UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

ISABELLA STERZA DE OLIVEIRA BUTZEN
ARACELI CIOTTI DE MARINS
DANIELA TRENTIN NAVA

MANUAL DIGITAL PARA CONSTRUÇÃO DE GRÁFICOS NO RSTUDIO: UM GUIA PRÁTICO EM LINGUAGEM R PARA DOCENTES

TOLEDO

2026

RESUMO

Este trabalho apresenta o desenvolvimento de um recurso educacional em formato de manual digital, voltado a docentes, com o objetivo de apoiar a construção de gráficos no RStudio por meio da linguagem R, abrangendo desde a instalação e a configuração do ambiente até a aplicação de comandos fundamentais para importação, tratamento e visualização de dados. O manual organiza-se em etapas progressivas, contemplando orientações operacionais, exemplos comentados e procedimentos replicáveis para a geração de representações gráficas aplicáveis a contextos de análise e ensino. A elaboração do recurso foi orientada por uma abordagem prática, ancorada na execução de *scripts* e na produção de saídas gráficas diretamente vinculadas aos códigos apresentados, buscando favorecer autonomia e segurança no uso da ferramenta. Como estratégia de transparência e reprodutibilidade, os *scripts* empregados ao longo do trabalho e na construção dos gráficos são disponibilizados em seção pós-textual e em repositório no *GitHub*. Espera-se que o material contribua para a qualificação do trabalho docente com dados e para o fortalecimento de práticas de visualização aplicadas à educação.

Palavras-chave: Linguagem R; Visualização de dados; Formação docente.

ABSTRACT

This study presents the development of an educational resource in the form of a digital manual aimed at teachers, with the purpose of supporting the creation of graphs in RStudio using the R language. It covers procedures ranging from installing and configuring the environment to applying fundamental commands for data import, processing, and visualization. The manual is organized into progressive stages, including operational guidance, annotated examples, and replicable procedures for generating graphical representations applicable to analysis and teaching contexts. The resource was developed through a practical approach, grounded in the execution of scripts and in graphical outputs directly linked to the codes presented, seeking to foster users' autonomy and confidence in using the tool. As a strategy to ensure transparency and reproducibility, the scripts used throughout the study and in the construction of the graphs are made available in a post-textual section and in a GitHub repository. It is expected that the material will contribute to enhancing teachers' work with data and to strengthening visualization practices applied to education.

Keywords: R language; Data visualization; Teacher education.

SUMÁRIO

1 APRESENTAÇÃO

Prezado professor(a), este manual foi desenvolvido com o objetivo de apoiar professores no uso da linguagem `R` (R Core Team, 2024) e do ambiente `RStudio` (Posit, 2026) como ferramentas pedagógicas para o ensino de estatística e visualização de dados. Mais do que apresentar comandos prontos, a proposta é promover a autonomia docente na criação de gráficos significativos, que contribuam para o desenvolvimento do pensamento crítico e do letramento estatístico dos alunos.

Por que usar R e RStudio para ensinar estatística?

O `R` é uma linguagem de programação voltada para análise de dados e estatística, amplamente utilizada em ambientes acadêmicos e profissionais. No contexto educacional, seu uso oferece vantagens significativas: permite a criação de gráficos altamente personalizáveis, facilita o trabalho com dados reais coletados em sala de aula e contribui para o desenvolvimento de competências digitais. Ao utilizar o R, os alunos têm contato com uma linguagem moderna, desenvolvendo habilidades essenciais para o século XXI, como pensamento computacional, interpretação de dados e autonomia na resolução de problemas.

Ao utilizar o `R`, o professor pode modificar praticamente todos os elementos de um gráfico (cores, títulos, eixos, legendas, escalas e estilos), o que possibilita a construção de visualizações adaptadas ao conteúdo e aos objetivos pedagógicos. Essa flexibilidade torna os gráficos mais expressivos e eficazes na comunicação de padrões, tendências e relações entre dados.

Complementando o uso do R, o `RStudio` é um ambiente de desenvolvimento integrado (IDE) que organiza e facilita o trabalho do professor. Com uma interface amigável, o RStudio permite escrever, executar e salvar scripts, visualizar gráficos, explorar dados e acompanhar o histórico de comandos. Essa estrutura torna o processo de criação mais acessível, mesmo para quem está iniciando no uso da linguagem.

Ao incorporar o R e o RStudio à prática docente, o professor amplia seu repertório didático, promove o letramento estatístico e transforma a sala de aula em um espaço de investigação e construção coletiva de conhecimento.

2 PRIMEIROS PASSOS NO RSTUDIO

Este capítulo orienta a configuração do ambiente necessário para acompanhar as etapas seguintes do material. A proposta é garantir que você consiga instalar o R (R Core Team, 2024) e o RStudio (Posit, 2026), organizar o trabalho em um projeto, criar um *script*, importar uma base de dados e deixar os pacotes essenciais prontos para uso, de modo que, nos próximos capítulos, o foco esteja na construção e interpretação de gráficos. Para facilitar a compreensão, este capítulo conta com um vídeo de apoio, disponibilizado no *YouTube*, no link `https://youtu.be/P0ckTM_zMBw` (Butzen, 2026).

Ao final deste capítulo, você será capaz de:

- Instalar o R e o RStudio no Windows.
- Reconhecer as principais áreas do RStudio (*Console*, *Source*, *Environment* e *Plots*).
- Criar um projeto e um *script* para organizar seu trabalho.
- Importar uma base de dados (`.csv` e `.xlsx`).
- Instalar e carregar pacotes, preparando o ambiente para criar gráficos.

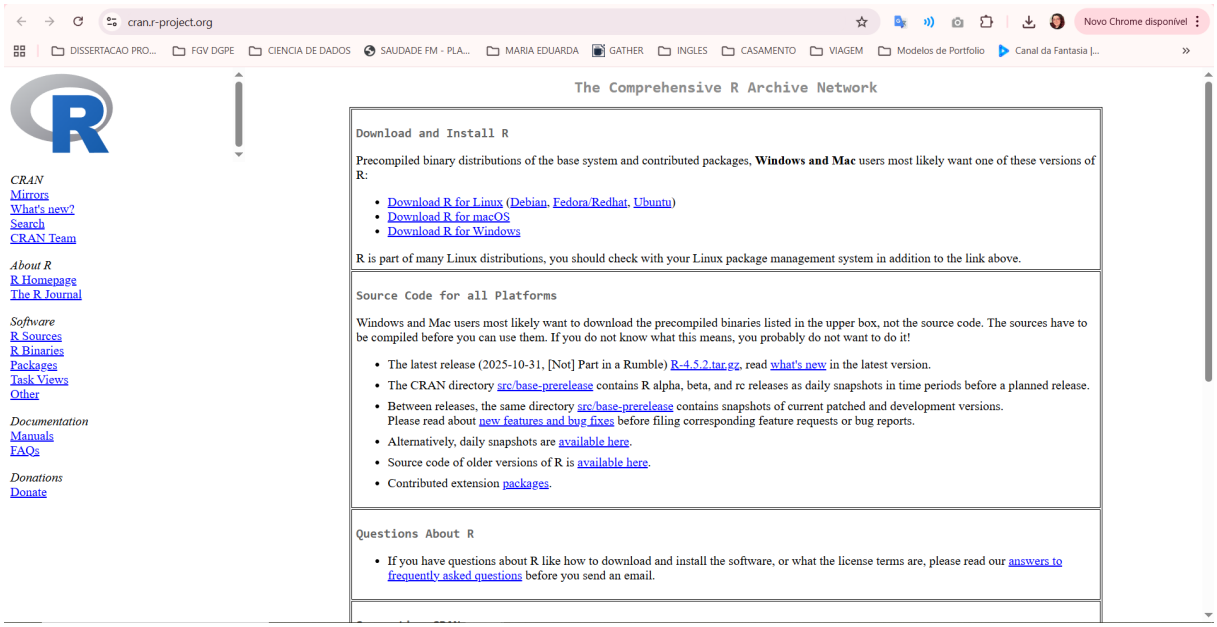
2.1 Instalação do R e RStudio

O ecossistema de trabalho é composto por dois softwares distintos que atuam em conjunto: o R (a linguagem de programação) e o RStudio (a interface que facilita o uso da linguagem). Neste material, a instalação é apresentada para Windows, por ser o sistema operacional mais utilizado. As imagens desta seção foram obtidas a partir dos sites oficiais em janeiro de 2026.

2.1.1 Instalando o R

A instalação do R é realizada pelo site oficial do CRAN (*Comprehensive R Archive Network*) (CRAN, 2026), que funciona como repositório central e oficial da linguagem. O acesso deve ser feito pelo endereço `https://cran.r-project.org/`, no qual estão disponíveis as opções de *download* para diferentes sistemas operacionais, conforme ilustrado na Figura 1.

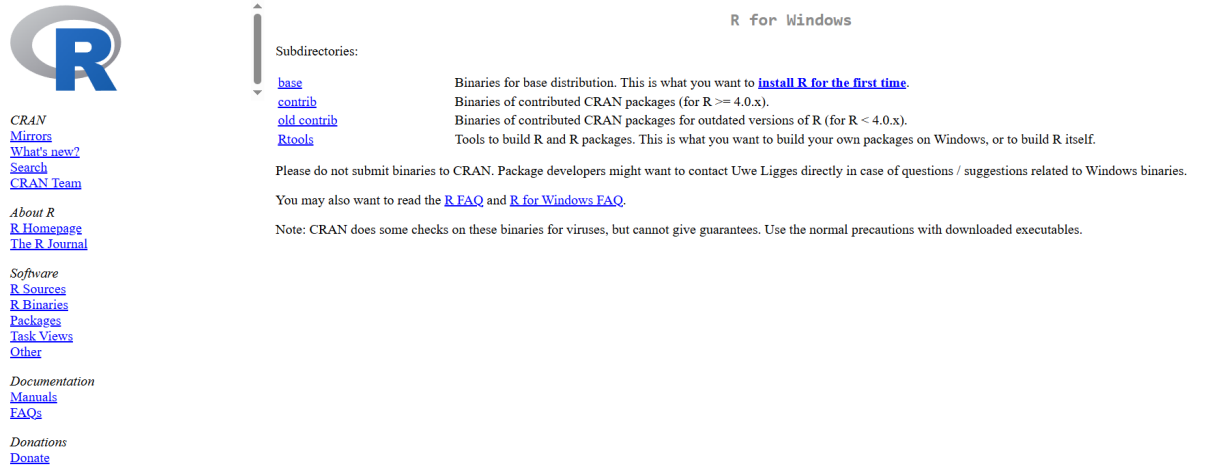
Figura 1 – Página inicial do CRAN (*Comprehensive R Archive Network*)



Fonte: Autoras (2026)

Para realizar o *download*, selecione a opção correspondente ao seu sistema operacional. Neste material, estamos realizando a instalação no Windows; portanto, selecione *Download R for Windows*. Uma nova página será aberta, conforme a Figura 2. Nessa etapa, clique em *base* para acessar o instalador principal do R para Windows.

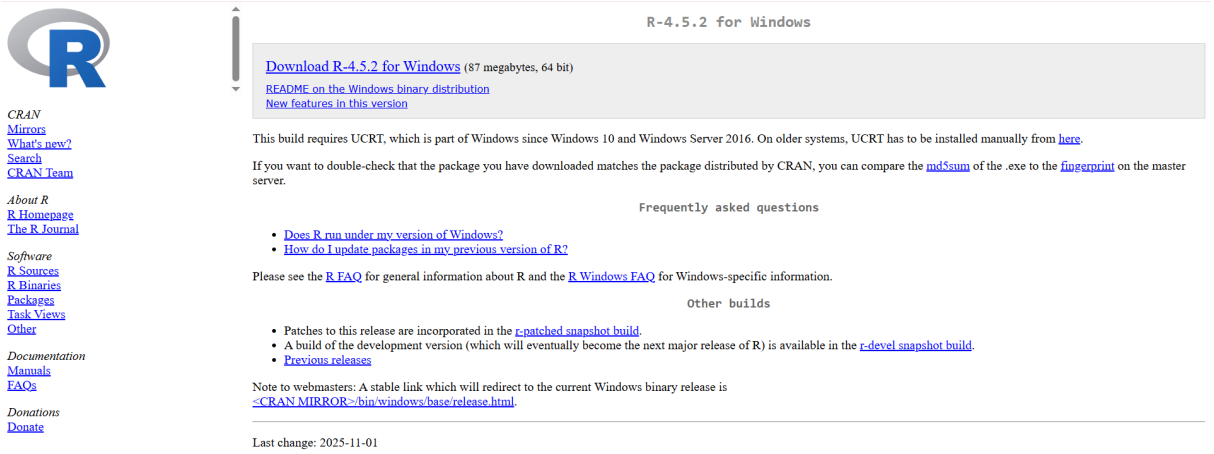
Figura 2 – Página do CRAN com a opção *Download R for Windows* e o acesso à seção *base*



Fonte: Autoras (2026)

Na página *base*, selecione o link do instalador mais recente (por exemplo, *Download R-x.x.x for Windows*). No momento em que este material foi elaborado, a versão vigente era 4.5.2, conforme a Figura 3. Ao clicar, o arquivo será baixado e, em geral, ficará disponível na pasta *Downloads*.

