, pode-se medir o peso dos pacientes na amostra antes e depois de realizar a dieta e analisar as diferenças usando um teste-t de amostra pareada.

Ao aplicar e interpretar o teste-t pareado é preciso identificar:

- **H0:** As médias são iguais
- **H1:** As médias são diferentes
- **Escolher o teste:** Teste-t pareado
- **p-value 0,05:**
    - se p-value < 0,05, rejeita-se H0 (conclui-se que as médias são diferentes)
    - se p-value >= 0,05, aceita-se H0 (conclui-se que as médias são iguais)

Exemplo: Suponha que você deseje saber se existe diferença entre as médias peso ao ultrassom e peso ao nascer dos recém-nascidos. Neste exemplo, temos as duas medidas de peso para cada recém-nascido, caracterizando uma análise pareada do peso.

```
# Filtrar registros removendo os nulos nas duas colunas
dados2 = dados.dropna(how = 'any', subset=['PESO_US', 'PESO_NASCER'])

# Selecionar os dois grupos
grupo1 = dados2.PESO_US
grupo2 = dados2.PESO_NASCER

# Teste-t de médias para variáveis pareadas
# H0 = As médias são iguais
res = stats.ttest_rel(grupo1, grupo2)
print('Teste-t de média pareado, p-value:', f' {res.pvalue:.3f}')
```

Resultado: Teste-t de médias pareado, p-value : <0,001
A partir do p-value rejeita-se a hipótese nula (H0 - As médias são iguais). Ou seja, as médias dos pesos dos recém-nascidos no ultrassom e ao nascer são diferentes.

7.4 Variável dependente (em mais de dois momentos)

7.4.1 ANOVA de Medidas Repetidas

O ANOVA de medidas repetidas é um teste estatístico usado para avaliar se há diferença estatisticamente significativa entre as médias de duas ou mais amostras que são observadas em diferentes momentos ou em condições diferentes. O ANOVA de medidas repetidas é baseado na hipótese de que as médias das amostras são iguais. Se a diferença entre as médias foi estatisticamente significativa, conclui-se que a variável independente afeta a variável dependente.

Ao aplicar e interpretar o teste ANOVA é preciso identificar:

- **H0:** As médias são iguais
- **H1:** As médias são diferentes
- **Escolher o teste:** ANOVA
- **p-value 0,05:**
    - se p-value < 0,05, rejeita-se H0 (conclui-se que as médias são diferentes)
    - se p-value >= 0,05, aceita-se H0 (conclui-se que as médias são iguais)

Para exemplificar este tópico, serão comparados os pesos durante o ultrassom, ao nascer e à alta para cada recém-nascido. Por sua vez, a função **AnovaRM** da biblioteca **Statsmodels** pressupõe que o conjunto de dados esteja disposto num formato diferente do que trabalhamos até o momento. Especificamente para esta análise, foi criado um novo arquivo onde os registros do mesmo paciente foram repetidos e todas as medições de pesos foram colocados nos respectivos registros diferentes, e por sua vez, criado uma coluna para indicar que tipo de peso continha cada linha.

Adicionalmente, o ANOVA de medidas repetidas não permite identificar qual das amostras é diferente da outra, apenas que há uma diferença estatisticamente significativa entre elas. Para identificar qual das amostras é diferente, é preciso realizar um teste de comparação múltipla, como o teste de Tukey, apresentado anteriormente.

Exemplo: Suponha que você deseje saber se existe diferença entre as médias peso ao ultrassom, peso ao nascer e peso à alta dos recém-nascidos. Neste exemplo, temos as três medidas de peso para cada recém-nascido, caracterizando uma análise pareada do peso.

```
# Ler o arquivo
dados_anova = pd.read_excel('BD_PARTOS_anova.xlsx')

# Fazer o cálculo da Anova para medidas repetidas
anova = AnovaRM(data = dados_anova , depvar = 'PESO', subject =
'PACIENTE', within = ['TIPOPESO']).fit()

# Mostrar tabela com o resultado da AnovaRM
print(anova)

# Teste de Tukey-HSD
# H0: as médias são iguais (entre os subgrupos analisados)

resultado = stsmodels.pairwise_tukeyhsd(endog=dados_anova['PESO'],
groups=dados_anova['TIPOPESO'], alpha=0.05)

# Mostrar a tabela com os resultados entre o subgrupos
print(resultado)
```

Resultado:

Teste-F AnovaRM p-value : <0.001

A partir do resultado rejeita-se a hipótese nula (H0 - As médias são iguais). Ou seja, existe diferença entre as médias dos três grupos (peso no ultrassom, peso ao nascer e peso à alta). Adicionalmente, a tabela apresentada, mostra cada uma das combinações entre os grupos e o respectivo teste de Tukey-HSD realizado. Na coluna **p-adj** pode ser observado que em todas as combinações possíveis o p-value é significativo, ou seja existem diferenças em todos os subgrupos.

Anova

	F Value	Num DF	Den DF	Pr > F
TIPOPESO	62349.4394	2.0000	3188.0000	0.0000

Multiple Comparison of Means - Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
PESO_ALTA	PESO_NASCER	150.0	0.0	102.9388	197.0612	True
PESO_ALTA	PESO_US	-857.246	0.0	-904.3072	-810.1847	True
PESO_NASCER	PESO_US	-1007.246	0.0	-1054.3072	-960.1847	True

Figura 42: Resultados do teste Tukey-HSD para as combinações possíveis

7.5 Bibliotecas e funções utilizadas

Funções da biblioteca Scipy

```
# Importar as Biblioteca
import scipy.stats as stats
# Test t-Student para uma média
stats.ttest_1samp()
# Teste de Levene
stats.levene()
# Teste-t de Student para duas médias
stats.ttest_ind()
# Teste-t Pareado
stats.ttest_rel()
# Teste F - ANOVA
stats.f_oneway()
```

Funções da biblioteca Statsmodels

```
# Importar as Biblioteca
import statsmodels.stats.multicomp as stsmodels
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from statsmodels.stats.anova import AnovaRM
# Teste de Tukey-HSD
stsmodels.pairwise_tukeyhsd()
# Teste F - ANOVA para medidas repetidas
AnovaRM().fit()
```

Saiba mais sobre as bibliotecas e funções utilizadas

Biblioteca	Função	Link de acesso	Referência
Scipy.stats	ttest_1samp()	Acesse o site	(44)
Scipy.stats	levene()	Acesse o site	(45)
Scipy.stats	ttest_ind()	Acesse o site	(46)
Scipy.stats	ttest_rel()	Acesse o site	(47)
Scipy.stats	f_oneway()	Acesse o site	(48)
Statsmodels	MultiComparison()	Acesse o site	(49)
Statsmodels	AnovaRM()	Acesse o site	(50)

8. Testes de medianas

Os testes de medianas são usados para comparar a mediana de duas ou mais amostras e avaliar se elas são estatisticamente significativamente diferentes. Eles são úteis quando a distribuição dos dados não é normal (assimétrica) ou quando os dados têm muitos valores atípicos (outliers).

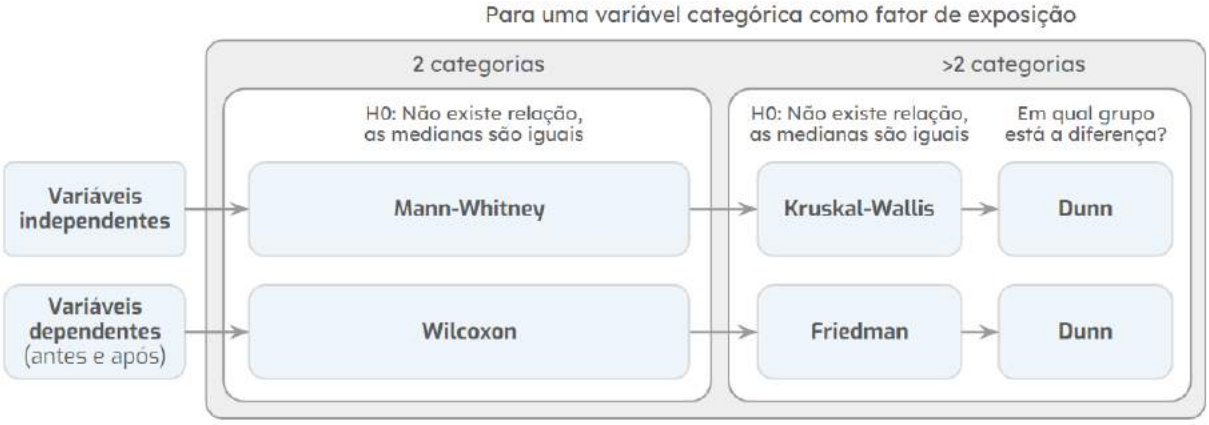


Figura 43: Fluxograma resumo para escolha dos testes de medianas (distribuições assimétricas)

8.1 Variável independente com duas categorias

8.1.1 Teste Mann-Whitney

O teste de Mann-Whitney é um teste estatístico não paramétrico usado para comparar a mediana de duas amostras independentes. Ele é usado quando há duas amostras com distribuições assimétricas. Ele é baseado na hipótese de que as duas amostras têm a mesma distribuição de valores.

Ao aplicar e interpretar o teste Mann-Whitney é preciso identificar:

- **H0:** As medianas são iguais
- **H1:** As medianas são diferentes
- **Escolher o teste:** Mann-Whitney
- **p-value 0,05:**
    - se p-value < 0,05, rejeita-se H0 (conclui-se que as medianas são diferentes)
    - se p-value >= 0,05, aceita-se H0 (conclui-se que as medianas são iguais)

Exemplo: Suponha que você queira saber se existe diferença entre as medianas da idade gestacional entre os recém-nascidos gemelares e não gemelares.

```
# Filtrar registros que sem nulos nas duas colunas
dados2 = dados.dropna(how = 'any', subset=['IG_OBSTETRA', 'GEMELAR'])

# Separar os dois grupos pela categoria
grupo1 = dados2[(dados2.GEMELAR == 'Sim')].IG_OBSTETRA
grupo2 = dados2[(dados2.GEMELAR == 'Não')].IG_OBSTETRA

# Teste de Mann-Whitney
# H0 = as medianas são iguais
res = stats.mstats.mannwhitneyu(grupo1, grupo2)

# Mostrar o resultado
print('Teste de Mann-Whitney - p-value: ', f'{res.pvalue:.3f}')
```

Resultado: Teste de Mann-Whitney - p-value: <0.001
A partir do p-value rejeita-se a hipótese nula (H0 - As medianas são iguais). Ou seja, existe diferença entre as medianas da idade gestacional de bebês gemelares e não gemelares.

8.2 Variável independente com mais de duas categorias

8.2.1 Teste de Kruskal-Wallis

O teste de Kruskal-Wallis é um teste estatístico não paramétrico usado para comparar a mediana de duas ou mais amostras independentes. Ele é usado quando a variável categórica possui mais de 2 categorias. Ele é baseado na hipótese de que as medianas das amostras são iguais e é menos sensível à violação dos pressupostos de distribuição normal e igualdade de variâncias. Além disso, o Kruskal-Wallis não permite identificar qual das amostras é diferente da outra, apenas que há uma diferença estatisticamente significativa entre elas. Para identificar qual das amostras é diferente, é preciso realizar um teste de comparação múltipla, como o teste de Dunn.

Ao aplicar e interpretar o teste Kruskal-Wallis é preciso identificar:

- **H0:** As medianas são iguais
- **H1:** As medianas são diferentes
- **Escolher o teste:** Mann-Whitney
- **p-value 0,05:**
    - se p-value < 0,05, rejeita-se H0 (conclui-se que as medianas são diferentes)
    - se p-value >= 0,05, aceita-se H0 (conclui-se que as medianas são iguais)

Exemplo: Suponha que você queira saber se existe diferença entre as medianas da duração da internação entre os três grupos de recém-nascidos (prematuro, termo-precoce e termo).

```
# Filtrar registros que sem nulos nas duas colunas
dados2 = dados.dropna(how = 'any', subset=['DURACAO_INT',
'IG_TERMO'])

# Separar os dois grupos pela categoria,
gp1 = dados2[(dados2.IG_TERMO == 'Prematuro')].DURACAO_INT
gp2 = dados2[(dados2.IG_TERMO == 'Termo-precoce')].DURACAO_INT
gp3 = dados2[(dados2.IG_TERMO == 'Termo')].DURACAO_INT

# Teste H - Kruskal-Wallis
# H0 = as medianas são iguais
res = stats.mstats.kruskal(gp1.values, gp2.values, gp3.values)
print('Teste de Kruskal-Wallis - p-value: ', f'{res.pvalue:.3f}')
```

Resultado: Teste de Kruskal-Wallis - p-value: <0.001
A partir do p-value rejeita-se a hipótese nula (H0 - As medianas são iguais). Ou seja, existe diferença entre as medianas da idade gestacional de bebês prematuros, termos-precoce e termos.

8.2.2 Teste post-hoc de Dunn

O teste post-hoc de Dunn é um teste estatístico de comparação múltipla usado para comparar a mediana de duas ou mais amostras independentes depois de um teste de Kruskal-Wallis ou ANOVA. Ele é baseado na hipótese de que as medianas das amostras são iguais. Ele é menos sensível à violação dos pressupostos de distribuição normal e igualdade de variâncias do que o teste de Tukey.