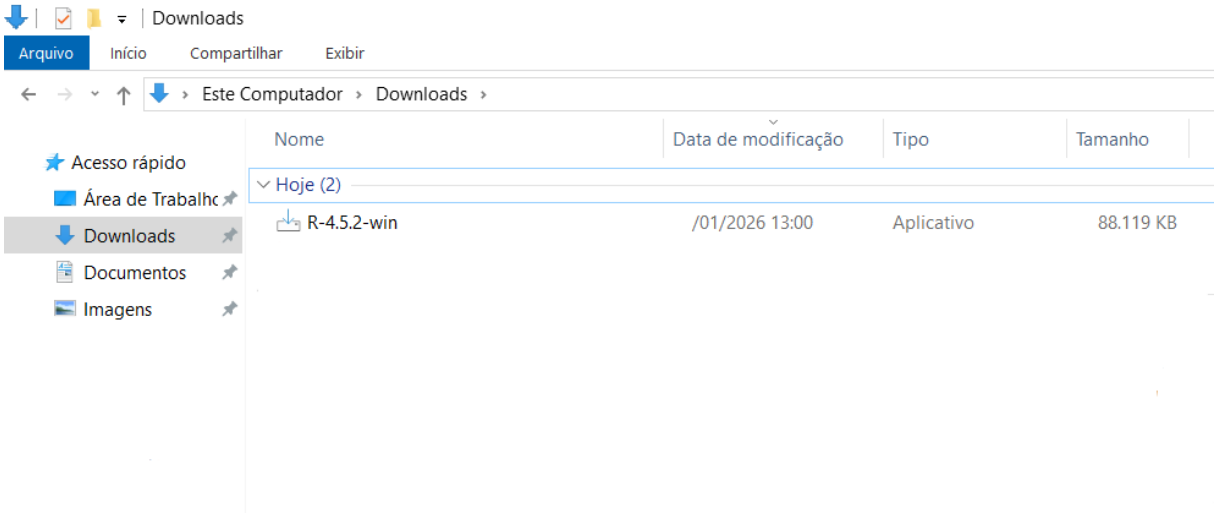
Figura 3 – Página *base* do CRAN com o link para *download* do instalador mais recente do R para Windows



Fonte: Autoras (2026)

Com o instalador baixado, execute o arquivo com um duplo clique para iniciar a instalação. A Figura 4 exemplifica a tela inicial do assistente de instalação. Recomenda-se manter as configurações padrão e avançar pelas etapas clicando em *Next* (ou *Avançar*), a depender do idioma do sistema operacional.

Figura 4 – Tela inicial do instalador do R no Windows



Fonte: Autoras (2026)

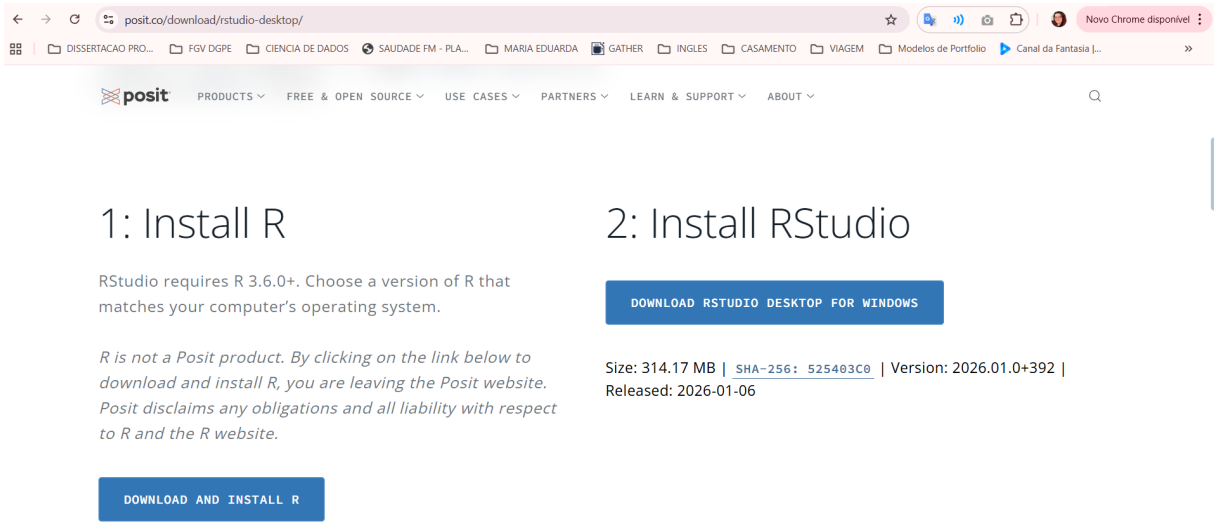
Ao final do processo, conclua a instalação clicando em *Finish* (ou *Concluir*). Com isso, o R estará instalado e pronto para ser utilizado em conjunto com o RStudio, que será instalado na subseção seguinte.

2.1.2 Instalando o RStudio

Após a instalação do R, o próximo passo consiste em instalar o RStudio Desktop, que atuará como ambiente de desenvolvimento integrado (IDE) para facilitar a escrita e execução de códigos. O *download* deve ser realizado diretamente no site da Posit, empresa responsável pelo desenvolvimento do software, por meio do endereço `https://posit.co/download/rstudio-desktop/` (Posit, 2026).

Ao acessar a página, localize a seção de *download* e selecione a versão gratuita (*Open Source License*), conforme apresentado na Figura 5. Ao rolar a página, o site identifica automaticamente o sistema operacional do usuário e disponibiliza o botão de *download* correspondente. No caso do Windows, basta clicar no botão para baixar o instalador executável.

Figura 5 – Página de *download* do RStudio Desktop no site da Posit, com indicação da opção gratuita (*Open Source License*)



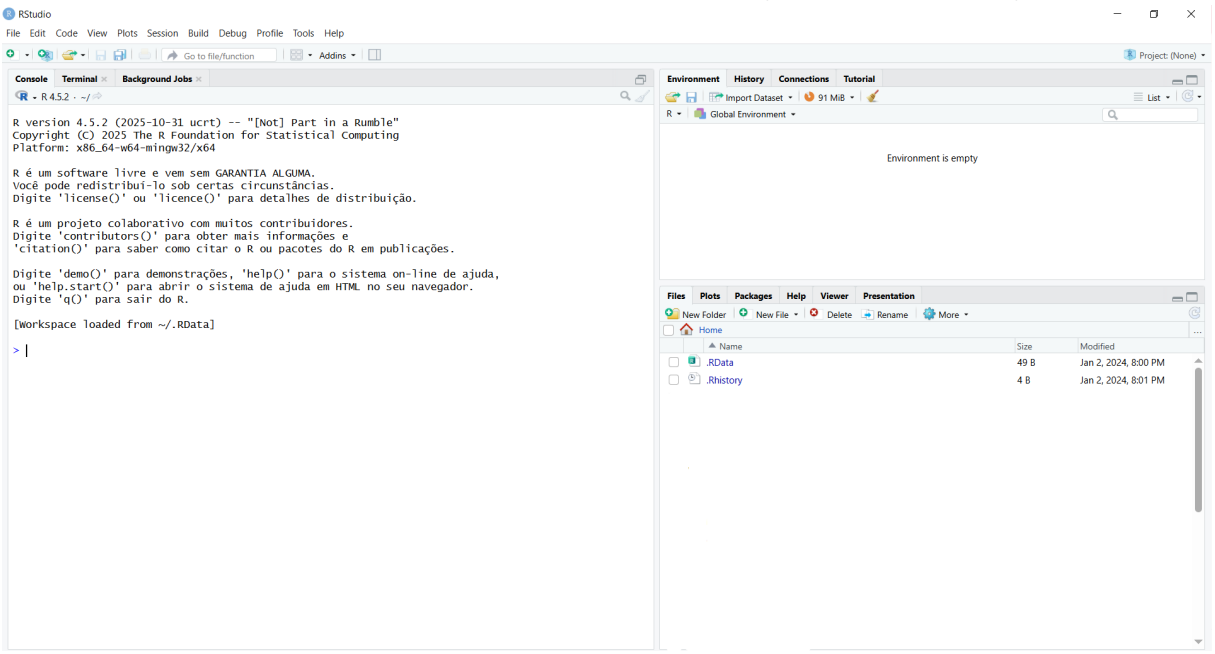
Fonte: Autoras (2026)

Com o arquivo baixado, inicie o instalador com um duplo clique. O assistente de instalação do RStudio é intuitivo e, de forma semelhante à instalação do R, recomenda-se avançar pelas telas clicando em *Next* (ou *Próximo*), mantendo o diretório de instalação padrão sugerido pelo sistema. Ao término, a tela de conclusão será exibida; clique em *Finish* (ou *Concluir*) para finalizar o procedimento.

Para verificar se a instalação foi bem-sucedida e se o RStudio reconheceu corretamente o R instalado anteriormente, abra o programa. A interface inicial deverá

carregar normalmente e o painel *Console* indicará que a sessão está ativa. A Figura 6 apresenta uma visão geral do RStudio no primeiro acesso.

Figura 6 – Visão geral do RStudio após a instalação (primeira execução)



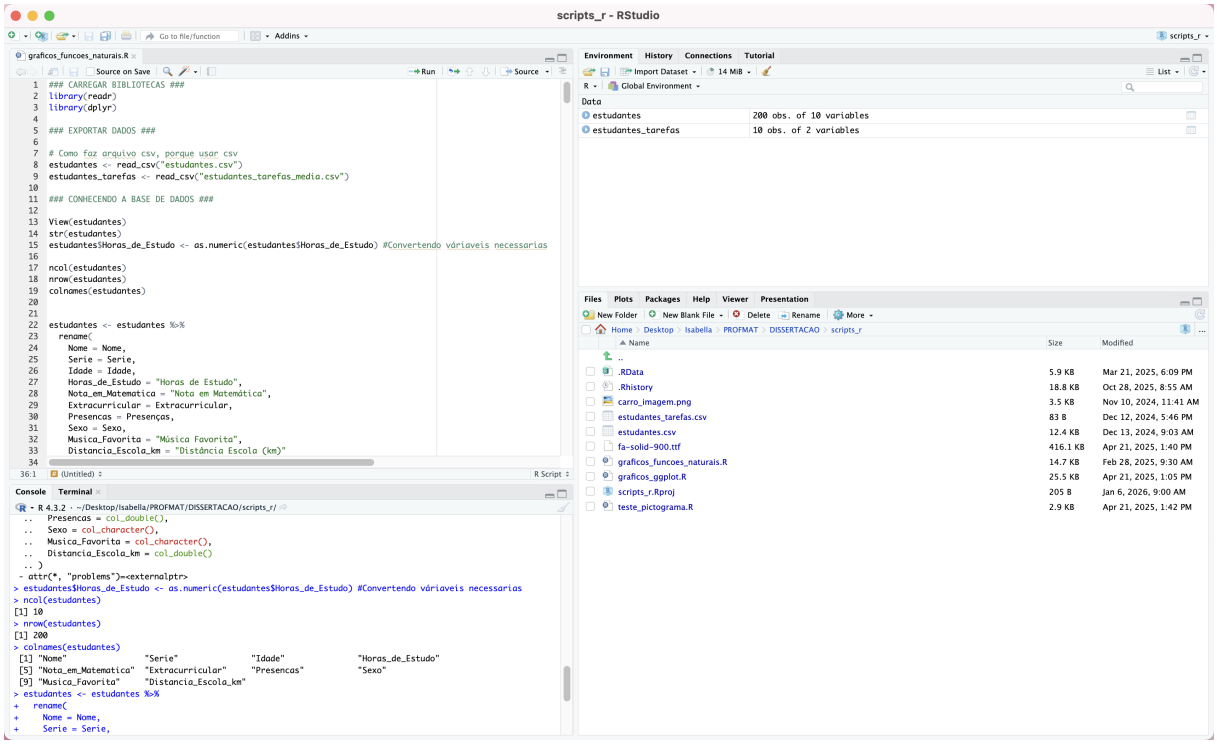
Fonte: Autoras (2026)

Do ponto de vista da interface, o RStudio distribui o trabalho em quatro áreas principais, o que auxilia o usuário a acompanhar, com clareza, o que foi escrito, o que foi executado e o que foi produzido ao longo da análise. Como se observa na Figura 7, o editor de *scripts* (painel superior esquerdo, geralmente identificado como *Source*) é o espaço de escrita e salvamento do código; o *Console* (inferior esquerdo) apresenta a execução dos comandos e seus retornos imediatos; o painel *Environment* (superior direito) lista os objetos criados na sessão (por exemplo, vetores, tabelas e bases de dados); e a área inferior direita integra *Files*, *Plots*, *Packages* e *Help*, organizando arquivos, visualizações e bibliotecas necessárias. Essa configuração favorece a organização do trabalho e facilita retomar e justificar cada etapa do processo.

2.2 Organização do Trabalho: Criando um Projeto e um Script

Para manter uma rotina de trabalho organizada, o RStudio oferece a funcionalidade de projetos (*R Projects*). Trabalhar com projetos é importante porque garante que todos os arquivos relacionados a uma atividade (por exemplo, *scripts*, bases de dados e gráficos gerados) fiquem armazenados em uma única pasta, facilitando a

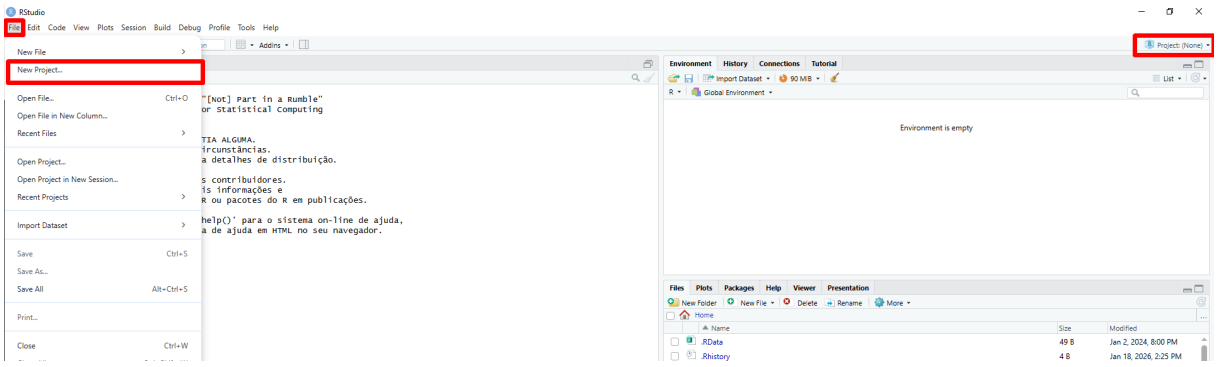
Figura 7 – Interface do RStudio e seus principais painéis de trabalho



Fonte: Autoras (2026)

reprodutibilidade e evitando erros de diretório. Para criar um novo projeto, clique no ícone de projetos no canto superior direito da interface ou acesse o menu *File* > *New Project*, conforme ilustrado na Figura 8.

Figura 8 – Acesso à opção *New Project* no RStudio



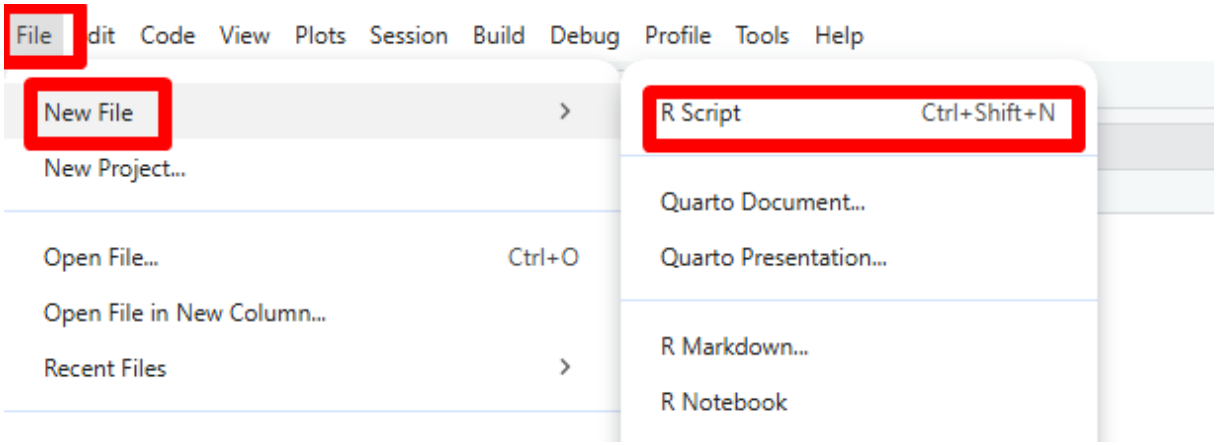
Fonte: Autoras (2026)

Ao selecionar a criação de um novo projeto, opte por *New Directory* (Novo Diretório) e, em seguida, *New Project* (Novo Projeto). Nesta etapa, defina o nome da pasta que conterá os arquivos e o local onde ela será salva no computador. Após clicar em *Create Project* (Criar Projeto), o RStudio será reiniciado já dentro desta nova pasta de trabalho.

Com o projeto ativo, o próximo passo é criar um *script*, que é o arquivo de texto

no qual os comandos serão escritos e salvos. Para isso, acesse *File* > *New File* > *R Script*, como demonstrado na Figura 9. Também é possível criar um novo *script* pelo ícone de novo arquivo no canto superior esquerdo e, então, selecionar *R Script*.

Figura 9 – Acesso à opção *File* > *New File* > *R Script* no RStudio

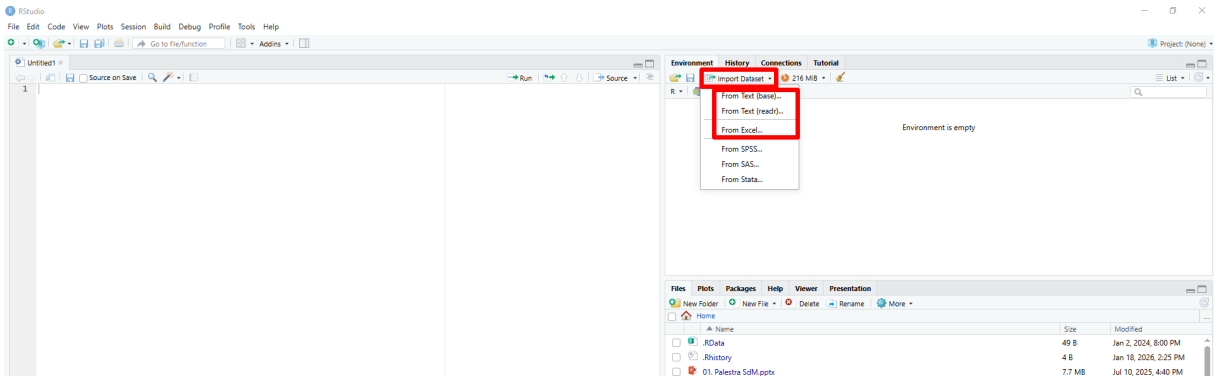


Fonte: Autoras (2026)

2.3 Importação da Base de Dados

Com o ambiente organizado e o projeto criado, é necessário carregar os dados que serão analisados. No painel *Environment* (superior direito), há a opção *Import Dataset*, onde é possível selecionar o tipo de base utilizada (a diferença entre arquivos `.csv` e `.xlsx` será discutida na Seção 3.2), conforme observado na Figura 10. A opção *From Text* é indicada para arquivos `.csv`, enquanto *From Excel* é indicada para arquivos `.xlsx`.

Figura 10 – Opção *Import Dataset* no painel *Environment* do RStudio

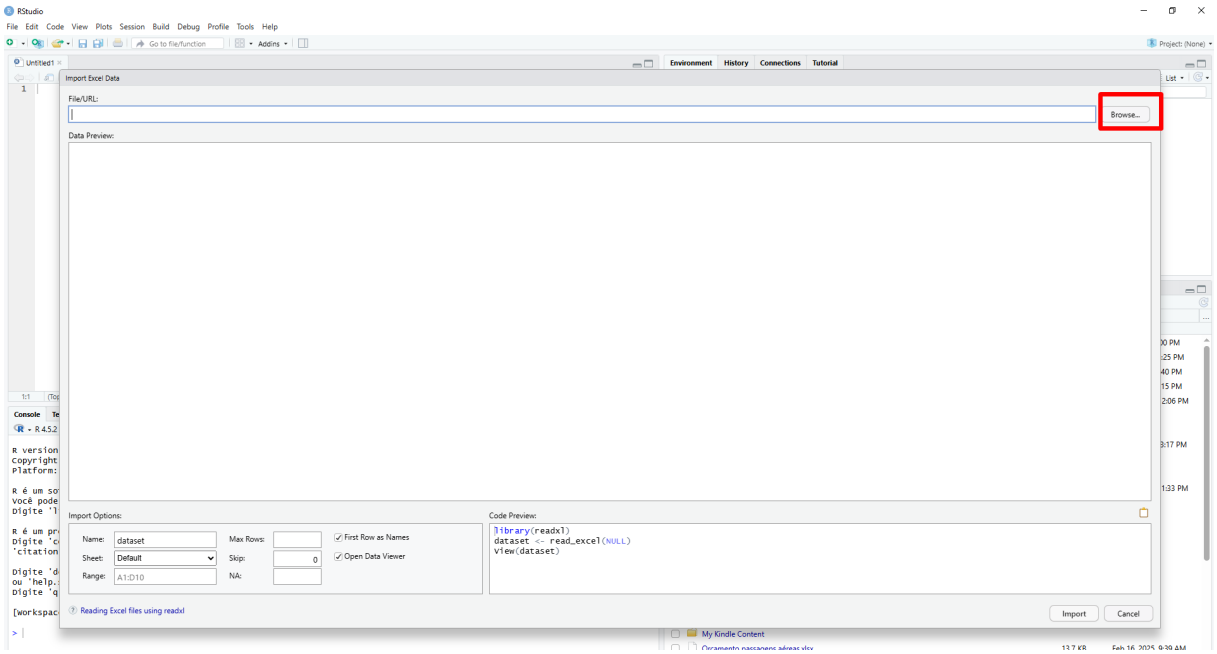


Fonte: Autoras (2026)

Após selecionar a opção apropriada, procure o arquivo desejado para importação. Para isso, clique em *Browse* e localize o arquivo, conforme a Figura 11. Em seguida,

será aberta uma janela de configuração com uma pré-visualização dos dados. Nessa tela, é possível verificar se os nomes das colunas e os tipos de variáveis (numéricas ou categóricas) foram identificados corretamente.

Figura 11 – Janela de importação com o botão *Browse* para selecionar o arquivo de dados



Fonte: Autoras (2026)

Caso a pré-visualização esteja de acordo com o esperado, clique em *Import*, conforme a Figura 12. A partir desse momento, os dados estarão disponíveis no painel *Environment* e prontos para serem utilizados na construção dos gráficos.