Ao aplicar e interpretar o teste Dunn é preciso identificar:

- **H0:** As medianas são iguais (entre os subgrupos analisados)
- **H1:** As medianas são diferentes
- **Escolher o teste:** Dunn
- **p-value 0,05:**
    - se p-value < 0,05, rejeita-se H0 (conclui-se que as medianas são diferentes)
    - se p-value >= 0,05, aceita-se H0 (conclui-se que as medianas são iguais)

Exemplo: A partir do exemplo anterior, suponha que você deseje saber em qual dos subgrupos está a diferença. Para isso é preciso realizar um pós-teste. Para utilizar os testes

post-hoc é preciso instalar a biblioteca no servidor, antes de fazer a importação para em seguida utilizá-la.

```
# Instalar no servidor a biblioteca específica do teste de DUNN
!pip install scikit-posthocs
```

```
# Importar as Biblioteca
import scikit_posthocs as sp

# Filtrar registros que sem nulos nas duas colunas
dados2 = dados.dropna(how = 'any', subset=['DURACAO_INT',
'IG_TERMO'])

# Separar os dois grupos pela categoria,
gp1 = dados2[(dados2.IG_TERMO == 'Prematuro')].DURACAO_INT
gp2 = dados2[(dados2.IG_TERMO == 'Termo-precoce')].DURACAO_INT
gp3 = dados2[(dados2.IG_TERMO == 'Termo')].DURACAO_INT

# Teste H - Kruskal-Wallis
# H0 = as medianas são iguais
res = stats.mstats.kruskal(gp1.values, gp2.values, gp3.values)
print('Teste de Kruskal-Wallis - p-value: ', f'{res.pvalue:.3f}')

# Criando uma lista com os 3 grupos de valores
dados3 = [gp1.values, gp2.values, gp3.values]

# Teste de Dunn test
# H0 = As medianas são iguais
resultado = sp.posthoc_dunn(dados3, p_adjust = 'bonferroni')
print(resultado.round(3))
```

Resultado: Teste de Kruskal-Wallis - p-value: <0.001
A partir do p-value, sabe-se que existe diferença entre as três medianas da idade gestacional. Para identificar em qual grupo está a diferença, analisa-se a tabela par-a-par com o **teste de Dunn** realizado. Note que a combinação linha-2 e coluna-3 (termo-precoce x termo) o **p-value = 0.066,** ou seja, nesta combinação não há diferenças entre as medianas, já nas combinações prematuro x termo-precoce e prematuro x termo existem diferenças (**p-value <0.001**).

```
Teste de Kruskal-Wallis - p-value:  0.000
     1      2      3
1  1.0  0.000  0.000
2  0.0  1.000  0.066
3  0.0  0.066  1.000
```

Figura 44: Resultado do teste de Dunn para as combinações de medianas

8.3 Variável dependente (em dois momentos)

8.3.1 Teste de Wilcoxon

O teste de Wilcoxon é um teste estatístico não paramétrico usado para comparar a mediana de duas amostras dependentes, ou seja, de duas amostras relacionadas (emparelhadas). Ele é baseado na hipótese de que as medianas das amostras são iguais e que as distribuições são assimétricas. No teste de Wilcoxon, cada indivíduo é medido/avaliado duas vezes, resultando em pares de observações. São comumente usados em estudos de caso-controle ou desenhos de medidas repetidas.

Ao aplicar e interpretar o Wilcoxon é preciso identificar:

- **H0:** As medianas são iguais
- **H1:** As medianas são diferentes
- **Escolher o teste:** Wilcoxon
- **p-value 0,05:**
    - se p-value < 0,05, rejeita-se H0 (conclui-se que as medianas são diferentes)
    - se p-value >= 0,05, aceita-se H0 (conclui-se que as medianas são iguais)

Exemplo: Suponha que você queira saber se existe diferença entre as medianas da idade gestacional aferida no momento da internação da gestante (IG_OBSTETRA) e no momento do parto (IG_PEDIATRA).

```
# Filtrar registros removendo os nulos nas duas colunas
dados2=dados.dropna(how='any',subset=['IG_OBSTETRA','IG_PEDIATRA'])
# Separar os dois grupos pela categoria
grupo1 = dados2.IG_OBSTETRA
grupo2 = dados2.IG_PEDIATRA

# Teste de Wilcoxon para variáveis pareadas (antes e após)
# H0 = as medianas são iguais
resultado = stats.wilcoxon(grupo1, grupo2)

# Mostrar o resultado
print('Teste de Wilcoxon - p-value:', f' {resultado.pvalue:.3f}')
```

Resultado: Teste de Wilcoxon - p-value: 0.001
A partir do p-value rejeita-se a hipótese nula (H0 - As medianas são iguais). Ou seja, existe diferença entre as medianas da idade gestacional aferida no momento da internação e no momento do parto.

8.4 Variável dependente (em mais de dois momentos)

8.4.1 Teste de Friedman

O teste de Friedman é um teste estatístico não paramétrico usado para comparar a mediana de duas ou mais amostras dependentes. Ele é baseado na hipótese de que as medianas das amostras são iguais. Este teste é menos sensível à violação dos pressupostos de distribuição normal e igualdade de variâncias. Assim como o teste de Kruskal-Wallis, o teste de Friedman não permite identificar qual das amostras é diferente da outra, apenas que há uma diferença estatisticamente significativa entre elas. Para identificar qual das amostras é diferente, é preciso realizar um teste de comparação múltipla, como o teste de Dunn.

Ao aplicar e interpretar o teste Friedman é preciso identificar:

- **H0:** As medianas são iguais
- **H1:** As medianas são diferentes
- **Escolher o teste:** Friedman
- **p-value 0,05:**
    - se p-value < 0,05, rejeita-se H0 (conclui-se que as medianas são diferentes)
    - se p-value >= 0,05, aceita-se H0 (conclui-se que as medianas são iguais)

Exemplo: Suponha que você queira saber se existe diferença entre as medianas do peso ao nascer medidos em três momentos diferentes (no ultrassom, ao nascer e na alta médica).

```
# Filtrar registros removendo os nulos nas duas colunas
dados2=dados.dropna(how='any',
subset=['PESO_US','PESO_NASCER', 'PESO_ALTA'])

# Separar os 3 grupos pela categoria
gp1 = dados2.PESO_US
gp2 = dados2.PESO_NASCER
gp3 = dados2.PESO_ALTA

# Teste Friedman
# H0 = as medianas são iguais
r = stats.mstats.friedmanchisquare(gp1.values,gp2.values,gp3.values)
# Mostrar o resultado
print('Teste de Friedman - p-value:', f'{r.pvalue:.3f}')
```

Resultado: Teste-t de Friedman - p-value : <0,001

A partir do p-value rejeita-se a hipótese nula (H0 - As medianas são iguais). Ou seja, existe diferença entre as três medianas do peso ao nascer. Entretanto, apenas com este teste não é possível afirmar em qual comparação está a diferença. Para descobrir onde está a diferença, aplica-se o teste post-hoc de Dunn.

Teste post-hoc de Dunn para o teste de Friedman

Exemplo: A partir do exemplo anterior, suponha que você deseje saber em qual dos subgrupos está a diferença, para isso é necessário executar o teste post-hoc de Dunn. Caso ainda não tenha sido executado, é preciso instalar a biblioteca no servidor, antes de fazer a importação para em seguida utilizá-la.

```
# Instalar no servidor a biblioteca específica do teste de DUNN
!pip install scikit-posthocs
# Importar as Biblioteca
import scikit_posthocs as sp

# Filtrar registros removendo os nulos nas duas colunas
dados2=dados.dropna(how='any',
subset=['PESO_US','PESO_NASCER', 'PESO_ALTA'])

# Separar os 3 grupos pela categoria
gp1 = dados2.PESO_US
gp2 = dados2.PESO_NASCER
gp3 = dados2.PESO_ALTA

# Teste Friedman
# H0 = as medianas são iguais
r = stats.mstats.friedmanchisquare(gp1.values,gp2.values,gp3.values)

# Mostrar o resultado
print('Teste de Friedman - p-value:', f'{r.pvalue:.3f}')

# Criando uma lista com os 3 grupos de valores
dados3 = [gp1.values, gp2.values, gp3.values]

# Teste de Dunn test
# H0 = As medianas são iguais
resultado = sp.posthoc_dunn(dados3, p_adjust = 'bonferroni')
print(resultado.round(3))
```

Resultado: Teste de Friedman - p-value: <0.001
A partir do p-value, sabe-se que existe diferença entre as três medianas do peso ao nascer. Para identificar em qual grupo está a diferença, analisa-se a tabela par-a-par com o **teste de Dunn** realizado. Note que em todas as combinações peso ultrassom x peso ao nascer, peso

ultrassom x peso a alta, peso ao nascer x peso a alta rejeita-se a hipótese nula, ou seja existem diferenças entre essas medianas (**p-value <0.001**).

```
Teste de Friedmann - p-value: 0.000
     1    2    3
1  1.0  0.0  0.0
2  0.0  1.0  0.0
3  0.0  0.0  1.0
```

Figura 45: Resultado do teste de Dunn para as combinações de medianas

8.5 Bibliotecas e funções utilizadas

Funções da biblioteca Scipy

```
# importar a biblioteca
import scipy.stats as stats
# Teste de Mann-Whitney
stats.mstats.mannwhitneyu()
# Teste Kruskal-Wallis
stats.mstats.kruskal()
# Teste de Wilcoxon
stats.wilcoxon()
# Teste Friedman
stats.mstats.friedmanchisquare()
```

Funções da biblioteca Scikit_posthocs

```
# importando a biblioteca
import scikit_posthocs as sp
# Teste de Dunn
sp.posthoc_dunn()
```

Saiba mais sobre as bibliotecas e funções utilizadas

Biblioteca	Função	Link de acesso	Referência
Scipy.stats	mannwhitneyu()	Acesse o site	(51)
Scipy.stats	kruskalwallis()	Acesse o site	(52)
Scipy.stats	wilcoxon()	Acesse o site	(53)
Scipy.stats	friedmanchisquare()	Acesse o site	(54)
Statology	dunns-test-python()	Acesse o site	(55)

9. Testes de correlação

Os testes de correlação são usados para verificar se há uma correlação entre duas variáveis quantitativas. Existem vários tipos de correlação, dependendo da natureza das variáveis e da relação entre elas. Alguns exemplos incluem:

- **Correlação linear:** uma correlação linear ocorre quando há uma relação direta ou inversamente proporcional entre as variáveis. Por exemplo, quanto maior o peso, maior a altura (correlação linear positiva) ou quanto menor o preço, maior a demanda (correlação linear negativa).
- **Correlação não linear:** uma correlação não linear ocorre quando há uma relação não proporcional entre as variáveis. Por exemplo, quanto maior o tempo, maior a velocidade inicial (correlação quadrática positiva) ou quanto maior a temperatura, menor a pressão (correlação inversa exponencial).
- **Correlação nula:** uma correlação nula ocorre quando não há relação entre as variáveis. Por exemplo, o número de dedos em uma mão e o número de fios de cabelo (correlação nula).

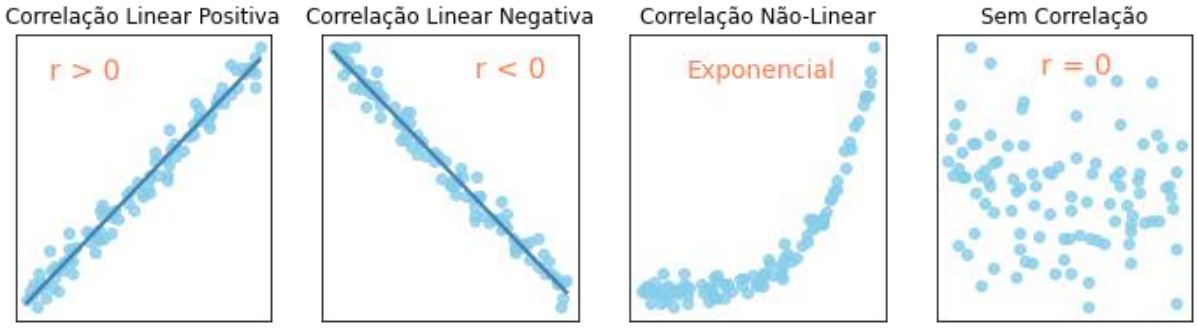


Figura 46: Exemplos de tipos de correlações

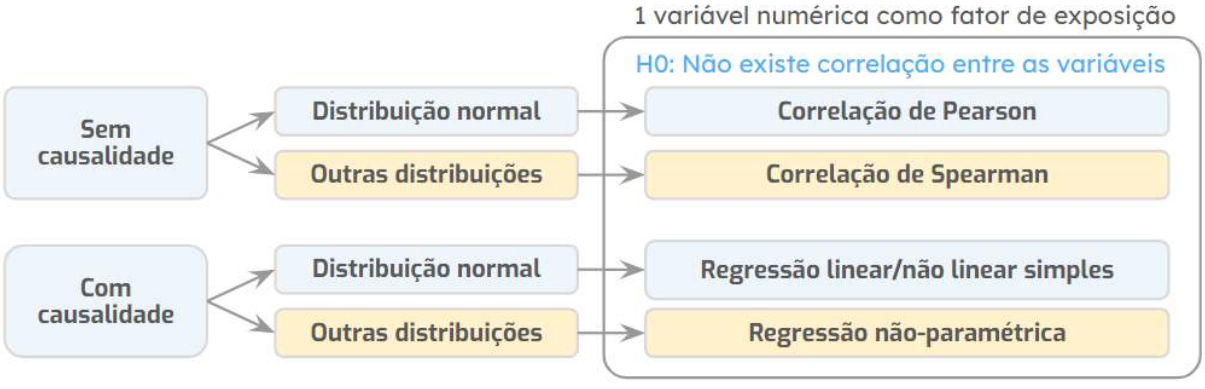


Figura 47: Fluxograma resumo para escolha dos testes de correlações

9.1 Correlação de Pearson

O teste de correlação de Pearson é usado para verificar a força e o tipo de correlação entre duas variáveis numéricas com distribuições normais (56). O coeficiente de correlação de Pearson é o mais comumente usado para medir a correlação linear (positiva ou negativa). Ele varia de -1 a 1, onde -1 indica uma correlação perfeitamente negativa, 1 indica uma correlação perfeitamente positiva e 0 indica ausência de correlação.