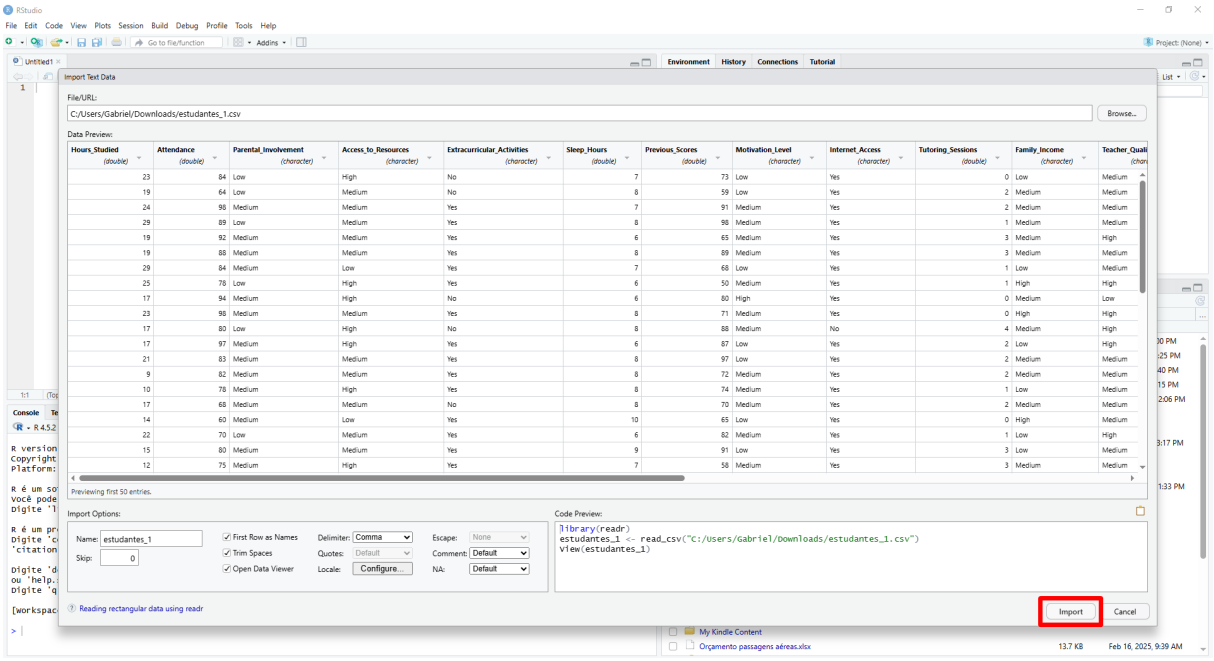
2.4 Instalação e Carregamento de Pacotes

Embora o R possua diversas funções nativas, grande parte do seu potencial de análise e visualização advém dos pacotes, que são coleções de funções desenvolvidas pela comunidade. Neste manual, o pacote fundamental é o `ggplot2` (Wickham, 2016), reconhecido por sua capacidade de criar gráficos altamente personalizáveis.

A instalação de um pacote é realizada apenas uma vez, por meio do comando `install.packages("ggplot2")`, que pode ser digitado diretamente no *Console* ou no *script*, conforme ilustrado na Figura 13. Caso o comando seja escrito no editor (*Source*), utilize o botão *Run* ou o atalho *Ctrl* + *Enter* para executá-lo.

É importante ressaltar que, embora a instalação seja feita apenas uma vez, o carregamento do pacote deve ser realizado sempre que uma nova sessão for ini-

Figura 12 – Botão *Import* para concluir a importação e disponibilizar os dados no *Environment*

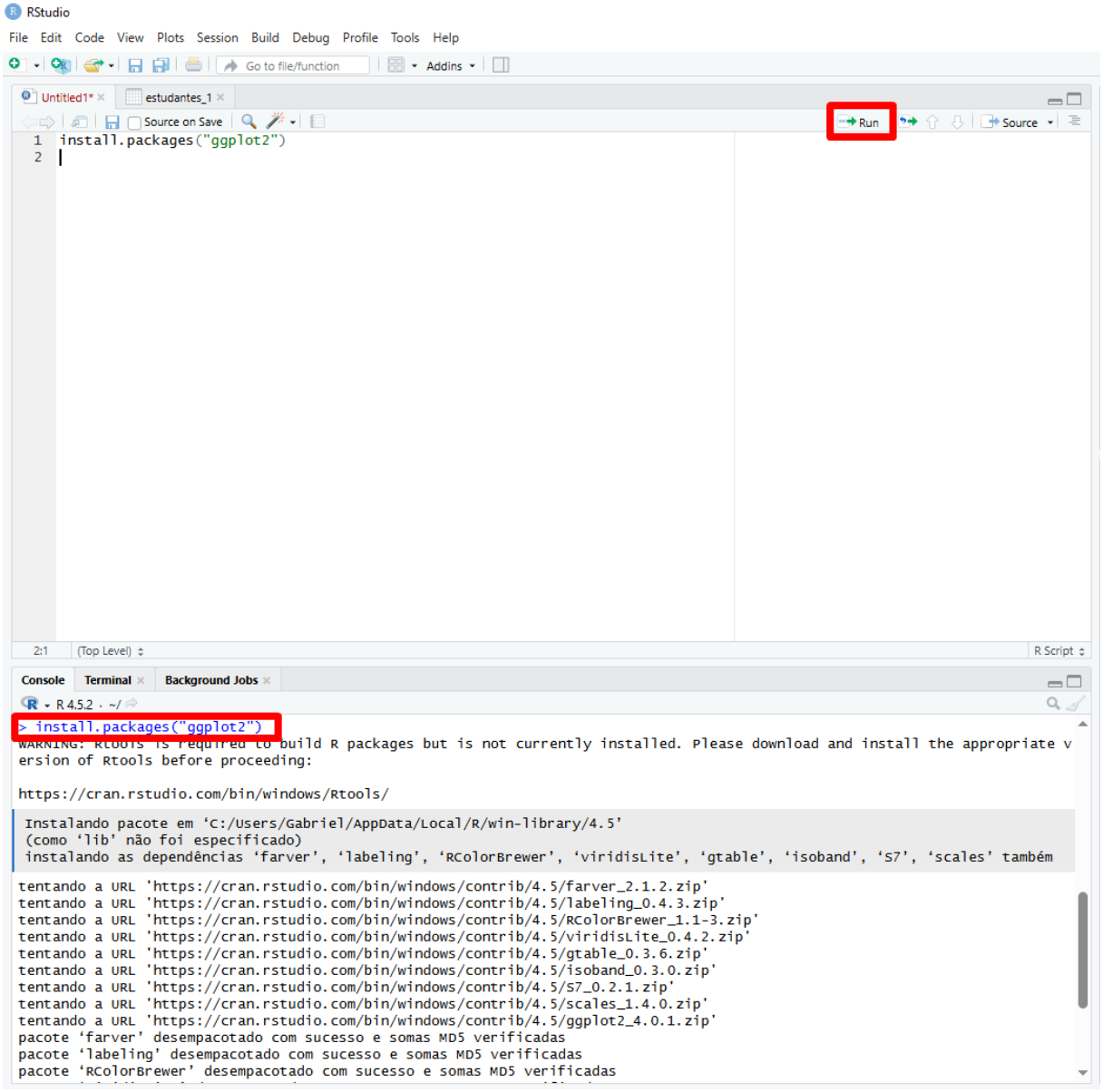


Fonte: Autoras (2026)

ciada (por exemplo, ao abrir novamente o RStudio). Para isso, utiliza-se a função `library(ggplot2)` no *script*. Esse comando ativa as ferramentas do pacote para a sessão atual de trabalho. A Figura 14 demonstra o processo de carregamento e a confirmação de que o pacote está pronto para uso.

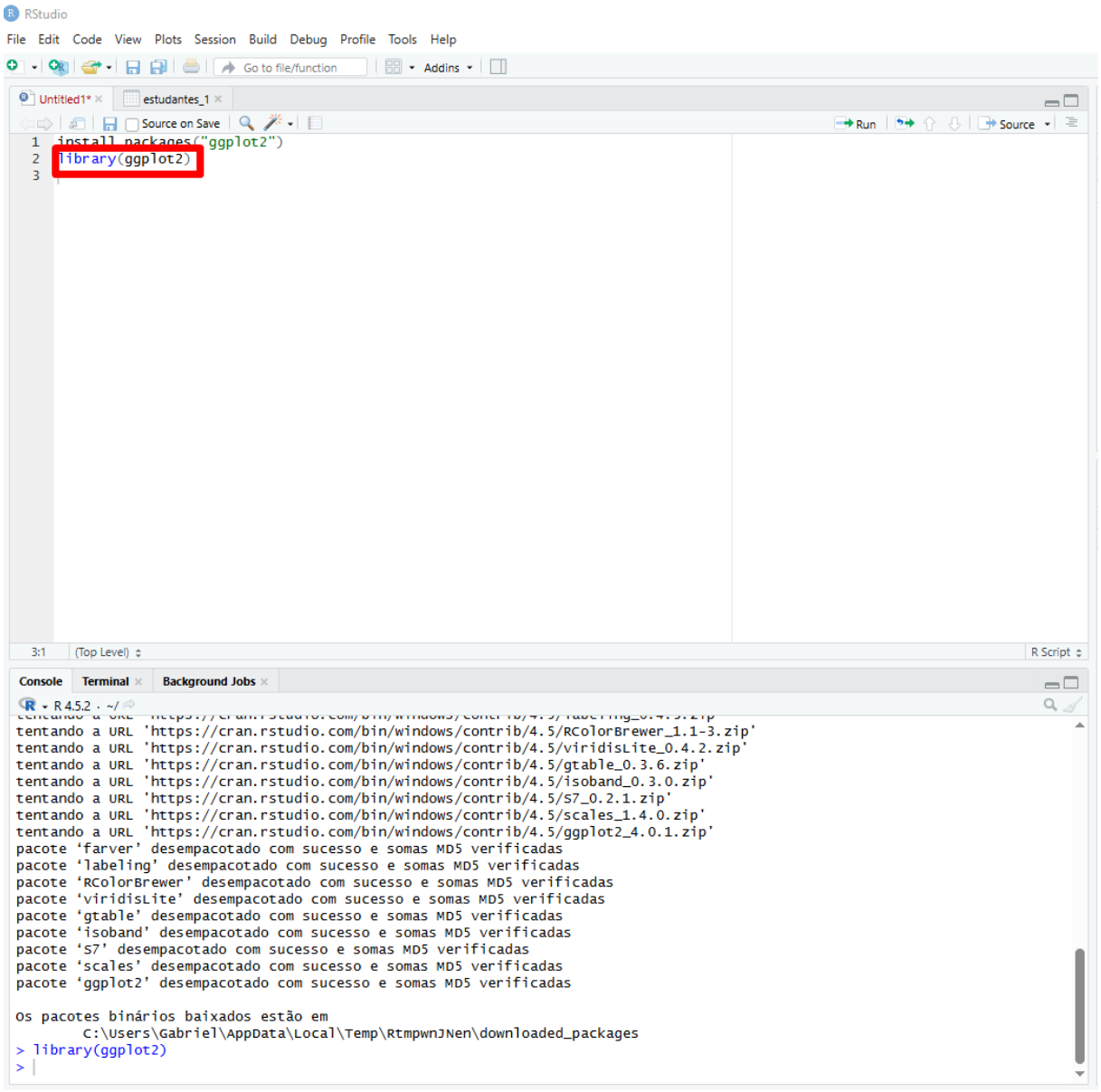
Em síntese, os comandos escritos no *script* devem ser executados por meio do botão *Run* ou pelo atalho *Ctrl* + *Enter*. Neste capítulo utilizamos o `ggplot2` como exemplo por ser o pacote empregado ao longo do manual, mas o procedimento é o mesmo para qualquer pacote.

Figura 13 – Execução do comando `install.packages("ggplot2")` no RStudio



Fonte: Autoras (2026)

Figura 14 – Execução do comando `library(ggplot2)` para carregar o pacote na sessão



Fonte: Autoras (2026)

3 CRIANDO O CONJUNTO DE DADOS

Para realizar análises e construir gráficos no R, é essencial dispor de uma base de dados estruturada, ou seja, um conjunto de informações organizadas que contenham os elementos necessários para os gráficos desejados. Neste recurso educacional, o foco será na construção dos seguintes tipos de gráfico: barras, histograma, setores, linhas e dispersão.

Para todos esses tipos de representação, é possível trabalhar com dados categóricos e numéricos. No entanto, o gráfico de linhas se destaca quando utilizado em uma análise temporal, ou seja, quando se deseja acompanhar a variação de determinada informação ao longo do tempo.

Dessa forma, serão utilizadas duas bases de dados distintas:

1. Uma base com dados categóricos e numéricos, adequada para gráficos de barras, setores, histograma e dispersão – que chamaremos de `estudantes`;
2. Uma base voltada para a análise temporal, especialmente útil para a construção de gráficos de linha – que chamaremos de `estudantes_tarefas`.

O objetivo deste capítulo é apresentar como essas bases foram criadas, para que o leitor possa replicar os exemplos e, se desejar, gerar suas próprias bases com dados mais adequados à sua realidade.

Ao pensar na criação de um conjunto de dados, há duas possibilidades principais: Gerar dados fictícios ou extrair dados (reais ou fictícios).

Para a geração de dados fictícios, será utilizada inteligência artificial (IA) — neste caso, o ChatGPT. Entretanto, essa tarefa pode ser realizada com outras ferramentas de IA disponíveis. Já para a extração de dados reais, uma opção é utilizar bases públicas disponíveis na internet ou em plataformas reconhecidas.