Ao aplicar e interpretar o teste de correlação de Pearson é preciso identificar:

- **H0:** Não existe correlação entre as variáveis
- **H1:** Existe correlação entre as variáveis
- **Escolher o teste:** Correlação de Pearson
- **p-value 0,05:**
    - se p-value < 0,05, rejeita-se H0 (conclui-se que existe correlação)
    - se p-value >= 0,05, aceita-se H0 (conclui-se que não existe correlação)

Diversos autores propõem escalas para interpretar qualitativamente o poder de correlação de Pearson, neste exemplo apresentamos a escala de Mukaka (56):

r	Interpretação
0	Sem correlação
0.01 - 0.30	Correlação desprezível
0.31 - 0.50	Correlação fraca
0.51 - 0.70	Correlação moderada
0.71 - 0.90	Correlação forte
0.91 - 1.00	Correlação muito forte

Tabela 6: Interpretação do Kappa de Cohen (56)

Exemplo: Suponha que você deseje saber se existe correlação entre a idade gestacional e o peso ao nascer dos recém-nascidos. Considerando que pelo menos uma das variáveis é uma distribuição normal ou tende para normal (como se comporta o peso), vamos usar o teste de correlação de Pearson.

```
# Filtrar registros removendo os nulos nas duas colunas
dados2 = dados.dropna(how = 'any', subset=['IG_OBSTETRA',
'PESO_NASCER'])
```

```
# Selecionar os dados das variáveis
X = dados2.IG_OBSTETRA.values
Y = dados2.PESO_NASCER.values

# Correlação de Pearson
# H0 não existe correlação entre as variáveis
r, p = stats.pearsonr(X, Y)

# Mostrar o resultado
print('Teste de Correlação de Pearson')
print('Pearson r:', r.round(3))
print('p-value:', p.round(3))

# Mostrar um gráfico de dispersão com as duas variáveis
graf1 = sns.regplot(x=dados2.IG_OBSTETRA, y=dados2.PESO_NASCER)
# Formatar os rótulos dos eixos
graf1.set_xlabel("Idade Gestacional")
graf1.set_ylabel("Peso ao Nascer")
```

Resultado: Teste de Correlação de Pearson

Pearson r: 0.745

p-value: 0.0

A partir do p-value rejeita-se a hipótese nula (H0 - Não existe relação entre as variáveis). Ou seja, existe uma correlação entre as variáveis e com base do valor do r de Pearson (r = 0,745) pode-se considerar com uma correlação forte.

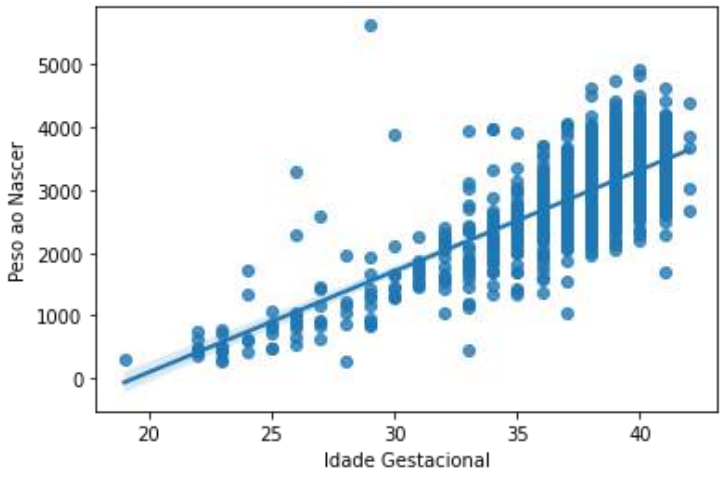


Figura 48: Gráfico de dispersão (correlação) entre a idade gestacional e o peso ao nascer

9.2 Correlação de Spearman

O coeficiente de correlação de Spearman é uma medida de correlação usada quando as variáveis são ordinais ou não seguem uma distribuição normal, ou seja, é recomendado para as distribuições assimétricas (56).

O coeficiente de Spearman é calculado como o coeficiente de correlação entre as classificações das variáveis em ordem. Ele varia de -1 a 1, onde -1 indica uma correlação perfeitamente negativa, 1 indica uma correlação perfeitamente positiva e 0 indica ausência de correlação. Ao aplicar e interpretar o teste de correlação de Spearman é preciso identificar:

- **H0:** Não existe correlação entre as variáveis
- **H1:** Existe correlação entre as variáveis
- **Escolher o teste:** Correlação de Spearman
- **p-value 0,05:**
    - se p-value < 0,05, rejeita-se H0 (conclui-se que existe correlação)
    - se p-value >= 0,05, aceita-se H0 (conclui-se que não existe correlação)

Para a interpretação qualitativa é utilizado as mesmas tabelas utilizadas para interpretar o coeficiente de Pearson. Em resumo, o coeficiente de Spearman não é uma medida de correlação linear, mas sim uma medida de correlação entre variáveis ordinais. Ele é usado quando o coeficiente de Pearson não pode ser aplicado devido às condições de distribuição das variáveis.

Exemplo: Suponha que você deseje saber se existe correlação entre a idade gestacional e a duração da internação dos recém-nascidos. Considerando que as variáveis são assimétricas, vamos usar o teste de correlação de Spearman.

```
# Filtrar registros removendo os nulos nas duas colunas
dados2 = dados.dropna(how = 'any', subset=['IG_PEDIATRA',
'DURACAO_INT'])

# Selecionar os dados das variáveis
X = dados2.IG_PEDIATRA.values
Y = dados2.DURACAO_INT.values

# Correlação de Spearman
# H0 não existe correlação entre as variáveis
r, p = stats.spearmanr(X, Y)

# Mostrar o resultado
print('Teste de Correlação de Spearman')
print('Spearman r:', r.round(3))
```

```
print('p-value:', p.round(3))

# Mostrar um gráfico de dispersão com as duas variáveis
graf1 = sns.regplot(x=dados2.IG_PEDIATRA, y=dados2.DURACAO_INT)
# Formatar os rótulos dos eixos
graf1.set_xlabel("Idade Gestacional")
graf1.set_ylabel("Duração da Internação")
```

Resultado: Teste de Correlação de Spearman

Spearman r: -0.159

p-value: 0.0

A partir do p-value rejeita-se a hipótese nula (H0 - Não existe relação entre as variáveis). Ou seja, existe uma correlação entre as variáveis e com base do valor do r de Spearman (r = -0,159) pode-se considerar como uma correlação negativa (inversamente proporcional) e desprezível.

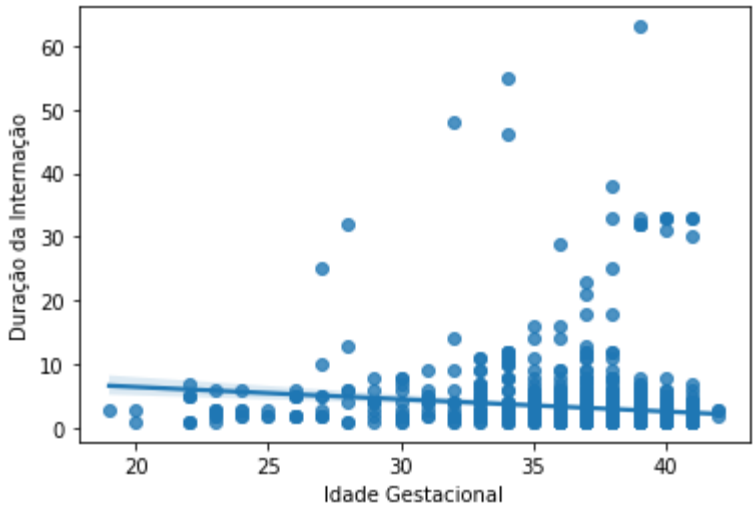


Figura 49: Gráfico de dispersão entre a idade gestacional e a duração da internação

9.3 Correlação de Kendall

O coeficiente de correlação de Kendall, também conhecido como tau de Kendall, é uma medida de correlação não paramétrica usada quando as variáveis são ordinais ou não seguem uma distribuição normal e dependentes (pareadas). Ele é semelhante ao coeficiente de Spearman, que também é usado para variáveis ordinais, mas é calculado de maneira diferente. Ele varia de -1 a 1, onde -1 indica uma correlação perfeitamente negativa, 1 indica uma correlação perfeitamente positiva e 0 indica ausência de correlação.

O tau de Kendall é um bom indicador da consistência e precisão das observações e pode ser usado para testar hipóteses sobre a existência de uma correlação entre as variáveis. No

entanto, ele não é tão preciso quanto o coeficiente de Spearman para detectar correlações fracas e pode ser afetado por outliers (observações atípicas).

Ao aplicar e interpretar o teste de correlação de Kendall é preciso identificar:

- **H0:** Não existe correlação entre as variáveis
- **H1:** Existe correlação entre as variáveis
- **Escolher o teste:** Correlação de Kendall
- **p-value 0,05:**
    - se p-value < 0,05, rejeita-se H0 (conclui-se que existe correlação)
    - se p-value >= 0,05, aceita-se H0 (conclui-se que não existe correlação)

Exemplo: Suponha que você deseje saber se existe correlação entre o escore de Apgar aferido ao 1º minuto de vida em relação ao 5º minuto de vida dos recém-nascidos. Considerando que as variáveis são assimétricas e ordinais, vamos usar o teste de correlação de Kendall.

```
# Filtrar registros removendo os nulos nas duas colunas
dados2 = dados.dropna(how = 'any', subset=['APGAR1', 'APGAR5'])

# Selecionar os dados das variáveis
X = dados2.APGAR1.values
Y = dados2.APGAR5.values

# Correlação de Kendall
# H0 não existe correlação entre as variáveis
r, p = stats.kendalltau(X, Y)

# Mostrar o resultado
print('Teste de Correlação de Kendall')
print('Kendall r:', r.round(3))
print('p-value:', p.round(3))

# Mostrar um gráfico de dispersão com as duas variáveis
graf1 = sns.regplot(x=dados2.APGAR1, y=dados2.APGAR5)
# Formatar os rótulos dos eixos
graf1.set_xlabel("Escore de Apgar ao 1º minuto")
graf1.set_ylabel("Escore de Apgar ao 5º minuto")
```

Resultado: Teste de Correlação de Kendall

Kendall r: 0.617

p-value: 0.0

A partir do p-value rejeita-se a hipótese nula (H0 - Não existe relação entre as variáveis). Ou seja, existe uma correlação entre as variáveis e com base do valor (r = 0,617) pode-se considerar com uma correlação moderada.

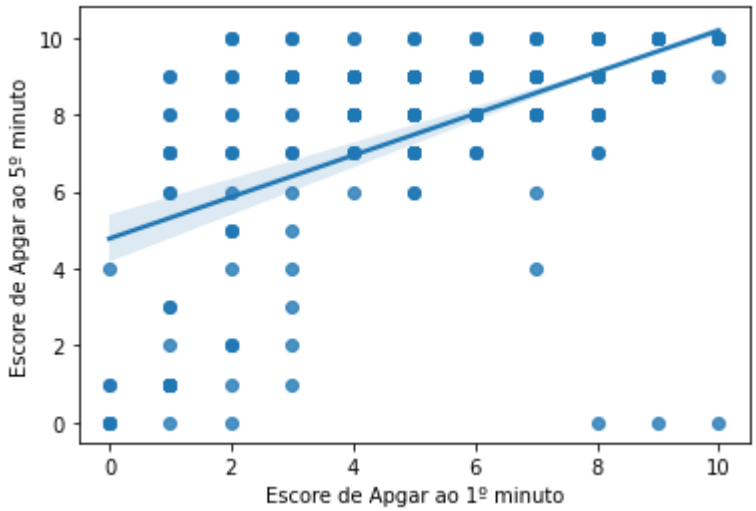


Figura 50: Gráfico de dispersão entre o Escore de Apgar do 1º minuto e do 5º minuto

9.4 Coeficiente de Correlação Intraclasse

O coeficiente de correlação intraclasse (CCI) é um teste de correlação usado para medir a consistência e a precisão de medidas repetidas em uma amostra. Ele é usado quando há múltiplos observadores ou medidas em cada elemento da amostra. Ele é expressado em uma escala de -1 a 1, onde -1 indica uma correlação perfeitamente negativa, 1 indica uma correlação perfeitamente positiva e 0 indica ausência de correlação. Valores próximos a 1 indicam alta consistência e precisão das medidas repetidas.

O CCI é um bom indicador da qualidade de medidas repetidas e pode ser usado para avaliar a validade de testes e medidas. No entanto, ele não é um teste de significância e não pode ser usado para testar hipóteses.

Algumas funções ao calcular o CCI retorna uma tabela com um conjunto de valores, a escolha do mais adequado depende da análise do contexto do pesquisador. Na biblioteca e função que usaremos neste módulo, o resultado nos trará um conjunto com seis CCIs diferentes, que devem ser analisados conforme as seguintes suposições (57):

- **ICC1:** cada um dos pacientes foi medido apenas por um subconjunto de avaliadores, e não é o mesmo subconjunto de avaliadores que mediu cada paciente.
- **ICC2:** cada um dos pacientes foi medido apenas por um subconjunto de avaliadores, mas é o mesmo subconjunto de avaliadores que mediu cada paciente.
- **ICC3:** cada um dos pacientes foi medido por toda a população de avaliadores (portanto, não precisamos levar em consideração a variabilidade entre avaliadores)

- **ICC1k, ICC2k e ICC3k:** a confiabilidade dos k avaliadores (média) quando trabalham em grupo (enquanto ICC1, ICC2 e ICC3 representam a confiabilidade dos avaliadores como indivíduos). Esses valores sempre serão maiores, porque vários avaliadores trabalhando juntos sempre fornecerão um resultado estatisticamente mais confiável. Exemplo para avaliar confiabilidade do estudo como um todo, contando com todos os avaliadores do estudo, recomenda-se o ICC2k.

Diversos autores propõem escalas para interpretar qualitativamente os valores de CCI, neste exemplo apresentamos as escalas de Cicchetti (58) e de Koo and Li (59):

CCI	Cicchetti (1994)
0.75 a 1.00	Excelente
0.60 a 0.75	Boa
0.40 a 0.60	Razoável
< 0.40	Pobre

CCI	Koo and Li (2016)
0.90 a 1.00	Excelente
0.75 a 0.90	Boa
0.50 a 0.75	Moderada
< 0.50	Pobre

Tabela 7: Interpretações para os valores do Coeficiente de Correlação Intraclasse

Para exemplificar este tópico, serão comparadas as medições realizadas com dois equipamentos diferentes (A e B) para cada recém-nascido. Por sua vez, a função **intraclass_corr** da biblioteca **Pingouin** pressupõe que o conjunto de dados esteja disposto num formato diferente do que trabalhamos até o momento. Então, especificamente para esta análise, foi criado um novo arquivo onde os registros do mesmo paciente foram repetidos e todas as medições dos equipamentos foram colocados nos respectivos registros diferentes, e por sua vez, criado uma coluna para indicar se a medição foi realizada pelo equipamento A ou B, conforme apresentado na figura a seguir.

	COD_PACIENTE	EQUIPAMENTO	VALOR
0	1	A	2120
1	2	A	2095
2	3	A	2130
3	4	A	2050
4	5	A	2134
...	...	...	...
1573	785	B	4035
1574	786	B	4040
1575	787	B	4050
1576	788	B	4070
1577	789	B	4085

1578 rows × 3 columns

Figura 51: Disposição dos dados na tabela para cálculo do ICC

Exemplo: Suponha que você deseje saber se existe concordância entre as medições realizadas com dois equipamentos diferentes (A e B). Para o exemplo vamos usar o ICC1 (single rates absolute): um grupo de equipamentos (equipamento A e B) não representa toda a população de equipamentos existentes que poderiam ser utilizados para avaliar o paciente, queremos saber a confiabilidade entre as medições destes equipamentos específicos.

```
# instalar a biblioteca pingouin
!pip install pingouin

# importar a biblioteca
import pingouin as pg

# Escolher o arquivo
arquivo='/content/drive/MyDrive/Colab Notebooks/IADS/BD_PARTOS_ICC_EQ.xlsx'

# Ler o arquivo
dados = pd.read_excel(arquivo)

# Calcular o Coeficiente de Correlação Intraclasse
ICC = pg.intraclass_corr(data=dados, targets='COD_PACIENTE',
raters='EQUIPAMENTO', ratings='VALOR')

# Mostrar o ICC
ICC
```

Resultado: Coeficiente de Correlação Intraclasse, ICC1 = 0.95, IC95% [0.95, 0.96]
A partir dos resultados pode-se concluir que existe uma concordância excelente entre os equipamentos A e B.

	Type	Description	ICC	F	df1	df2	pval	CI95%
0	ICC1	Single raters absolute	0.952811	41.383137	788	789	0.0	[0.95, 0.96]
1	ICC2	Single random raters	0.953015	50.704755	788	788	0.0	[0.92, 0.97]
2	ICC3	Single fixed raters	0.961319	50.704755	788	788	0.0	[0.96, 0.97]
3	ICC1k	Average raters absolute	0.975836	41.383137	788	789	0.0	[0.97, 0.98]
4	ICC2k	Average random raters	0.975942	50.704755	788	788	0.0	[0.96, 0.98]
5	ICC3k	Average fixed raters	0.980278	50.704755	788	788	0.0	[0.98, 0.98]

Figura 52: Resultados para o cálculo do ICC entre os equipamentos A e B

9.5 Bibliotecas e funções utilizadas

Funções da biblioteca Scipy

```
# importar a biblioteca
import scipy.stats as stats
# Correlação de Pearson
stats.pearsonr()
# Correlação de Spearman
stats.spearmanr()
# Correlação de Kendall
stats.kendalltau()
```

Funções da biblioteca Seaborn

```
# Bibliotecas Seaborn
import seaborn as sns
# Gráfico de dispersão
sns.regplot()
```

Funções da biblioteca Pingouin

```
# Bibliotecas pingouin
import pingouin as pg
# Coeficiente de Correlação Intraclasse
pg.intraclass_corr()
```

Saiba mais sobre as bibliotecas e funções utilizadas

Biblioteca	Função	Link de acesso	Referência
Scipy.stats	pearsonr()	Acesse o site	(60)
Scipy.stats	spearmanr()	Acesse o site	(61)
Scipy.stats	kendalltau()	Acesse o site	(62)
Seaborn	regplot()	Acesse o site	(63)
Statology	intraclass_corr()	Acesse o site	(64)

10. Estilizando gráficos

Como vimos nos capítulos anteriores, existem várias ferramentas e técnicas para visualizar dados, incluindo gráficos de barras, gráficos de linhas, gráficos de dispersão, gráficos de setores, histogramas, *boxplots*, mapas e muitos outros. Para escolher o gráfico mais adequado para visualizar os dados, é importante considerar o tipo de dados e o objetivo da análise. Além disso, é importante verificar se o gráfico está legível e facilita a compreensão dos dados. Neste capítulo final, optamos por apresentar alguns exemplos de formatação de gráficos, além da utilização de algumas outras funções gráficas ainda não apresentadas.

10.1 Rótulos dos eixos e títulos

Em Python, é possível alterar os rótulos dos eixos dos gráficos e dos títulos usando as bibliotecas de visualização de dados, como o Matplotlib e o Seaborn. No Matplotlib, por exemplo, para alterar os rótulos dos eixos, você pode usar as funções **xlabel** e **ylabel**, que permitem definir os rótulos para o eixo x e y, respectivamente. Para alterar o título do gráfico, você pode usar a função **title**. No Seaborn, você pode usar as funções **set_xlabel**, **set_ylabel** e **set_title** para definir os rótulos dos eixos e o título do gráfico, respectivamente. Em geral, na programação existem diversas maneiras de se fazer algo que se deseja, além de que cada biblioteca pode implementar a funcionalidade com alguma especificidade, por isso, procure sempre consultar a documentação oficial de cada biblioteca utilizada.

Exemplo: Ajustar os eixos x, y e o respectivo título do gráfico.

```
# Histograma do peso ao nascer
graf1 = sns.histplot(dados.PESO_NASCER, kde=True, linewidth=0)
graf1.set_title('Histograma do Peso ao Nascer')
graf1.set_xlabel('Peso ao Nascer')
graf1.set_ylabel('Quantidade')
```

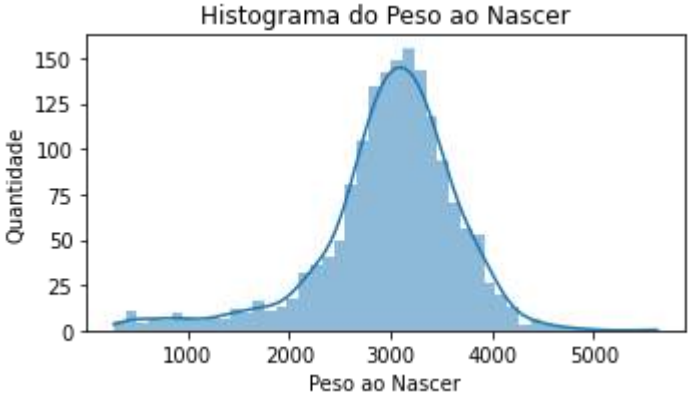


Figura 53: Histograma do peso ao nascer

10.2 Definindo cores

As bibliotecas gráficas do Python como Matplotlib (65), Seaborn (66), Plotly, entre outras, suportam uma variedade de formatos de cores para personalização de gráficos. A seguir estão alguns exemplos de como as corem podem ser especificadas em diversas bibliotecas:

- **Nome da cor:** especificar o nome de uma cor como *"red"* ou *"blue"*.
- **Abreviação hexadecimal:** usar um código hexadecimal de seis dígitos, como "#FF0000" para vermelho ou "#00FF00" para verde.
- **Abreviação RGB:** usar uma tupla de três números que representam as quantidades de vermelho, verde e azul, respectivamente, em uma escala de 0 a 1. Por exemplo: (1, 0, 0) para vermelho e (0, 1, 0) para verde.
- **Abreviação RGBA:** é semelhante ao formato RGB, mas com um quarto valor que representa a porcentagem de transparência desejada.
- **Nome CSS:** pode-se usar nomes de cores CSS como *"aquamarine"* ou **"gold"**.

Exemplo: Ajustar a cor do gráfico.

```
# Histograma da idade gestacional
graf1 = sns.histplot(dados.IG_OBSTETRA, bins=22, color='coral')
graf1.set_title('Histograma da Idade Gestacional')
graf1.set_xlabel('Semanas')
graf1.set_ylabel('Quantidade')
```

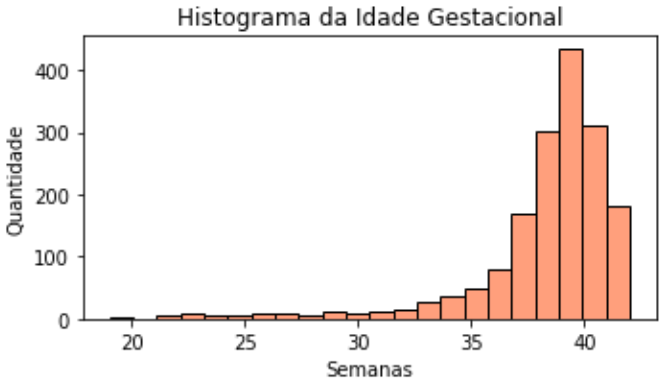


Figura 54: Histograma da idade gestacional

Exemplo: Escolhendo cores para cada faixa de valores de um gráfico.

```
# Tabela resumo com a contagem das categorias
tabela = dados.IG_TERMO.value_counts()
```

```
# Definir conjunto de cores
cores = ['lightblue', 'pink', 'moccasin']

# Gráfico de setores, considerando o conjunto de cores definido
graf2 = tabela.plot.pie(autopct='%1.1f%%', colors = cores)
graf2.set_title('Taxas de prematuridade e termos')
graf2.set_ylabel('')
```

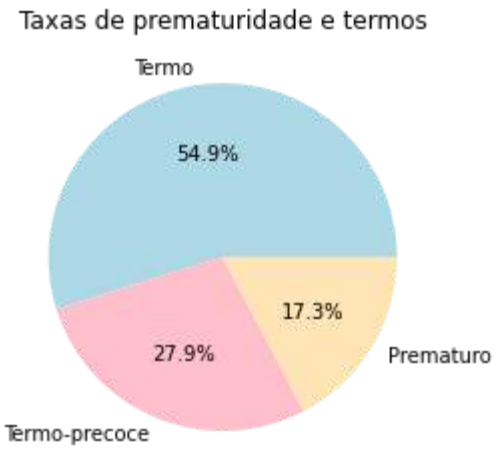


Figura 55: Gráfico de setor para prematuros, termo-precoce e termo

Algumas bibliotecas como a Seaborn e Plotly, por exemplo, também oferecem uma variedade de opções de cores, incluindo paletas de cores pré-definidas que podem ser usadas para criar gráficos com cores consistentes e atraentes. Algumas das paletas de cores mais conhecidas incluem a paleta de cores categóricas "Set1", a paleta de cores sequenciais "Blues", e a paleta de cores divergentes "RdBu". As paletas de cores do Seaborn podem ser facilmente utilizadas em gráficos por meio do argumento "palette" (66).

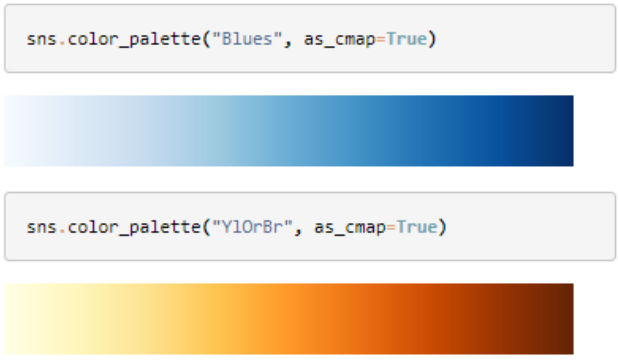


Figura 56: Exemplos de paletas de cores do Seaborn. Fonte: (66)

Em geral, a escolha das cores depende do contexto do gráfico e do objetivo da visualização. É importante escolher cores que sejam claras, distintas e fáceis de interpretar para o público-alvo.

Figura 57: Exemplo de nomes das cores CSS do Matplotlib - Fonte: Matplotlib (65)

10.3 Formatando fontes

Em Python, podemos formatar as fontes dos textos dos gráficos usando as mesmas bibliotecas Matplotlib e o Seaborn. Por exemplo, no Matplotlib, você pode usar as funções **xlabel**, **ylabel** e **title** e no Seaborn pode fazer uso das **set_xlabel**, **set_ylabel** e **set_title** todas com utilização de forma similares. As funções **xticks** e **yticks** permitem formatar as fontes dos números dos eixos, respectivamente.

Exemplo: Formatando todos os textos de um gráfico.

```
# Histograma da idade gestacional
graf1 = sns.histplot(dados.IG_PEDIATRA, bins=22,
color='yellowgreen')

# Formatando o título
graf1.set_title('Histograma da Idade Gestacional', fontsize=14,
fontweight='bold', color='green')

# Formatando os rótulos dos eixos
graf1.set_xlabel('Semanas', fontsize=14, fontweight='bold',
color='steelblue')
graf1.set_ylabel('Quantidade', fontsize=14, fontweight='bold',
color='steelblue')

# Formatando os ticks (números dos eixos)
plt.xticks(fontsize=12, fontweight='bold', color='coral')
plt.yticks(fontsize=12, fontweight='bold', color='coral')
```

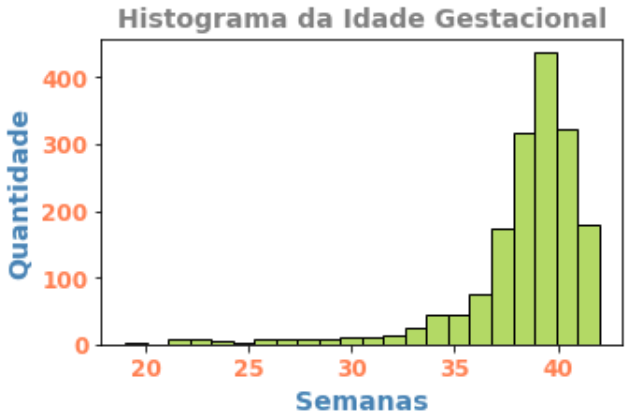


Figura 58: Histograma com formatação dos textos

10.4 Incluindo anotações e elementos nos gráficos

Em alguns casos, precisamos incluir alguma anotação, elemento ou destaque sobre um gráfico já realizado em Python. É possível adicionar linhas horizontais e verticais com as funções **axhline** e **axvline**, respectivamente. Para adicionar uma anotação em um ponto específico do gráfico, você pode usar a função **annotate** ou a função **text**. Outro recurso bem utilizado é destacar uma faixa de valores do gráfico com as funções **axhspan** e **axvspan**. Observe que existem muitas outras formas e funções em cada biblioteca para adicionar linhas, anotações e outros elementos em diferentes tipos de gráficos, sempre consulte a documentação oficial para saber mais detalhes.