Como este recurso educacional é voltado para professores, uma abordagem prática e acessível é a criação de um formulário online, como o Google Forms, para coleta de dados reais dos próprios alunos. Essa estratégia torna o processo mais contextualizado e próximo da realidade escolar.

Além dessas sugestões, é possível recorrer a bases públicas disponíveis na internet. Entre as opções de dados governamentais, destacam-se o Portal de Dados Abertos do Governo Federal (`https://dados.gov.br/`) e o Instituto Brasileiro de Geografia e Estatística – IBGE (`https://www.ibge.gov.br/`), que oferecem informações sobre

áreas como saúde, educação, segurança pública, economia e outros temas relevantes. Adicionalmente, plataformas como o Kaggle (https://www.kaggle.com/datasets) disponibilizam conjuntos de dados abrangendo diversos assuntos — como filmes, séries, jogos, vendas de restaurantes e saúde — podendo conter informações reais ou simuladas.

Antes da criação do conjunto de dados, é necessário definir quais variáveis serão utilizadas. Como o objetivo é possibilitar a construção de diferentes tipos de gráficos, optou-se por variáveis que se adequam ao contexto escolar e que permitem a representação por meio das visualizações propostas. A seguir, são apresentadas as variáveis escolhidas[1], juntamente com a justificativa para sua inclusão:

- **Idade**: variável numérica, utilizada para a construção de gráfico de barras, possibilitando a visualização da distribuição de idades entre os estudantes;
- **Sexo**: variável categórica, também apropriada para gráfico de barras, permitindo a comparação entre o número de estudantes de cada sexo;
- **Presenca**: variável numérica, também apropriada para gráfico de barras, permitindo a comparação entre o número de estudantes com diferentes quantidades de presença no mês.
- **Nota_em_Matematica**: variável numérica contínua, adequada para a construção de um histograma, possibilitando observar a distribuição das notas obtidas pelos estudantes na avaliação;
- **Serie**: variável categórica (por exemplo: 6º ano, 7º ano, 8º ano), indicada para gráfico de setores, permitindo visualizar a proporção de estudantes por série;
- **Horas_de_Estudo**: variável numérica que será relacionada à `Nota_em_Matematica` por meio de um gráfico de dispersão, permitindo investigar possíveis correlações entre dedicação aos estudos e desempenho acadêmico;
- **Nota**: variável numérica referente à nota de tarefa associada a uma dimensão temporal (**Semana**), apropriada para gráficos de linha, possibilitando acompanhar a evolução do desempenho dos estudantes ao longo do tempo. Estas variáveis compõem a base `estudantes_tarefas`.

[1] Os nomes das variáveis foram padronizados sem acentos e sem “ç” para evitar problemas de codificação e garantir compatibilidade na importação e no uso em diferentes *softwares*.

3.1 Conjunto de Dados gerado por Inteligência Artificial

A criação de dados fictícios mostra-se bastante útil no contexto educacional, sobretudo quando se pretende construir exemplos ilustrativos sem recorrer a dados sensíveis ou reais. Por meio desses dados, é possível explorar diferentes tipos de visualizações e análises no R de forma controlada, ética e segura.

Para esse fim, é necessário que os dados sejam gerados de maneira aleatória e em quantidade suficiente para que os gráficos produzidos sejam significativos. Embora seja viável elaborar manualmente cada valor, o uso de ferramentas de inteligência artificial torna esse processo mais ágil e permite uma geração de dados mais diversificada e realista. Desde que as instruções sejam bem definidas, a inteligência artificial é capaz de criar conjuntos de dados coerentes e adequados ao contexto desejado.

O conjunto de instruções fornecido à inteligência artificial é denominado *prompt*. Um *prompt* consiste em um texto que orienta a ferramenta sobre o conteúdo que deve ser produzido. No presente caso, o *prompt* descreve quais variáveis devem compor o conjunto de dados, quais faixas de valores são esperadas, o tipo de dado (numérico ou categórico), bem como o contexto escolar que se deseja simular.

Com as variáveis devidamente definidas, foi possível elaborar um *prompt* claro e objetivo para a geração dos dados fictícios. Os dados utilizados neste recurso educacional são inteiramente simulados e foram gerados por inteligência artificial, com base em instruções específicas (os arquivos das base de dados geradas estão disponíveis em anexo). A seguir, apresentam-se os *prompts* utilizados para a criação dos dois conjuntos de dados:

Prompt para o dataset `estudantes_tarefas`*:*

> Crie um dataset chamado estudantes_tarefas que traga duas colunas: Semana (variando da semana 1 a 10) e Nota (com notas entre 5 e 10), representando a média de tarefas na semana em uma turma. Ao final, forneça os dados em formato CSV, prontos para serem baixados e utilizados em análises no R.

Prompt para o dataset `estudantes`*:*

Crie para mim um dataset fictício chamado estudantes com 200 observações (linhas), simulando alunos de uma escola do Ensino Fundamental II. Este conjunto de dados será utilizado para criar diferentes tipos de gráficos em R.
As colunas devem ser nomeadas da seguinte forma:
- Nome: Nome do estudante (pode ser um nome aleatório; não será usado em análises, apenas para identificação).
- Serie: Série escolar do estudante, com valores categóricos entre “6º ano”, “7º ano”, “8º ano” e “9º ano”.
- Idade: Idade do estudante, variando entre 11 e 15 anos.
- Horas_de_Estudo: Quantidade média de horas de estudo por dia, valor contínuo entre 0 e 5 horas.
- Nota_em_Matematica: Nota na prova de matemática, variando de 0 a 10. Deve seguir aproximadamente uma distribuição normal, simulando a realidade de desempenho acadêmico. É importante que exista correlação positiva com Horas_de_Estudo, mas também inclua algumas exceções (ex.: alunos que estudam pouco e tiram nota alta, ou estudam muito e tiram nota baixa), para tornar os dados mais realistas.
- Presencas: Número de dias presentes no mês (de um total de aproximadamente 22 dias letivos), com valores típicos.
- Sexo: Gênero do estudante, com os valores “Masculino” e “Feminino”, distribuídos de forma equilibrada.
Ao final, forneça os dados em formato CSV, prontos para serem baixados e utilizados em análises no R.

Esses *prompts* orientaram a ferramenta de inteligência artificial na criação de conjuntos de dados coerentes, contextualizados e apropriados para o ensino de visualizações estatísticas com R.

Com instruções bem definidas, a inteligência artificial é capaz de gerar a base de dados desejada e fornecer um arquivo com a extensão solicitada, pronto para uso em análises. Caso seja necessário, o *prompt* pode ser ajustado ou incrementado para atender às necessidades específicas do projeto.

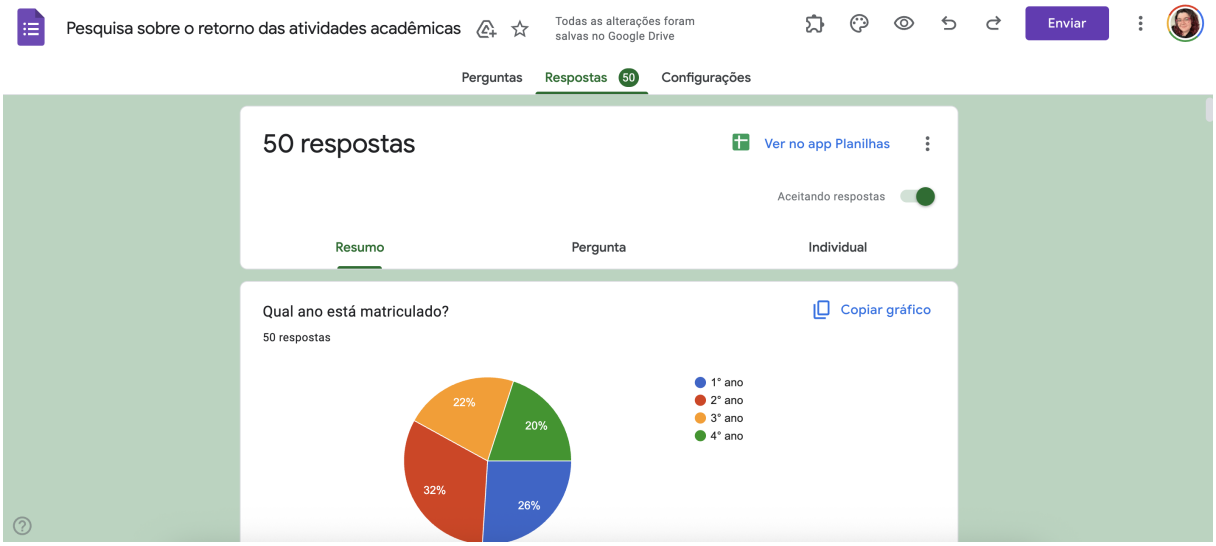
3.2 Conjunto de Dados gerado por Google Formulários

Outra forma de criar um conjunto de dados é a partir de um *Google Formulário*, podendo inclusive envolver a participação dos estudantes na sua elaboração. A docente pode convidá-los a refletir sobre possíveis variáveis relevantes para a construção da base de dados, promovendo um exercício colaborativo e contextualizado.

A criação do formulário é simples e intuitiva, sendo acessível por meio do link: `https://docs.google.com/forms/u/0/`. Após o recebimento das respostas, é possível exportar os dados, etapa essencial para a realização das análises gráficas no R.

Para exportar as respostas, basta acessar a aba “Respostas”, clicar nos três pontos verticais, ao lado do ícone de “Ver no app Planilhas”.

Figura 15 – Aba “Respostas” do *Google Formulários*, com destaque para o ícone dos três pontos verticais (opções)

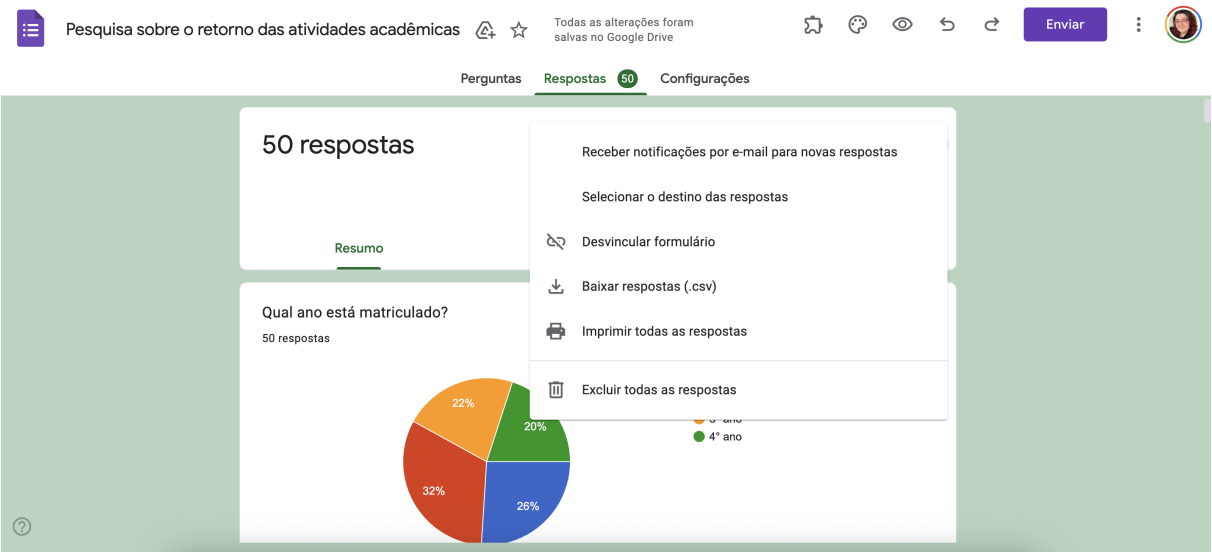


Fonte: Autoras (2026)

Ao clicar nos três pontos, surgirá a opção “Baixar respostas (.csv)”. Essa é a opção recomendada para a obtenção dos dados em um formato compatível com o R.

Vale ressaltar as vantagens do formato CSV: O arquivo CSV (*Comma-Separated Values*) é muito mais leve que arquivos do tipo `.xlsx` - arquivos do tipo Excel- , permitindo o uso de grandes volumes de dados com maior rapidez e menor consumo de memória. Apesar de parecer “quebrado” quando aberto diretamente em alguns programas (por exemplo, com todos os dados em uma única coluna), o CSV é um formato padrão

Figura 16 – Menu de opções exibido após clicar no ícone dos três pontos verticais



Fonte: Autoras (2026)

amplamente reconhecido por softwares de análise de dados, como o `RStudio`, o `Python` e o `Power BI`.