Exemplo: Incluindo linha e texto no gráfico de histograma.

```
# Criando o gráfico
graf1 = sns.histplot(dados.IG_OBSTETRA, binwidth=1)

# Incluindo uma linha no gráfico (axvline, axhline)
graf1.axvline(36, color='red', linestyle='--', linewidth=1.5)

# Incluindo um texto e uma seta no gráfico
graf1.annotate('36 semanas', color='red', xytext=(27, 300),
fontsize=11, xy=(36, 200), arrowprops=dict(color='red',
shrink=0.1))

# Definindo os valores do eixo x
graf1.set_xticks([20,25,30,35,40])

# Formatando os rótulos do eixos x e y
graf1.set_xlabel('Idade Gestacional', fontsize=12)
graf1.set_ylabel('Quantidade', fontsize=12)
```

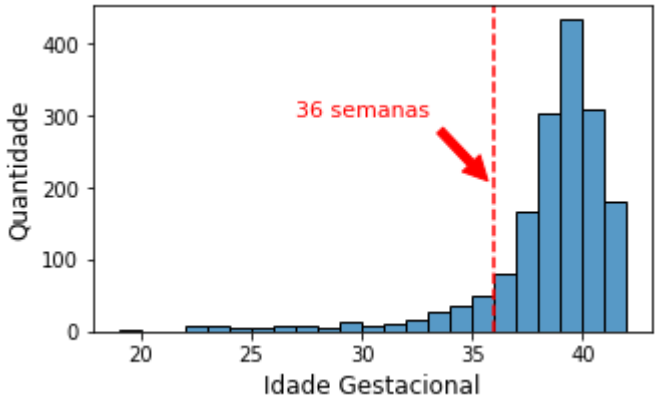


Figura 59: Histograma da idade gestacional com linha e anotações

Exemplo: Incluindo uma faixa e texto no gráfico de histograma.

```
# Criando o gráfico
graf1 = sns.histplot(dados.IG_OBSTETRA, binwidth=1)

# Incluindo uma faixa no gráfico (axhspan, axvspan)
plt.axvspan(36, 38, alpha=0.5, color='gold')

# Incluindo um texto e uma seta no gráfico
graf1.annotate('Termo-precoce', color='gray', xytext=(25, 300),
fontsize=11, xy=(36, 300), arrowprops=dict(color='gray',
shrink=0.1))
```

```
# Definindo os valores do eixo x
graf1.set_xticks([20,25,30,35,40])

# Formatando os rótulos do eixos x e y
graf1.set_xlabel('Idade Gestacional', fontsize=12)
graf1.set_ylabel('Quantidade', fontsize=12)
```

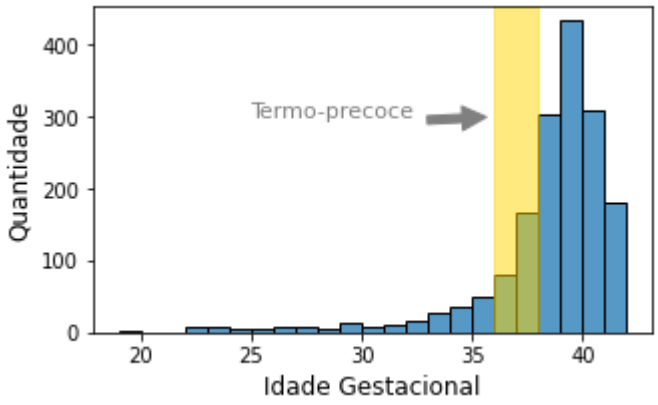


Figura 60: Histograma da idade gestacional com faixa e anotações

10.5 Múltiplos gráficos em uma figura

Usando a biblioteca Matplotlib, podemos criar múltiplos gráficos em uma única figura. A função **subplots** permite criar uma figura e com vários eixos (**subplots**) dentro dela e preencher cada subplot com o gráfico desejado (67).

Exemplo: Criando uma figura com 2 subplots (para dois gráficos).

```
# Mostrar vários gráficos em uma única figura
# figsize = tamanho da área da figura
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 4))

# Colocar aqui o código para cada gráfico (indicado o ax1 e ax2)

# ajustar o layout
plt.tight_layout()
```

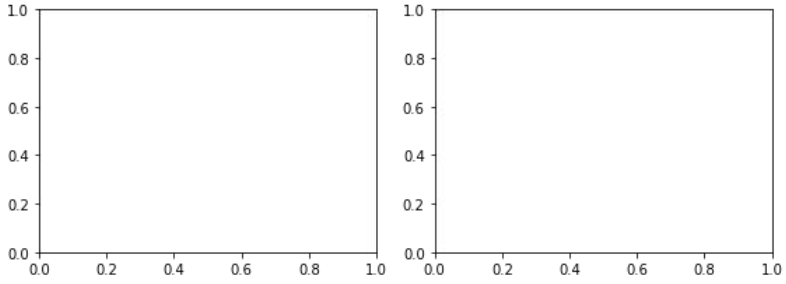


Figura 61: Figura com eixos (subplots) para dois gráficos

Exemplo: Criando uma figura com dois subplots com um histograma e um boxplot.

```
# Mostrar vários gráficos em uma única figura
# figsize define o tamanho da área da figura
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 4))

# histograma
graf1 = sns.histplot(dados.PESO_NASCER, ax=ax1, linewidth=0)
graf1.set_xlabel('Peso ao nascer', fontsize=12)
graf1.set_ylabel('Quantidade', fontsize=12)

# Boxplot
graf1 = sns.boxplot(data=dados, y='PESO_NASCER', x='TIPO_PARTO',
ax=ax2, palette='Blues', linewidth=1)
graf1.set_xlabel('Tipo de parto', fontsize=12)
graf1.set_ylabel('Peso ao nascer', fontsize=12)

# Ajustar o layout
plt.tight_layout()
```

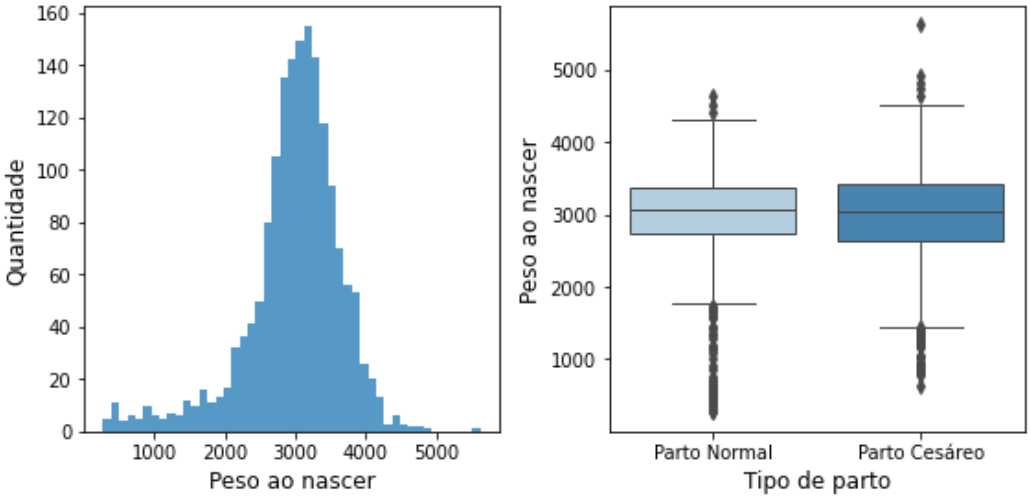


Figura 62: Figura com dois gráficos (histograma e boxplot)

Exemplo: Criando uma figura múltipla com seis gráficos.

```
# Mostrar vários gráficos em uma única figura
# figsize define o tamanho da área da figura
# subplots(3, 2) = 3 linhas e 2 colunas
fig, ((ax1, ax2), (ax3, ax4), (ax5, ax6)) = plt.subplots(3, 2,
figsize=(8, 8))
```

```
# Histograma GESTACOES
graf1 = sns.histplot(dados.GESTACOES, kde=True, linewidth=0,
bins=15, ax=ax1)
graf1.set_xlabel('N° de Gestações', fontsize=12)
graf1.set_ylabel('Quantidade', fontsize=12)

# Histograma PESO_NASCER
graf2 = sns.histplot(dados.PESO_NASCER, kde=True, linewidth=0,
ax=ax2)
graf2.set_xlabel('Peso ao Nascer', fontsize=12)
graf2.set_ylabel('Quantidade', fontsize=12)

# Histograma IG_OBSTETRA
graf3 = sns.histplot(dados.IG_OBSTETRA, kde=True, linewidth=0,
bins=22, ax=ax3)
graf3.set_xlabel('Idade Gestacional', fontsize=12)
graf3.set_ylabel('Quantidade', fontsize=12)

# Boxplot PESO_NASCER x TIPO_PARTO
graf4 = sns.boxplot(data=dados, y='PESO_NASCER', x='TIPO_PARTO',
linewidth=1, ax=ax4, palette='Blues')
graf4.set_xlabel('Tipo de parto', fontsize=12)
graf4.set_ylabel('Peso ao nascer', fontsize=12)

# Dispersão x="IG_OBSTETRA", y="IG_PEDIATRA"
graf5 = sns.scatterplot(data=dados, x="IG_OBSTETRA",
y="IG_PEDIATRA", ax=ax5)
graf5.set_xlabel('IG Obstetra', fontsize=12)
graf5.set_ylabel('IG Pediatra', fontsize=12)

# Dispersão x="IG_OBSTETRA", y="PESO_NASCER"
graf6 = sns.scatterplot(data=dados, x="IG_OBSTETRA",
y="PESO_NASCER", ax=ax6)
graf6.set_xlabel('IG Obstetra', fontsize=12)
graf6.set_ylabel('Peso ao Nascer', fontsize=12)

# Ajustar o layout para não haver sobreposição
plt.tight_layout()

# Salvar a figura com os gráficos
plt.savefig('Figura1.png', format='png', dpi=300)
```

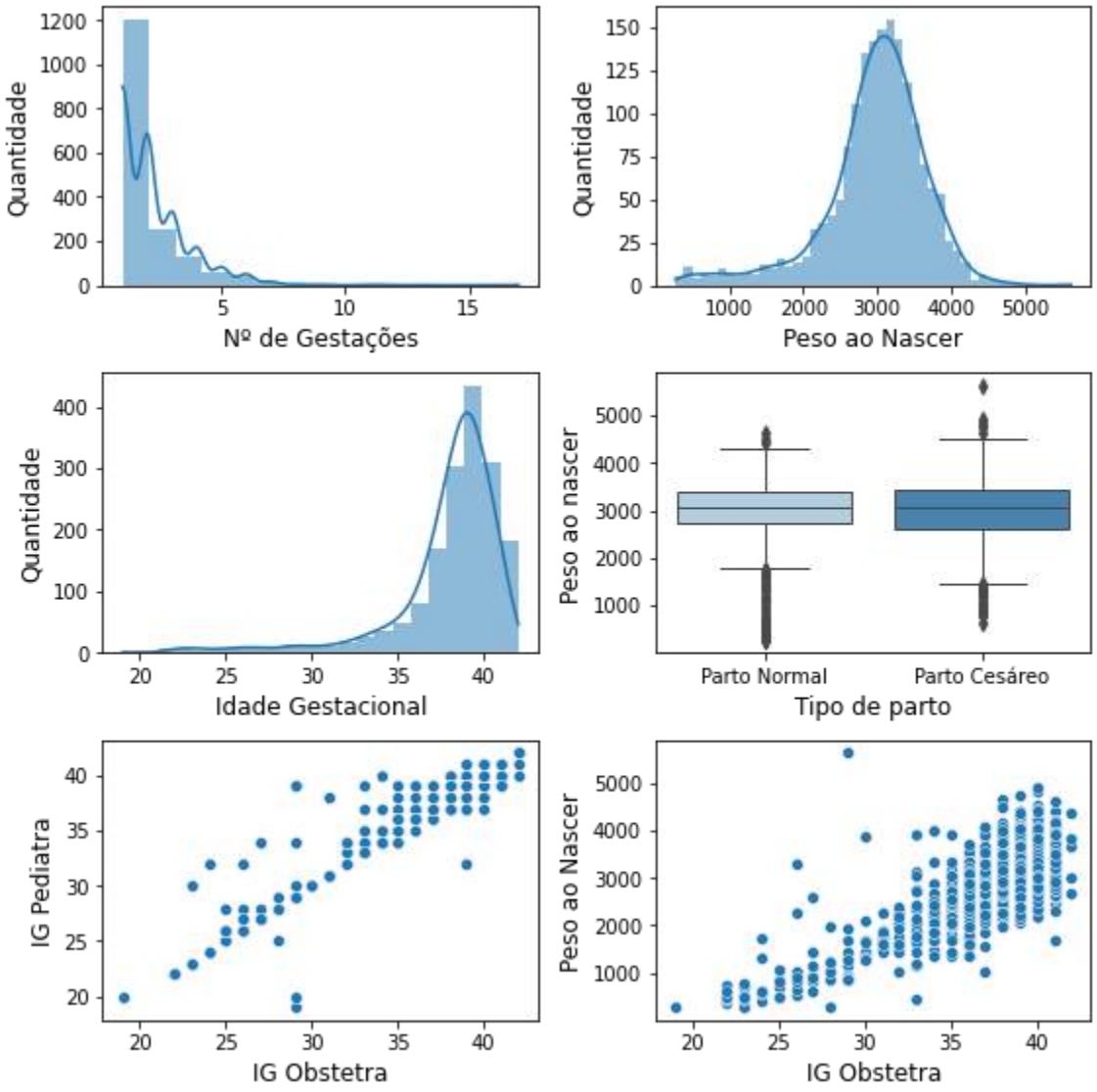


Figura 63: Figura múltipla com seis gráficos

10.6 Exemplos de outros gráficos e combinações

Exemplo: Ao utilizar um gráfico de histograma do Seaborn, e você definir variáveis numéricas para os dois eixos (x e y), você terá uma distribuição representada em formato de mapa de calor.

```
# Histograma como mapa de calor quando x e y são numéricas
graf1 = sns.histplot(dados, x='IG_OBSTETRA', y='PESO_NASCER',
bins=20, cbar=True)

# Formatar o título do gráfico
graf1.set_title('Histograma da idade gestacional e peso')
```

```
# Formatar os rótulos dos eixos
graf1.set_xlabel("Peso ao Nascer")
graf1.set_ylabel("Quantidade")
```

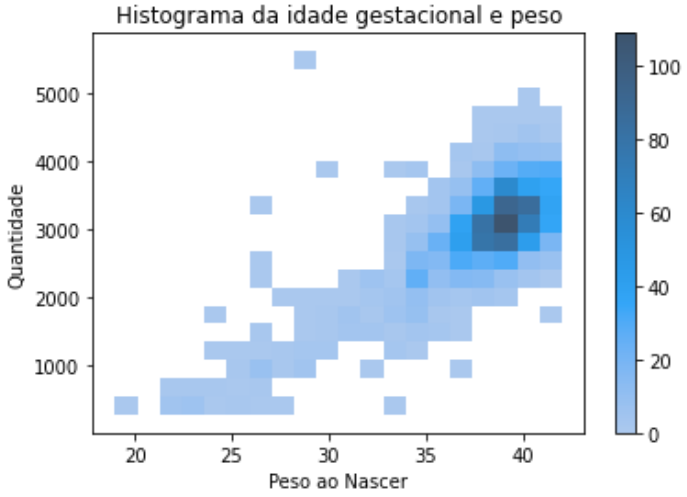


Figura 64: Gráfico de histograma como mapa de calor

Exemplo: Existem diversas formas de analisar um histograma de uma variável numérica (como o peso ao nascer) mas em subgrupos separados (por exemplo pelo sexo do recém nascido). Uma das soluções pode ser utilizar o gráfico **FacetGrid** do **Seaborn**.

```
# Utilizar o FacetGrid para criar histogramas por subgrupos
grafico = sns.FacetGrid(dados, col='SEXO')
grafico.map(sns.histplot, 'PESO_NASCER')
```

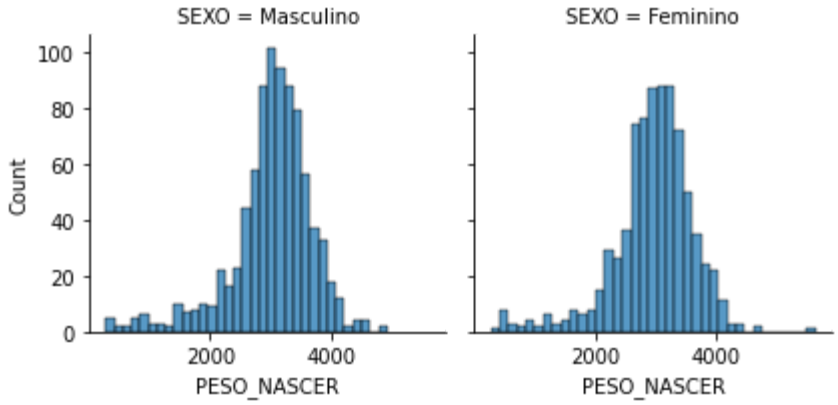


Figura 65: Gráfico de histograma por subgrupos

Exemplo: O **FacetGrid** permite que seja especificada a ordem que se deseja que os subgrupos sejam mostrados.

```
# Criar uma variável contendo a ordem desejada
ordem = ['Prematuro', 'Termo-precoce', 'Termo']
```

```
# Utilizar o FacetGrid para criar histogramas por subgrupos
grafico = sns.FacetGrid(dados, col='IG_TERMO', col_order=ordem)
grafico.map(sns.histplot, 'PESO_NASCER')
```

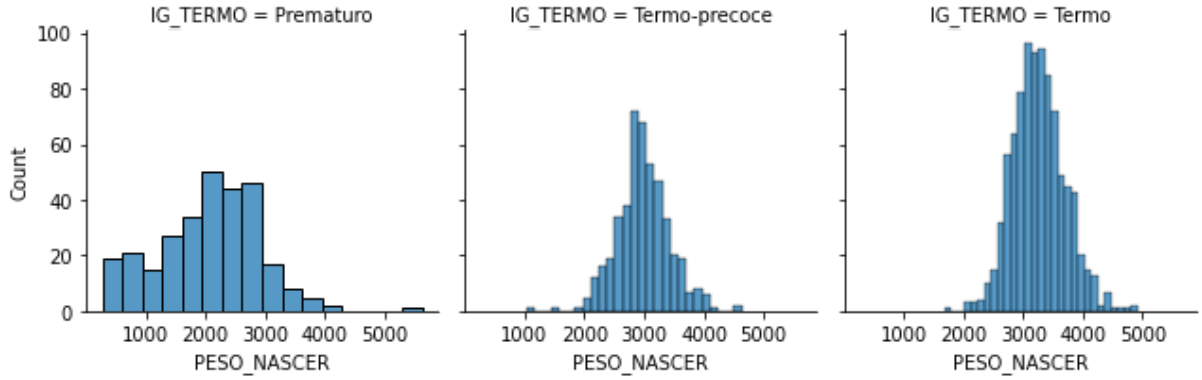


Figura 66: Gráfico de histograma por subgrupos ordenados

Exemplo: A estratégia de ordenação apresentada no exemplo anterior pode ser aplicada a diversos contextos e outros gráficos.

```
# Criar uma variável contendo a ordem desejada
ordem = ['Prematuro', 'Termo-precoce', 'Termo']

# Criar um gráfico de boxplot com ordenação
graf1 = sns.boxplot(data=dados, y='PESO_NASCER', x='IG_TERMO',
order=ordem, palette="Paired", linewidth=1)

# Formatar o título do gráfico
graf1.set_title('Boxplot do peso ao nascer agrupado')

# Formatar os rótulos dos eixos
graf1.set_xlabel("Idade Gestacional")
graf1.set_ylabel("Peso ao Nascer")
```

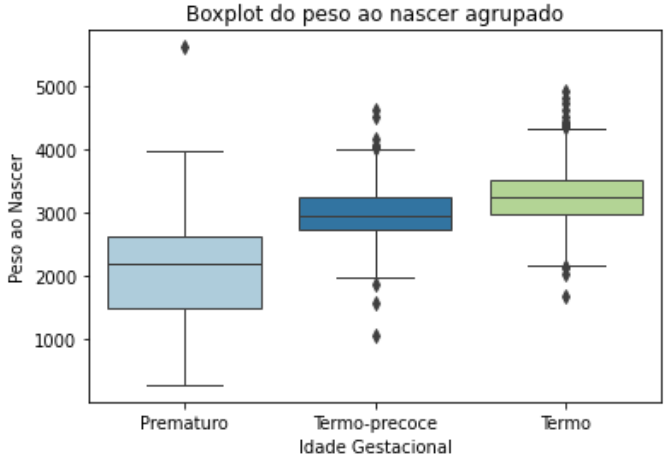


Figura 67: Gráfico de boxplot ordenado

Exemplo: Também é possível formatar a legenda, definir onde se deseja que seja posicionada, entre outros recursos.

```
# Criar uma variável contendo a ordem desejada
ordem = ['Prematuro', 'Termo-precoce', 'Termo']

# Boxplot agrupado e subdividido por categoria
graf1 = sns.boxplot(data=dados, y='PESO_NASCER', x='TIPO_PARTO',
hue='IG_TERMO', hue_order=ordem, palette="Paired", linewidth=1)

# Formatar o título do gráfico
graf1.set_title('Boxplot do peso ao nascer agrupado')

# Formatar os rótulos dos eixos
graf1.set_xlabel("Tipo de Parto")
graf1.set_ylabel("Peso ao Nascer")

# Colocar a legenda ao lado do gráfico
graf1.legend(bbox_to_anchor=(1,1), title='Idade Gestacional')
```

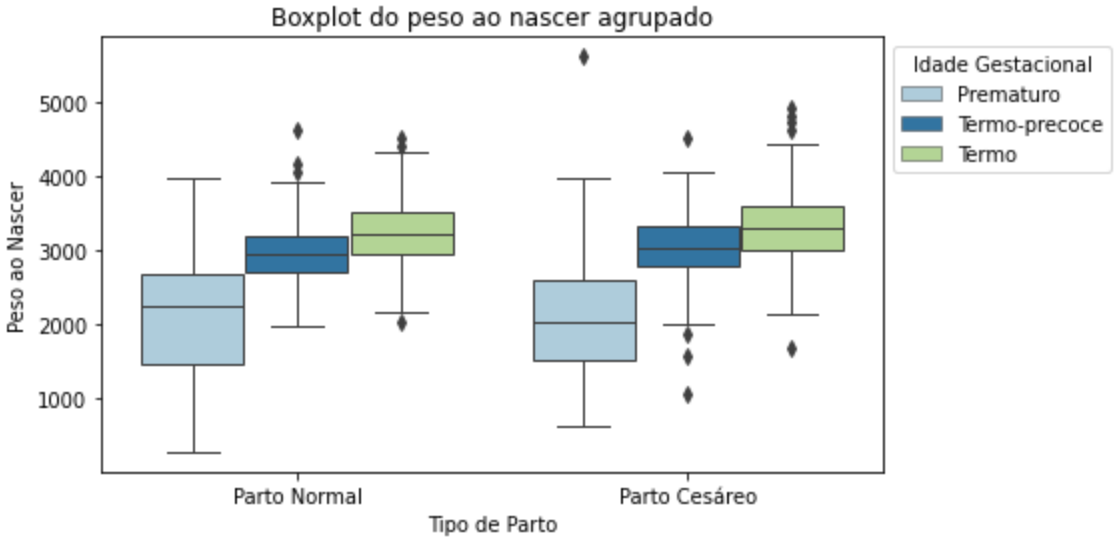


Figura 68: Gráfico de boxplot ordenado

Exemplo: A partir das soluções anteriores, neste exemplo veremos os agrupamentos e formatação de legenda em um gráfico de dispersão.

```
# Criar uma variável contendo a ordem desejada
ordem = ['Prematuro', 'Termo-precoce', 'Termo']

# Gráfico de dispersão
```

```
graf1 = sns.scatterplot(data=dados, x="IG_OBSTETRA",
y="PESO_NASCER", hue="IG_TERMO", hue_order=ordem, palette='Set1')

# Formatar o título do gráfico
graf1.set_title('Gráfico de dispersão')

# Formatar os rótulos dos eixos
graf1.set_xlabel("Idade Gestacional")
graf1.set_ylabel("Peso ao Nascer")

# Colocar a legenda ao lado do gráfico
graf1.legend(bbox_to_anchor=(1,1), title='Classificação')
```

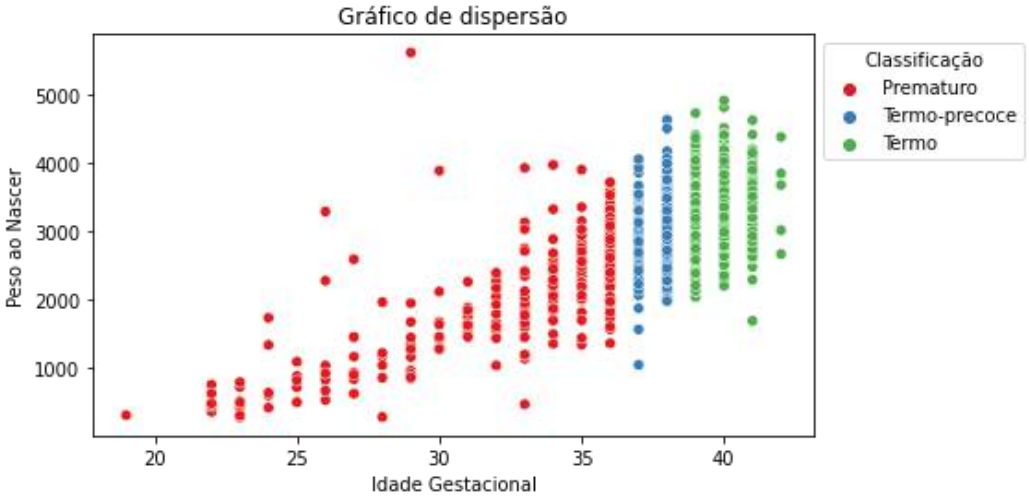


Figura 69: Gráfico de boxplot ordenado

Exemplo: Outro tipo de gráfico é o **Regplot** da **Seaborn**. Ele também é um gráfico de dispersão como o Scatterplot, porém apresenta a linha de regressão linear e as faixas com os limites de confiança (por padrão 95% de confiança, mas pode ser ajustada).

```
# Gráfico de dispersão com linha de tendência, com IC:95%
graf1 = sns.regplot(data=dados, x='IG_OBSTETRA', y='APGAR5', ci=95 )

# Formatar o título do gráfico
graf1.set_title('Gráfico de dispersão')

# Formatar os rótulos dos eixos
graf1.set_xlabel("Idade Gestacional")
graf1.set_ylabel("Apgar de 5° minuto")
```

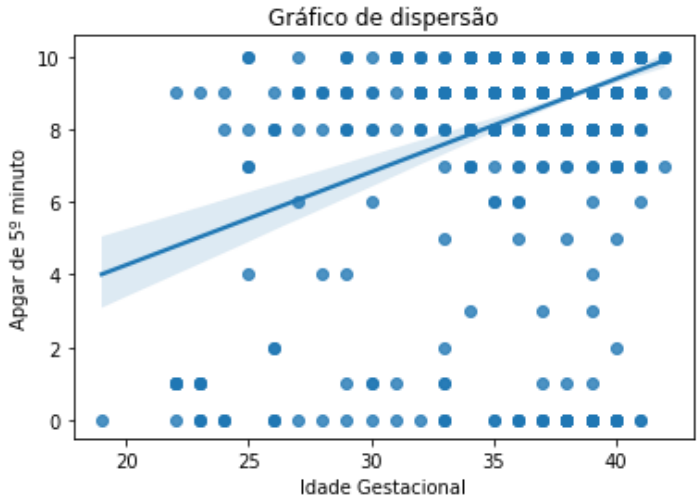


Figura 70: Gráfico de dispersão com linha de tendência

Exemplo: Outro tipo de gráfico é o **Lmplot** da **Seaborn**. Ele também é um gráfico de dispersão como o Scatterplot, porém calcula uma linha de regressão linear para cada subgrupo analisado e as respectivas faixas com os limites de confiança.

```
# Gráfico dispersão com linha de regressão para cada subgrupo
graf1 = sns.lmplot(data=dados, x="IG_OBSTETRA", y="PESO_NASCER",
hue="VIVO")

# Formatar os rótulos dos eixos x e y
graf1.set_axis_labels("IG Obstetra", "Peso ao Nascer")
```