Além disso, o CSV é um formato aberto, ou seja, pode ser lido e manipulado por praticamente qualquer sistema operacional ou linguagem de programação, sem necessidade de ferramentas proprietárias. E também evita incompatibilidades entre versões de programas de planilha, já que trata-se de um formato puro baseado em texto.

Portanto, a escolha do CSV é ideal quando se deseja utilizar os dados em ambientes de análise estatística e visualização, como no caso das aulas com `RStudio`.

4 FUNÇÕES NATIVAS VERSUS GGPLOT2

Segundo a documentação do R (R Core Team, 2024), todas as funções e conjuntos de dados do R são armazenados em *packages*, chamadas aqui de pacotes, e, após serem carregadas, é possível utilizar seu conteúdo.

No caso, o R já vem com uma biblioteca base (ou padrão). Nela, encontram-se alguns pacotes já instalados, como é o caso do pacote `graphics`, que permite a criação de gráficos *base*. Isso significa que esse pacote é nativo do R, ou seja, não é necessário instalá-lo para utilizar suas funções. Essas são as chamadas funções nativas do R.

Para a criação de gráficos, destacam-se as funções `plot()`, `barplot()`, `pie()` e `hist()`, que são os gráficos abordados neste Manual.

Em contrapartida, o R oferece a possibilidade de instalar pacotes adicionais, sendo o `ggplot2` (Wickham, 2016) um dos mais populares para a criação de gráficos. Esse pacote se destaca pela sua flexibilidade e capacidade de personalização na construção de visualizações.

O `ggplot2` segue a teoria desenvolvida por Wilkinson (2005), conhecida como *"The Grammar of Graphics"* (A Gramática dos Gráficos). De acordo com essa teoria, qualquer gráfico estatístico pode ser descrito por um conjunto de camadas ou componentes.

Com base nesse conceito, um gráfico criado com o `ggplot2` é formado por várias camadas que, juntas, compõem a visualização final. Em outras palavras, o processo de construção do gráfico envolve a adição gradual de camadas, sendo que cada uma delas tem a função de definir um aspecto específico da visualização, como os dados, as escalas, as formas geométricas, as legendas, as facetas e as anotações (Faria, 2024).

Utiliza-se a seguinte estrutura básica para os gráficos do ggplot:

```
ggplot(data = df, aes(x = <variável_x> , y = <variável_y> )) +
       tipo_grafico() +
        labs(title = , x = , y = )
```

Cada linha representa uma camada do gráfico e elas são agrupadas com o sinal de +. Assim, na primeira linha define-se qual é a base de dados utilizada (`data`) e quais serão as variáveis (`aes`). Na segunda linha especifica-se qual o tipo de gráfico, como

`geom_bar`, `geom_histogram`, `geom_line`, `geom_point`. Na terceira linha inserem-se os rótulos do gráfico, como título, rótulos do eixo `x` e `y`.

A partir dessa estrutura básica, é possível aprimorar e personalizar os gráficos.

É importante lembrar que o `ggplot2` não é um pacote nativo, portanto, é necessário instalá-lo antes de utilizá-lo, conforme explicado no capítulo 2.

E quais as principais diferenças entre fazer os gráficos com as funções nativas e com o ggplot?

A escolha entre as funções nativas e o `ggplot2` depende de diversos fatores, como a preferência do usuário, o tamanho da base de dados e o nível de complexidade exigido pela visualização. Em termos gerais, as funções nativas do R são frequentemente utilizadas em contextos mais simples, onde a criação de gráficos básicos é suficiente. Por outro lado, o `ggplot2` se destaca em ambientes profissionais e acadêmicos devido à sua grande flexibilidade e à capacidade de criar visualizações mais sofisticadas e personalizáveis.

A seguir, apresentam-se as principais vantagens de utilizar cada uma dessas abordagens.

Base R

- **Simplicidade para gráficos básicos:** Gráficos simples podem ser criados mais rapidamente usando o `base R`, especialmente para visualizações diretas, como gráficos de dispersão ou histogramas.
- **Menos sobrecarga:** Não há necessidade de pacotes adicionais, o que pode simplificar o fluxo de trabalho para tarefas rápidas e simples.
- **Mais controle:** Alguns usuários preferem o controle mais preciso que o `base R` oferece para elementos específicos do gráfico, embora isso possa exigir mais código.

Ggplot2

- **Gramática dos Gráficos:** O `ggplot2` é baseado na Gramática dos Gráficos, tornando-o intuitivo para empilhar gráficos e construir visualizações complexas de forma incremental.
- **Estética e Personalização:** Ele fornece amplas opções de personalização, permitindo gráficos mais visualmente atraentes com menos código.
- **Mais fácil para gráficos complexos:** O `ggplot2` lida com visualizações complexas de forma mais simples, como facetas, que permitem a criação de múltiplos gráficos com base em subconjuntos diferentes de dados.

5 GRÁFICO DE BARRAS

O gráfico de barras é uma das representações gráficas mais utilizadas no ensino de estatística e análise de dados. Ele permite visualizar e comparar a frequência de diferentes categorias de dados de forma intuitiva. Em sala de aula, esse tipo de gráfico é amplamente empregado para apresentar resultados de pesquisas, desempenho de alunos, distribuição de respostas em questionários e outros conjuntos de dados categóricos.

Sua principal vantagem é a facilidade de interpretação, tornando-se uma ferramenta essencial para introduzir conceitos estatísticos e promover a análise crítica dos dados pelos alunos.

Inicialmente, será demonstrado como plotar o gráfico de barras utilizando funções nativas do R e, posteriormente, utilizando funções da biblioteca `ggplot2`. Serão apresentadas as linhas de código e, ao lado, o gráfico gerado por essas linhas.

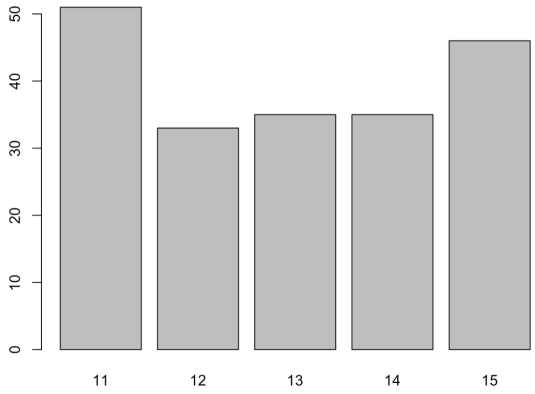
As Figuras apresentadas neste capítulo são de autoria própria, geradas a partir da execução dos códigos apresentados ao longo do texto. Assim, cada imagem corresponde diretamente à saída gráfica produzida pelos scripts indicados na seção.

5.1 Gráfico de Barras por funções nativas

A função nativa do R para a criação de gráficos de barras é `barplot`. Para consultar os parâmetros disponíveis para essa função, utiliza-se o comando `?barplot`, que exibe a documentação correspondente em inglês.

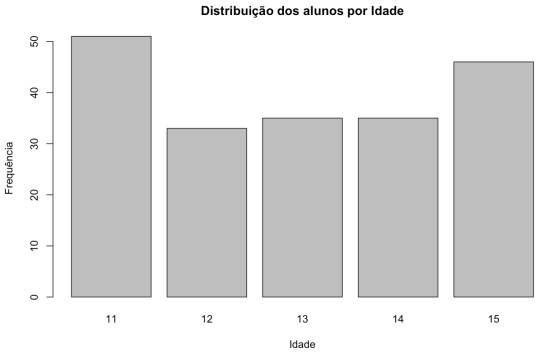
Para gerar um gráfico de barras, é necessário escolher a variável da base de dados que será representada graficamente. Neste exemplo, utiliza-se a variável `Idade` da base de dados `estudantes`. Para isso, calcula-se a tabela de frequências dessa variável e, em seguida, aplica-se a função `barplot`, conforme ilustrado no exemplo abaixo:

```
barplot(table(estudantes$Idade))
```



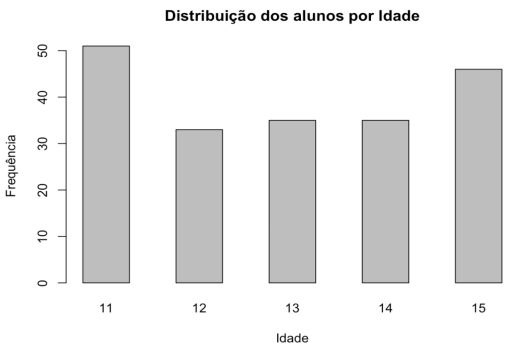
Para inserir os rótulos do gráfico utiliza-se os seguintes argumentos: `main` que insere o título principal do gráfico, `xlab` para inserir o rótulo do eixo x e `ylab` para inserir o rótulo do eixo y.

```
barplot(table(estudantes$Idade),
        main = "Distribuição dos
                alunos por Idade",
        xlab = "Idade",
        ylab = "Frequência")
```



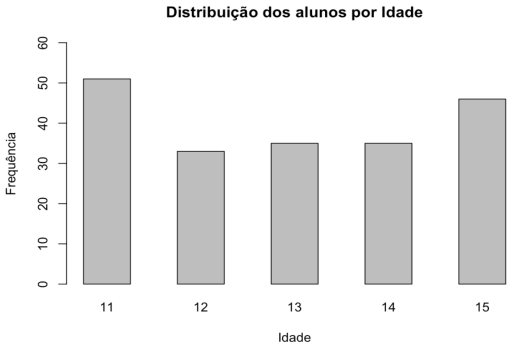
O argumento `space` define o espaço entre as barras de forma proporcional à largura de uma barra. Por exemplo, se `space = 0.5`, o espaço entre as barras será 50% da largura de uma barra. Se `space` for 1, o espaço entre as barras será 100% da largura de uma barra, ou seja, o mesmo tamanho da largura de uma barra.

```
barplot(table(estudantes$Idade),
        main = "Distribuição dos
                alunos por Idade",
        xlab = "Idade",
        ylab = "Frequência",
        space = 1)
```



A função `barplot` automaticamente define os limites numéricos dos eixos `x` e `y`, mas é possível alterar esses limites utilizando os argumentos `xlim` e `ylim`, passando os valores no formato `(valor_inicio, valor_final)`. No exemplo abaixo, o valor de `x` é mantido inalterado, enquanto o limite de `y` é alterado para `(0, 60)`, ou seja, o valor inicial do eixo `y` é 0 e o valor máximo do eixo `y` é 60.

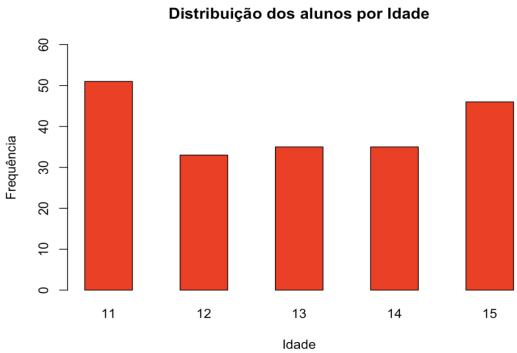
```
barplot(table(estudantes$Idade),
        main = "Distribuição dos
                alunos por Idade",
        xlab = "Idade",
        ylab = "Frequência",
        space = 1,
        ylim = c(0, 60))
```



Para inserir cor às barras do gráfico, utiliza-se o argumento `col`. As cores podem ser editadas de duas formas: aplicando uma única cor para todas as barras ou atribuindo uma cor diferente para cada categoria representada.