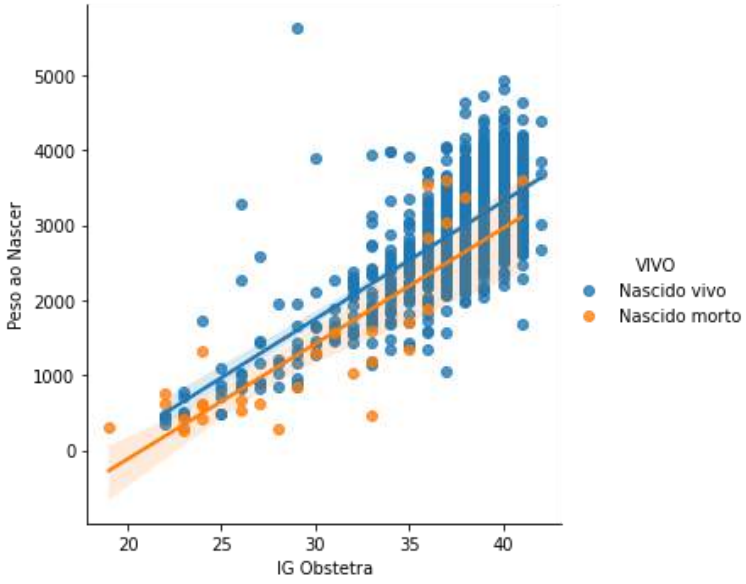


Figura 71: Gráfico de dispersão com linhas de regressão por subgrupo

Exemplo: Outro tipo de gráfico que também está associado ao gráfico de dispersão é o **Jointplot** da **Seaborn**. Este gráfico apresenta a correlação dos valores no gráfico central e os histogramas de cada variável x e y.

```
# Jointplot: combina o gráfico de dispersão com histogramas
graf1 = sns.jointplot(x=dados.IG_OBSTETRA, y=dados.PESO_NASCER,
kind="scatter")

# Formatar os rótulos dos eixos x e y
graf1.set_axis_labels("IG Obstetra", "Peso ao Nascer")
```

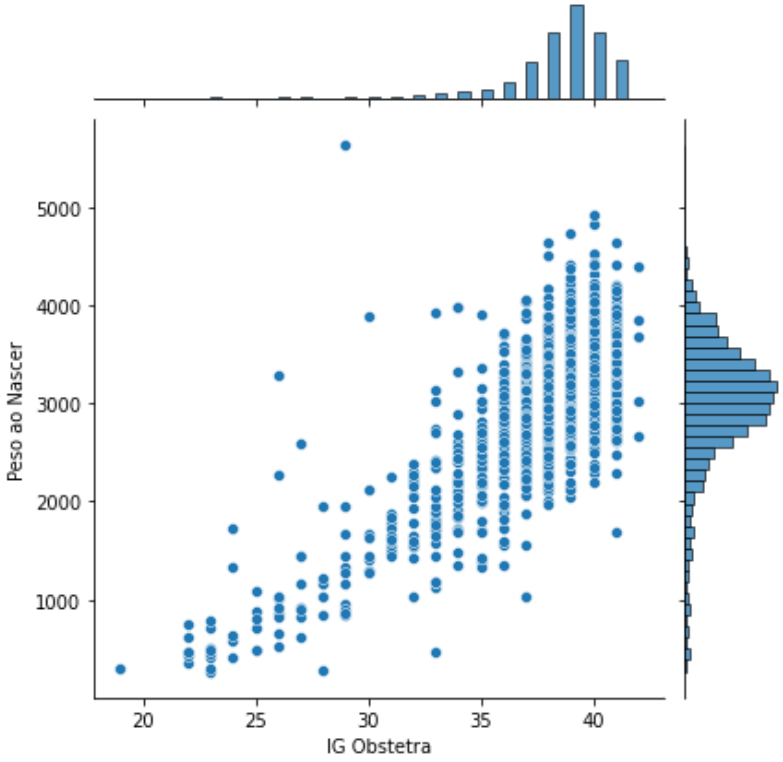


Figura 72: Gráfico de dispersão com histogramas

Exemplo: O gráfico de **ViolinPlot** da **Seaborn** é similar ao gráfico de boxplot, entretanto ele apresenta uma estimativa de densidade do kernel da distribuição (KDE) (uma suavização) ao invés de mostrar os pontos reais da amostra como o boxplot. Esta pode ser uma maneira eficaz e atraente de mostrar várias distribuições de dados de uma só vez, mas lembre-se de que o KDE é influenciado pelo tamanho da amostra, em amostras relativamente pequenas podem parecer enganosamente suaves.

```
# Ordenação das categorias
ordem = ['Prematuro', 'Termo-precoce', 'Termo']
```

```
# Gráfico do tipo ViolinPlot
graf1 = sns.violinplot(data=dados, x="IG_TERMO", y="PESO_NASCER",
hue="SEXO", split=True, inner="quart", linewidth=1, order=ordem,
palette="Paired")

# Formatar o título do gráfico
graf1.set_title('Gráfico de ViolinPlot')

# Formatar os rótulos dos eixos
graf1.set_xlabel("Idade Gestacional")
graf1.set_ylabel("Peso ao Nascer")

# Colocar a legenda ao lado do gráfico
graf1.legend(bbox_to_anchor=(1.4,1), title='Sexo do recém nascido')
```

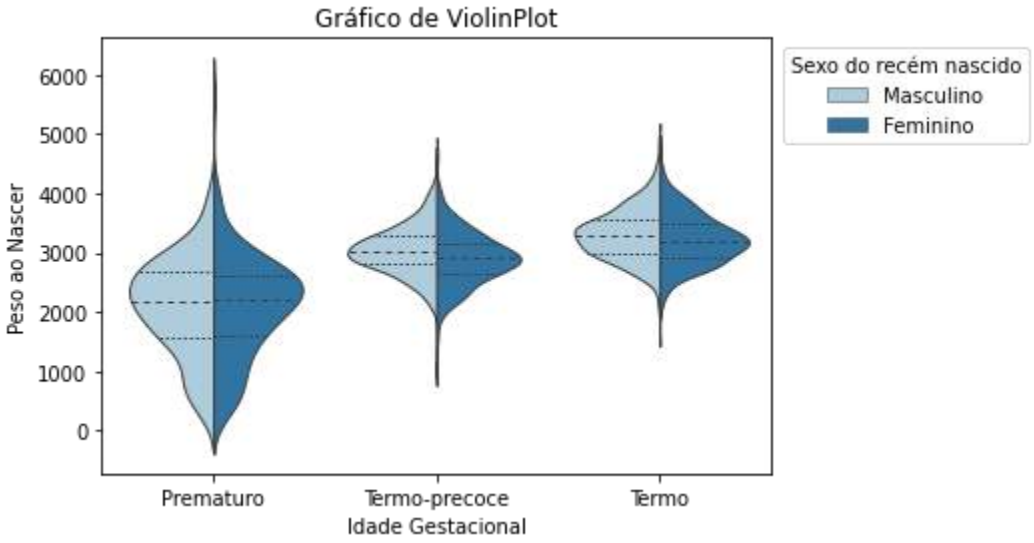


Figura 73: Gráfico Violinplot - Boxplot com KDE

Exemplo: O gráfico de **PairPlot** da **Seaborn** apresenta vários gráficos com as correlações combinadas entre todas as variáveis numéricas do dataset informado. Ele é útil quando se deseja de forma rápida e visual analisar as combinações possíveis na base de dados.

```
# Para que o PairPlot fique adequado ao entendimento, deve-se
selecionar poucas colunas
colunas = ['DURACAO_INT', 'PESO_NASCER', 'IG_OBSTETRA', 'APGAR5',
'TIPO_PARTO']
dados2 = dados[colunas]

# PairPlot cruza todas as variáveis numéricas
# Neste exemplo, faz um histograma (KDE) na diagonal principal
# Os gráficos da diagonal superior podem ser diferentes dos
mostrados na diagonal inferior (com eixo invertido)
```

```
# Neste exemplo, além do cruzamento das variáveis numéricas, estamos analisando em subgrupos por tipo de parto
g = sns.pairplot(dados2, hue="TIPO_PARTO", palette="Set2", diag_kind="kde", height=2.5)
```

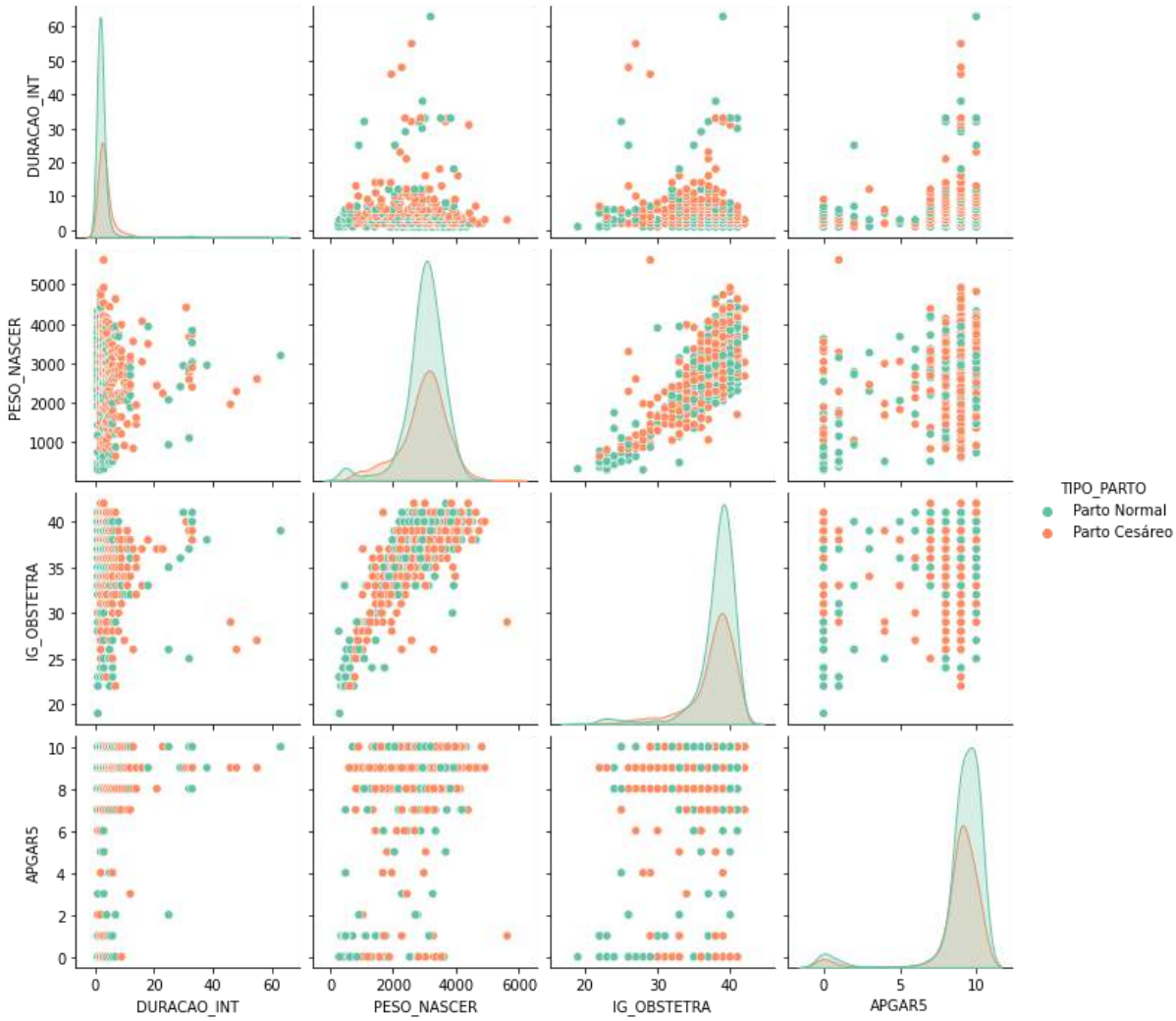


Figura 74: Gráfico PairPlot

Saiba que existem centenas de outros formatos e estilizações de gráficos. Procuramos mostrar aqui apenas alguns exemplos de formatação básica para que você consiga explorar a documentação oficial de cada biblioteca de forma autônoma. Para saber mais sobre os gráficos apresentados consulte os sites oficiais das bibliotecas Matplotlib (68), Seaborn (69) e outras que desejar utilizar.