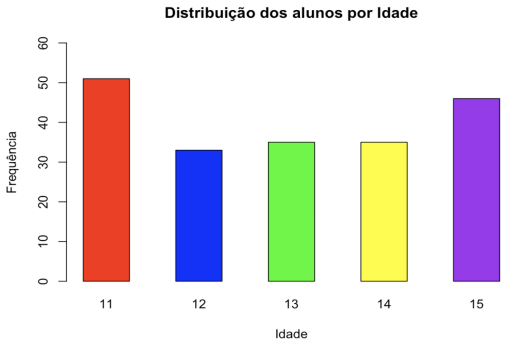
Para definir uma única cor, basta especificá-la em inglês. Além disso, é possível visualizar a lista de cores disponíveis no R utilizando o comando `colors()` no console.

```
barplot(table(estudantes$Idade),
        main = "Distribuição dos
                alunos por Idade",
        xlab = "Idade",
        ylab = "Frequência",
        space = 1,
        ylim = c(0, 60),
        col = "red")
```



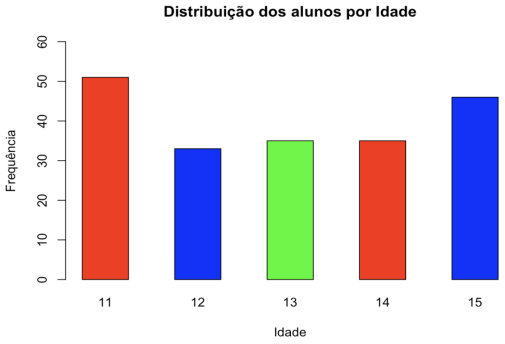
Para definir várias cores no gráfico, atribuindo uma cor diferente para cada categoria, utiliza-se um vetor com todas as cores desejadas. Para isso, emprega-se a função `c("cor_1", "cor_2", ...)`, com os nomes das cores em inglês. A cor correspondente a `cor_1` será aplicada à primeira barra, `cor_2` à segunda barra e assim sucessivamente.

```
barplot(table(estudantes$Idade),
        main = "Distribuição dos
                alunos por Idade",
        xlab = "Idade",
        ylab = "Frequência",
        space = 1,
        ylim = c(0, 60),
        col = c("red", "blue",
                "green", "yellow",
                "purple"))
```



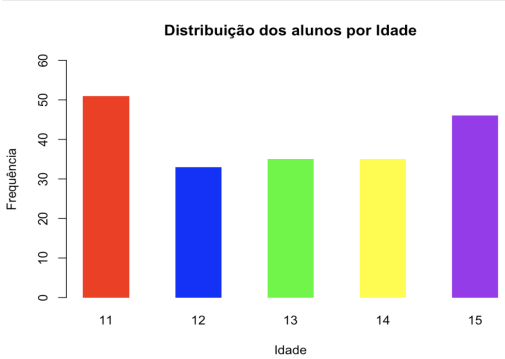
Se o vetor apresentar menos cores do que a quantidade de barras, essas cores serão repetidas sucessivamente para as próximas barras, conforme o exemplo abaixo.

```
barplot(table(estudantes$Idade),
        main = "Distribuição dos
                alunos por Idade",
        xlab = "Idade",
        ylab = "Frequência",
        space = 1,
        ylim = c(0, 60),
        col = c("red", "blue",
                "green"))
```



Outro elemento gráfico interessante são as bordas das barras. É possível ajustar a espessura das bordas, modificar seu estilo ou até mesmo removê-las, o que é bastante comum. Para isso, utiliza-se o argumento `border`. Caso deseje remover as bordas, basta atribuir o valor `NA`, indicando ausência de borda.

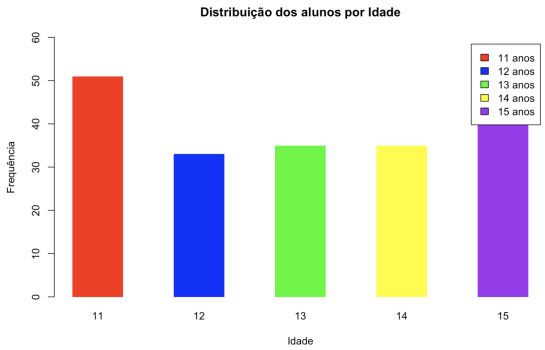
```
barplot(table(estudantes$Idade),
        main = "Distribuição dos
                alunos por Idade",
        xlab = "Idade",
        ylab = "Frequência",
        space = 1,
        ylim = c(0, 60),
        col = c("red", "blue",
                "green", "yellow",
                "purple"),
        border = NA)
```



Ao utilizar gráficos de barras com categorias distintas, é útil adicionar uma legenda. Para isso, utiliza-se o argumento `legend.text`, onde insere, de acordo com a ordem das cores, os rótulos que cada cor representa.

Assim como são definidas múltiplas cores, para especificar os rótulos de várias categorias, utiliza-se um vetor, como em `c("categoria_1", "categoria_2", ...)`.

```
barplot(table(estudantes$Idade),
        main = "Distribuição dos
                alunos por Idade",
        xlab = "Idade",
        ylab = "Frequência",
        space = 1,
        ylim = c(0, 60),
        col = c("red", "blue",
                "green"),
        border = NA,
        legend.text = c("11 anos",
            "12 anos", "13 anos",
            "14 anos", "15 anos"))
```

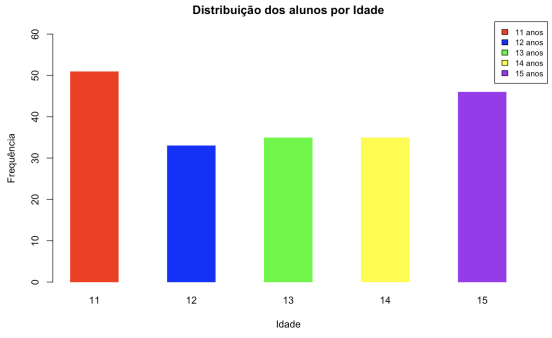


Ao adicionar uma legenda ao gráfico de barras, pode ser necessário ajustá-la para melhorar a legibilidade e evitar sobreposição com as barras ou outros elementos do gráfico. O R permite modificar tanto o tamanho da legenda quanto seu posicionamento por meio do argumento `args.legend`, que recebe uma lista de parâmetros de personalização.

Com esse argumento, é possível, por exemplo, reduzir o tamanho do texto da legenda com `cex`, definir sua posição na área do gráfico com `x`, e ajustar seu deslocamento com `inset`. Esses ajustes garantem que a legenda fique visível sem comprometer a interpretação do gráfico.

Abaixo, apresenta-se um exemplo de uso do argumento `args.legend`, onde a legenda é posicionada no canto superior direito, com um texto ligeiramente menor e um pequeno deslocamento:

```
barplot(table(estudantes$Idade),
        main = "Distribuição dos
                alunos por Idade",
        xlab = "Idade",
        ylab = "Frequência",
        space = 1,
        ylim = c(0, 60),
        col = c("red", "blue",
                "green"),
        border = NA,
        legend.text = c("11 anos",
            "12 anos", "13 anos",
            "14 anos", "15 anos"),
         args.legend = list(cex = 0.8,
                    x = "topright",
                    inset = -0.05))
```

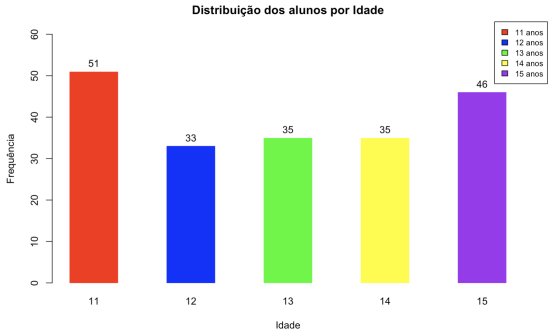


Por fim, é possível adicionar rótulos com os valores numéricos de cada barra do gráfico. Esses valores são independentes do gráfico, ou seja, eles são sobrepostos ao gráfico já gerado, utilizando a função `text`. Para isso, utiliza-se o argumento `labels`, que recebe a tabela de referência com os valores a serem exibidos.

Para posicionar os valores corretamente, é necessário determinar a coordenada `x` e `y` onde os rótulos serão colocados. No caso de um gráfico de barras, é desejado que os valores fiquem centralizados sobre cada barra, o que significa usar a mesma coordenada `x` de cada barra, e o valor de `y` deve ser ligeiramente superior ao topo de cada barra, com um pequeno deslocamento para garantir que os rótulos não fiquem colados às barras.

No exemplo abaixo, o código aplica os ajustes ao gráfico já plotado, adicionando os valores de cada barra de forma centralizada e acima delas:

```
text(
  x = grafico_barras,
  y = table(estudantes$Idade) + 2,
  labels = table(estudantes$Idade)
)
```



Com isso, as principais funcionalidades da função `barplot` para criar gráficos de barras no R foram exploradas, incluindo personalizações como cores, espaçamento, limites dos eixos, legendas e a adição de rótulos numéricos. Essas configurações permitem gerar visualizações mais informativas e bem ajustadas ao contexto dos dados analisados.

Na próxima seção, será apresentado o processo de criação gráficos de barras utilizando a biblioteca `ggplot2`, que oferece uma abordagem mais flexível e estilizada para visualizações de dados.

5.2 Gráfico de Barras pelo ggplot2

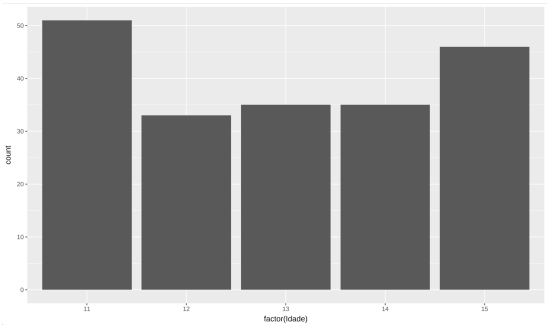
Como mencionado anteriormente no capítulo 4, os gráficos criados a partir do `ggplot2` seguem uma estrutura em camadas, sendo sempre necessário especificar a

base de dados utilizada, as variáveis a serem plotadas no gráfico, bem como o tipo de gráfico desejado.

Dando continuidade ao exemplo apresentado na sessão anterior, será construído um gráfico de barras para a variável `Idade` da base de dados `estudantes`. Para a construção do gráfico de barras, será utilizada a função `geom_bar`.

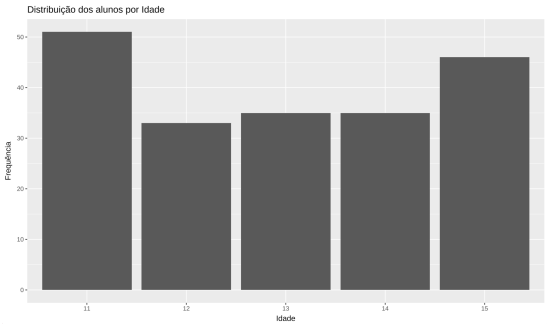
Embora a variável `Idade` seja numérica, neste caso ela será tratada como categórica - ou seja, como valores discretos e não contínuos. Para isso, é necessário convertê-la em fator por meio da função `factor`. Dessa forma, o gráfico contará a frequência com que cada idade aparece na base de dados, conforme exemplo a seguir.

```
ggplot(data = estudantes,
       aes(x = factor(Idade)))+
  geom_bar()
```



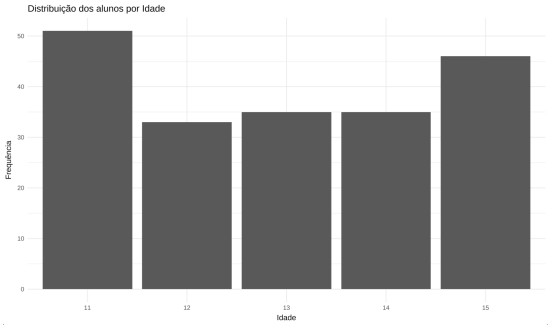
Para inserir os rótulos em um gráfico criado com o pacote `ggplot2`, utiliza-se a função `labs()`, que permite especificar o título principal do gráfico por meio do argumento `title`, o rótulo do eixo x com o argumento `x` e o rótulo do eixo y com o argumento `y`.

```
ggplot(data = estudantes,
       aes(x = factor(Idade)))+
  geom_bar() +
  labs(title = "Distribuição dos
            alunos por Idade",
       x = "Idade",
       y = "Frequência")
```



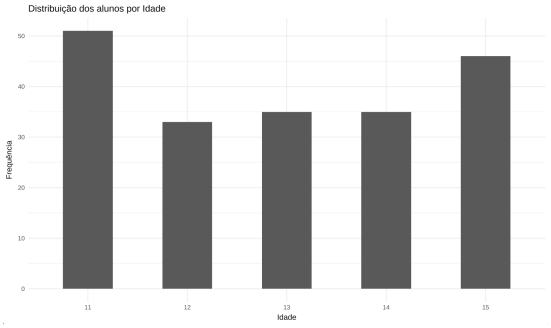
Para deixar o gráfico com um visual mais limpo e agradável, é possível utilizar a função `theme_minimal()`, que aplica um tema com grade (grid) mais discreta e elementos visuais simplificados.

```
ggplot(data = estudantes,
        aes(x = factor(Idade)))+
  geom_bar() +
  labs(title = "Distribuição dos
            alunos por Idade",
       x = "Idade",
       y = "Frequência")+
  theme_minimal()
```