Referências bibliográficas

1. Brasil. Estratégia de Saúde Digital para o Brasil 2020-2028 [Internet]. 1º ed. Brasília: Ministério da Saúde; 2020. Disponível em: https://bvsms.saude.gov.br/bvs/publicacoes/estrategia_saude_digital_Brasil.pdf
2. WHO. Global strategy on digital health 2020-2025 [Internet]. World Health Organization; 2021 [citado 8 de abril de 2023]. Disponível em: https://apps.who.int/iris/handle/10665/344249
3. Petrovskiy MI. Outlier Detection Algorithms in Data Mining Systems. Program Comput Softw. 1º de julho de 2003;29(4):228–37.
4. Shortliffe EH, Cimino JJ, organizadores. Biomedical Informatics: Computer Applications in Health Care and Biomedicine [Internet]. New York, NY: Springer; 2006 [citado 8 de abril de 2023]. (Health Informatics). Disponível em: http://link.springer.com/10.1007/0-387-36278-9
5. Shortliffe EH, Barnett GO. Biomedical Data: Their Acquisition, Storage, and Use. Em: Shortliffe EH, Cimino JJ, organizadores. Biomedical Informatics: Computer Applications in Health Care and Biomedicine [Internet]. New York, NY: Springer; 2006 [citado 8 de abril de 2023]. p. 46–79. (Health Informatics). Disponível em: https://doi.org/10.1007/0-387-36278-9_2
6. Cruz-Correia RJ, Rodrigues PP, Freitas A, Almeida FC, Chen R, Costa-Pereira A. Data Quality and Integration Issues in Electronic Health Records. Em: Information Discovery on Electronic Health Records. Chapman and Hall/CRC; 2009.
7. Peek N, Holmes J, Martin-Sanchez F, Sun J. Big data analytics in biomedicine and health: trends and challenges. Stud Health Technol Inform. 1º de janeiro de 2013;192:1237.
8. Netto A, Maciel F. Python Para Data Science: E Machine Learning Descomplicado. 1ª edição. Alta Books; 2021.
9. Python. A Biblioteca Padrão do Python [Internet]. Python documentation. 2023 [citado 15 de março de 2023]. Disponível em: https://docs.python.org/3/library/index.html
10. Google. Google Colaboratory [Internet]. 2023 [citado 17 de março de 2023]. Disponível em: https://colab.research.google.com/
11. Pandas. Biblioteca Pandas- Python Data Analysis Library [Internet]. 2023 [citado 17 de março de 2023]. Disponível em: https://pandas.pydata.org/
12. Matplotlib. Biblioteca Matplotlib: Visualization with Python [Internet]. 2023 [citado 17 de março de 2023]. Disponível em: https://matplotlib.org/
13. NumPy. Biblioteca NumPy [Internet]. 2023 [citado 17 de março de 2023]. Disponível em: https://numpy.org/
14. Seaborn. Biblioteca Seaborn: statistical data visualization [Internet]. 2023 [citado 17 de março de 2023]. Disponível em: https://seaborn.pydata.org/
15. VanderPlas J. Python Data Science Handbook: Essential Tools for Working with Data. O'Reilly Media, Inc.; 2016. 548 p.
16. Downey AB. Think Stats. O'Reilly Media; 2014. 226 p.
17. Pandas. Função: pandas.read_excel [Internet]. 2023 [citado 18 de março de 2023]. Disponível em: https://pandas.pydata.org/docs/reference/api/pandas.read_excel.html
18. Pandas. Função: pandas.Series.value_counts [Internet]. 2023 [citado 18 de março de 2023]. Disponível em: https://pandas.pydata.org/docs/reference/api/pandas.Series.value_counts.html?highlight=value_counts#pandas.Series.value_counts
19. Pandas. Função: pandas.DataFrame.describe [Internet]. 2023 [citado 18 de março de 2023].

Disponível em:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.describe.html?highlight=describe#pandas.DataFrame.describe

20. Arango HG. Bioestatística-Teórica e Computacional. 3ª edição. Guanabara Koogan; 2011.
21. Chandola V, Banerjee A, Kumar V. Anomaly detection: A survey. ACM Comput Surv. 30 de julho de 2009;41(3):15:1-15:58.
22. Pandas. Função: pandas.Series.plot.bar [Internet]. 2023 [citado 18 de março de 2023]. Disponível em:
https://pandas.pydata.org/docs/reference/api/pandas.Series.plot.bar.html?highlight=plot%20bar#pandas.Series.plot.bar
23. Seaborn. Função: seaborn.histplot [Internet]. 2023 [citado 18 de março de 2023]. Disponível em:
https://seaborn.pydata.org/generated/seaborn.histplot.html#seaborn.histplot
24. Seaborn. Função: seaborn.boxplot [Internet]. 2023 [citado 18 de março de 2023]. Disponível em:
https://seaborn.pydata.org/generated/seaborn.boxplot.html#seaborn.boxplot
25. Scipy. Função: scipy.stats.zscore [Internet]. 2023 [citado 18 de março de 2023]. Disponível em:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.zscore.html
26. Glen S. Slovin's Formula: What is it and When do I use it? [Internet]. Statistics How To. 2023 [citado 22 de março de 2023]. Disponível em:
https://www.statisticshowto.com/probability-and-statistics/how-to-use-slovins-formula/
27. Ryan. Sample Size Determination and Power. 1ª edição. Hoboken, N.J: Wiley; 2013.
28. Miot HA. Avaliação da normalidade dos dados em estudos clínicos e experimentais. J Vasc Bras. 2017;16(2):88–91.
29. Pandas. Função: pandas.DataFrame.sample [Internet]. 2023 [citado 18 de março de 2023]. Disponível em: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.sample.html
30. Pandas. Função: pandas.DataFrame.iloc [Internet]. 2023 [citado 18 de março de 2023]. Disponível em: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.iloc.html
31. NumPy. Função: numpy.arange [Internet]. 2023 [citado 18 de março de 2023]. Disponível em:
https://numpy.org/doc/stable/reference/generated/numpy.arange.html
32. Scikit-learn. Função: sklearn.model_selection.StratifiedShuffleSplit [Internet]. scikit-learn. 2023 [citado 18 de março de 2023]. Disponível em:
https://scikit-learn/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html
33. Scipy. Função: scipy.stats.normaltest [Internet]. 2023 [citado 18 de março de 2023]. Disponível em:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.normaltest.html
34. Scipy. Função: scipy.stats.shapiro [Internet]. 2023 [citado 18 de março de 2023]. Disponível em:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.shapiro.html
35. Scipy. Função: scipy.stats.kstest [Internet]. 2023 [citado 18 de março de 2023]. Disponível em:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kstest.html
36. Viera AJ, Garrett JM. Understanding interobserver agreement: the kappa statistic. Fam Med. maio de 2005;37(5):360–3.
37. SciPy. Função: scipy.stats.chi2_contingency [Internet]. 2023 [citado 20 de março de 2023]. Disponível em:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chi2_contingency.html
38. SciPy. Função: scipy.stats.fisher_exact [Internet]. 2023 [citado 20 de março de 2023]. Disponível em: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.fisher_exact.html
39. SciPy. Função: scipy.stats.contingency.relative_risk [Internet]. 2023 [citado 20 de março de 2023]. Disponível em:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.contingency.relative_risk.html

40. Statsmodels. Função: statsmodels.stats.contingency_tables.mcnemar [Internet]. 2023 [citado 20 de março de 2023]. Disponível em: https://www.statsmodels.org/devel/generated/statsmodels.stats.contingency_tables.mcnemar.html

41. Scikit-learn. Função: sklearn.metrics.cohen_kappa_score [Internet]. 2023 [citado 20 de março de 2023]. Disponível em: https://scikit-learn/stable/modules/generated/sklearn.metrics.cohen_kappa_score.html

42. Scikit-learn. Função: sklearn.metrics.confusion_matrix [Internet]. scikit-learn. 2023 [citado 20 de março de 2023]. Disponível em: https://scikit-learn/stable/modules/generated/sklearn.metrics.confusion_matrix.html

43. Seaborn. Função: seaborn.heatmap [Internet]. 2023 [citado 20 de março de 2023]. Disponível em: https://seaborn.pydata.org/generated/seaborn.heatmap.html

44. SciPy. Função: scipy.stats.ttest_1samp [Internet]. 2023 [citado 20 de março de 2023]. Disponível em: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_1samp.html#scipy.stats.ttest_1samp

45. SciPy. Função: scipy.stats.levene [Internet]. 2023 [citado 20 de março de 2023]. Disponível em: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.levene.html

46. SciPy. Função: scipy.stats.ttest_ind [Internet]. 2023 [citado 20 de março de 2023]. Disponível em: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html#scipy.stats.ttest_ind

47. SciPy. Função: scipy.stats.ttest_rel [Internet]. 2023 [citado 20 de março de 2023]. Disponível em: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_rel.html

48. SciPy. Função: scipy.stats.f_oneway [Internet]. 2023 [citado 20 de março de 2023]. Disponível em: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.f_oneway.html

49. Statsmodels. Função: statsmodels.sandbox.stats.multicomp.MultiComparison [Internet]. 2023 [citado 20 de março de 2023]. Disponível em: https://www.statsmodels.org/dev/generated/statsmodels.sandbox.stats.multicomp.MultiComparison.html

50. Statsmodels. Função: statsmodels.stats.anova.AnovaRM [Internet]. 2023 [citado 20 de março de 2023]. Disponível em: https://www.statsmodels.org/dev/generated/statsmodels.stats.anova.AnovaRM.html

51. SciPy. Função: scipy.stats.mstats.mannwhitneyu [Internet]. 2023 [citado 24 de março de 2023]. Disponível em: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.mstats.mannwhitneyu.html

52. SciPy. Função: scipy.stats.mstats.kruskalwallis [Internet]. 2023 [citado 24 de março de 2023]. Disponível em: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.mstats.kruskalwallis.html

53. SciPy. Função: scipy.stats.wilcoxon [Internet]. 2023 [citado 24 de março de 2023]. Disponível em: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wilcoxon.html#scipy.stats.wilcoxon

54. SciPy. Função: scipy.stats.mstats.friedmanchisquare [Internet]. 2023 [citado 24 de março de 2023]. Disponível em: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.mstats.friedmanchisquare.html

55. Zach. How to Perform Dunn's Test in Python [Internet]. Statology. 2023 [citado 24 de março de 2023]. Disponível em: https://www.statology.org/dunns-test-python/

56. Mukaka M. A guide to appropriate use of Correlation coefficient in medical research. Malawi Med J J Med Assoc Malawi. setembro de 2012;24(3):69–71.

57. Nicholls R. Statistics in Python: Intraclass Correlation Coefficient [Internet]. 2023 [citado 25 de março de 2023]. Disponível em: https://rowannicholls.github.io/python/statistics/agreement/intraclass_correlation.html

58. Cicchetti DV. Guidelines, criteria, and rules of thumb for evaluating normed and standardized assessment instruments in psychology. Psychol Assess. 1994;6:284–90.

59. Koo TK, Li MY. A Guideline of Selecting and Reporting Intraclass Correlation Coefficients for Reliability Research. J Chiropr Med. junho de 2016;15(2):155–63.

60. SciPy. Função: scipy.stats.pearsonr [Internet]. 2023 [citado 25 de março de 2023]. Disponível em: https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.pearsonr.html

61. SciPy. Função: scipy.stats.spearmanr [Internet]. 2023 [citado 25 de março de 2023]. Disponível em: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html

62. SciPy. Função: scipy.stats.kendalltau [Internet]. 2023 [citado 25 de março de 2023]. Disponível em: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kendalltau.html

63. Seaborn. Função: seaborn.regplot [Internet]. 2023 [citado 25 de março de 2023]. Disponível em: https://seaborn.pydata.org/generated/seaborn.regplot.html

64. Zach. How to Calculate Intraclass Correlation Coefficient in Python [Internet]. Statology. 2023 [citado 25 de março de 2023]. Disponível em: https://www.statology.org/intraclass-correlation-coefficient-python/

65. Matplotlib. List of named colors [Internet]. 2023 [citado 26 de março de 2023]. Disponível em: https://matplotlib.org/stable/gallery/color/named_colors.html

66. Seaborn. Choosing color palettes [Internet]. 2023 [citado 26 de março de 2023]. Disponível em: https://seaborn.pydata.org/tutorial/color_palettes.html

67. Matplotlib. Subplots, axes and figures — Matplotlib 3.7.1 documentation [Internet]. 2023 [citado 27 de março de 2023]. Disponível em: https://matplotlib.org/stable/gallery/subplots_axes_and_figures/index.html

68. Matplotlib. Biblioteca Matplotlib: Galeria de exemplos [Internet]. 2023 [citado 27 de março de 2023]. Disponível em: https://matplotlib.org/stable/gallery/index

69. Seaborn. Biblioteca Seaborn: Galeria de exemplos [Internet]. 2023 [citado 27 de março de 2023]. Disponível em: https://seaborn.pydata.org/examples/index.html

Sobre os autores

Prof. Juliano de Souza Gaspar
Professor convidado da Faculdade de Medicina da UFMG e da pós-graduação em Saúde Digital da UFG. É doutor em Saúde com foco em Saúde Digital pela Faculdade de Medicina da UFMG. Mestre em Informática Médica pela Universidade do Porto (Portugal) e graduado em Ciências da Computação pela Universidade do Vale do Itajaí.
Currículum Lattes: http://lattes.cnpq.br/3926707936198077

Me. Isaias José Ramos de Oliveira
Mestre em Saúde da Mulher pela Faculdade de Medicina da UFMG e graduado em Ciências da Computação pela UFMG. É especialista em Tecnologia da Informação e atua no Centro de Informática em Saúde da UFMG.
Currículum Lattes: http://lattes.cnpq.br/9610525418655146

Profa. Ana Paula Couto da Silva
Professora Associada do Departamento de Ciência da Computação da UFMG. Doutora em Engenharia de Sistemas e Computação pela COPPE/UFRJ. É bolsista de Produtividade em Pesquisa do CNPq (Nível 2) e publicou mais de 35 artigos em periódicos e 90 artigos em anais de congressos.
Currículum Lattes: http://lattes.cnpq.br/2408991231058279

Profa. Cristiane dos Santos Dias
Professora Adjunta do Departamento de Pediatria da UFMG. Membro da Diretoria de Educação e Difusão do Conhecimento do CI-IA Saúde. Preceptora do Programa de Residência Médica em Pediatria da FHEMIG.
Currículum Lattes: http://lattes.cnpq.br/9267111928280200

Profa. Zilma Silveira Nogueira Reis
Professora Associada da Faculdade de Medicina da UFMG, onde coordena o Centro de Informática em Saúde. Bolsista de Produtividade, Desenvolvimento Tecnológico e Extensão Inovadora do CNPq (Nível 2). Coordenadora de Educação e Difusão do Conhecimento do CI-IA Saúde.
Currículum Lattes: http://lattes.cnpq.br/5695664808243549

Realização

CI-IA
Centro de Inovação em Inteligência Artificial para a Saúde

Apoio

UFMG

ISBN: 978-65-86593-18-1