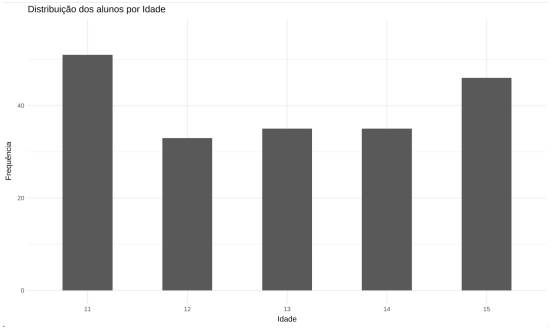
O argumento `width` define a largura das barras no gráfico. Como esse argumento está relacionado à formatação das barras, ele é especificado dentro da função `geom_bar()`. Por exemplo, se `width = 0.5`, as barras terão metade da largura padrão, resultando em mais espaço entre elas. Já se `width = 1`, as barras ocuparão toda a largura disponível, ficando praticamente encostadas umas nas outras.

```
ggplot(data = estudantes,
        aes(x = factor(Idade)))+
  geom_bar(width = 0.5) +
  labs(title = "Distribuição dos
            alunos por Idade",
       x = "Idade",
       y = "Frequência") +
  theme_minimal()
```



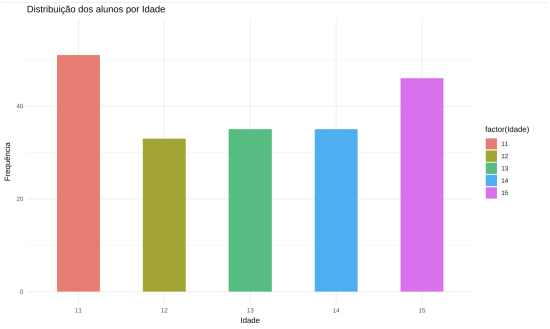
Os limites dos eixos `x` e `y` são definidos automaticamente pelo `ggplot2`, com base nos dados e na geometria escolhida (como `geom_bar()`). No entanto, é possível ajustá-los manualmente utilizando as funções `xlim()` e `ylim()`, passando os valores no formato `(valor_inicial, valor_final)`. No exemplo abaixo, os valores do eixo `x` são mantidos (por isso não foi utilizada a função `xlim`), enquanto os limites do eixo `y` são definidos como `(0, max(table(estudantes$Idade)) + 5)` — ou seja, o eixo `y` começa em 0 e vai até o valor da maior frequência observada acrescido de 5, criando uma folga visual no topo do gráfico.

```
ggplot(data = estudantes,
       aes(x = factor(Idade)))+
  geom_bar(width = 0.5) +
  labs(title = "Distribuição dos
            alunos por Idade",
       x = "Idade",
       y = "Frequência") +
  theme_minimal() +
  ylim(0,
    max(table(estudantes$Idade)) + 5)
```



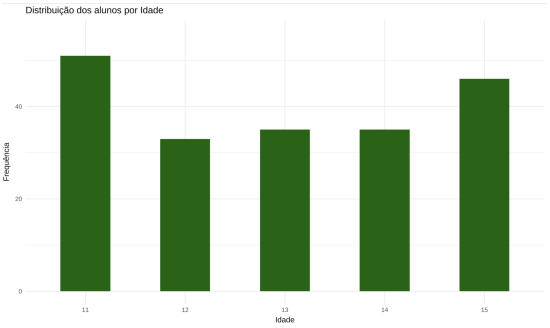
Para adicionar cores às barras do gráfico, utiliza-se o argumento `fill = variável` dentro da função `aes()`, uma vez que as cores estão relacionadas aos dados e suas características, e não à formatação estática do gráfico. Isso faz com que as barras sejam automaticamente coloridas de acordo com as categorias da variável especificada, além de gerar uma legenda correspondente.

```
ggplot(data = estudantes,
       aes(x = factor(Idade),
       fill = factor(Idade)))+
  geom_bar(width = 0.5) +
  labs(title = "Distribuição dos
            alunos por Idade",
       x = "Idade",
       y = "Frequência") +
  theme_minimal() +
  ylim(0,
    max(table(estudantes$Idade)) + 5)
```



Para definir uma única cor para todas as barras do gráfico, utiliza-se o argumento `fill` diretamente dentro da função `geom_bar()`, fora da função `aes()`. Isso indica que a cor não está relacionada a uma variável dos dados, mas sim é uma escolha estética fixa. Além disso, é possível visualizar a lista de nomes de cores disponíveis no R utilizando o comando `colors()` no console.

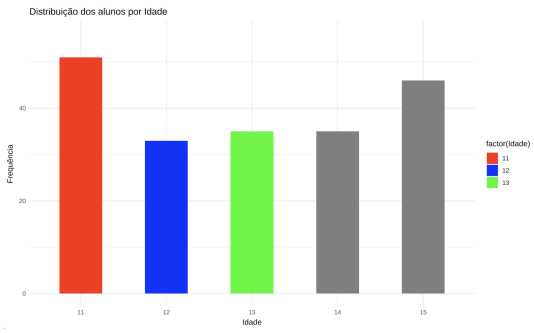
```
ggplot(data = estudantes,
       aes(x = factor(Idade))+
  geom_bar(width = 0.5,
           fill = "darkgreen")+
  labs(title = "Distribuição dos
            alunos por Idade",
       x = "Idade",
       y = "Frequência") +
  theme_minimal() +
  ylim(0,
    max(table(estudantes$Idade)) + 5)
```



Para definir manualmente cores diferentes para cada categoria de uma variável, utiliza-se a função `scale_fill_manual()` juntamente com o argumento `values`, onde é informado um vetor com as cores desejadas. Como as cores estão relacionadas aos valores da variável, o argumento `fill` deve ser incluído dentro da função `aes()`, ou seja, o preenchimento será definido com base nos dados.

No vetor passado ao argumento `values`, foram especificadas quais categorias receberão quais cores. As categorias que não forem explicitamente mencionadas manterão a cor padrão (geralmente um tom de cinza).

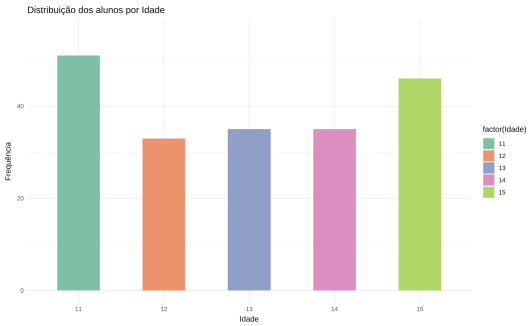
```
ggplot(data = estudantes,
        aes(x = factor(Idade),
        fill = factor(Idade)))+
  geom_bar(width = 0.5) +
  scale_fill_manual(values =
                  c("11" = "red",
                    "12" = "blue",
                    "13" = "green"))+
  labs(title = "Distribuição dos
                alunos por Idade",
       x = "Idade",
       y = "Frequência") +
  theme_minimal() +
  ylim(0,
    max(table(estudantes$Idade)) + 5)
```



A função `scale_fill_brewer()`, juntamente com o argumento `palette`, permite aplicar paletas de cores pré-definidas disponíveis no pacote `RColorBrewer`. Essas paletas são especialmente úteis para garantir uma boa harmonia visual e acessibilidade nos gráficos.

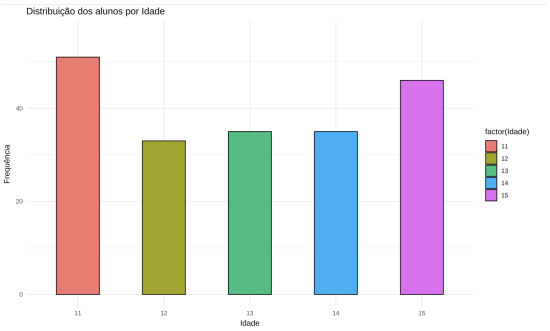
Para utilizá-la, é necessário instalar e carregar o pacote com o comando `library(RColorBrewer)` no início do código. Para visualizar todas as paletas disponíveis e obter mais informações, digite `?RColorBrewer` no console do R.

```
ggplot(data = estudantes,
        aes(x = factor(Idade),
        fill = factor(Idade)))+
  geom_bar(width = 0.5) +
  scale_fill_brewer(palette = "Set2")+
  labs(title = "Distribuição dos
            alunos por Idade",
       x = "Idade",
       y = "Frequência") +
  theme_minimal() +
  ylim(0,
    max(table(estudantes$Idade)) + 5)
```



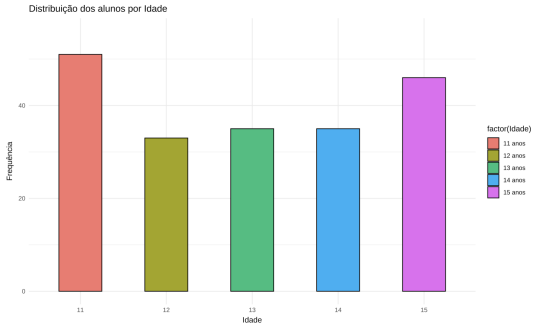
Outro elemento gráfico interessante são as bordas das barras, que podem ser personalizadas com o argumento color dentro da função `geom_bar()`. Esse argumento define a cor do contorno de cada barra. Por exemplo, para adicionar bordas pretas, usamos `color = "black"`. Caso se deseje remover completamente as bordas, basta atribuir o valor `color = NA`, indicando a ausência de cor para o contorno.

```
ggplot(data = estudantes,
        aes(x = factor(Idade),
        fill = factor(Idade)))+
  geom_bar(width = 0.5,
            color = "black") +
  labs(title = "Distribuição dos
                alunos por Idade",
       x = "Idade",
       y = "Frequência") +
  theme_minimal() +
  ylim(0,
    max(table(estudantes$Idade)) + 5)
```



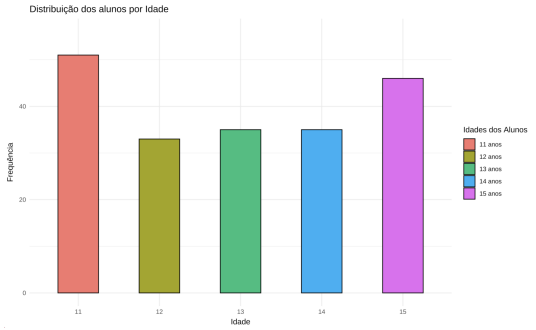
O `ggplot` adiciona automaticamente legendas quando necessário, mas muitas vezes é interessante personalizá-las. Para modificar os rótulos exibidos na legenda, podem ser utilizadas as funções `scale_fill_manual` (caso já esteja sendo usada para definir cores manualmente) ou `scale_fill_discrete`. Ambas aceitam o argumento `labels`, que deve conter um vetor com os nomes desejados, como em `c("categoria_1", "categoria_2", ...)`.

```
ggplot(data = estudantes,
        aes(x = factor(Idade),
        fill = factor(Idade)))+
  geom_bar(width = 0.5,
           color = "black") +
  scale_fill_discrete(labels =
                        c("11 anos",
                        "12 anos",
                        "13 anos",
                        "14 anos",
                        "15 anos"))+
  labs(title = "Distribuição dos
                alunos por Idade",
       x = "Idade",
       y = "Frequência") +
  theme_minimal() +
  ylim(0,
    max(table(estudantes$Idade)) + 5)
```



Para modificar o título exibido na legenda, utiliza-se o argumento `fill` dentro da função `labs()`. Assim, é possível substituir o nome padrão — que normalmente corresponde ao nome da variável utilizada no `fill` — por um título mais descritivo.

```
ggplot(data = estudantes,
        aes(x = factor(Idade),
        fill = factor(Idade)))+
  geom_bar(width = 0.5,
           color = "black") +
  scale_fill_discrete(labels =
                        c("11 anos",
                        "12 anos",
                        "13 anos",
                        "14 anos",
                        "15 anos"))+
  labs(title = "Distribuição dos
                alunos por Idade",
       x = "Idade",
       y = "Frequência",
       fill = "Idades dos Alunos"))+
  theme_minimal() +
  ylim(0,
    max(table(estudantes$Idade)) + 5)
```



Por fim, é possível adicionar rótulos com os valores numéricos de cada barra do gráfico, utilizando a função `geom_text` com o argumento `label`, que recebe os valores a serem exibidos.

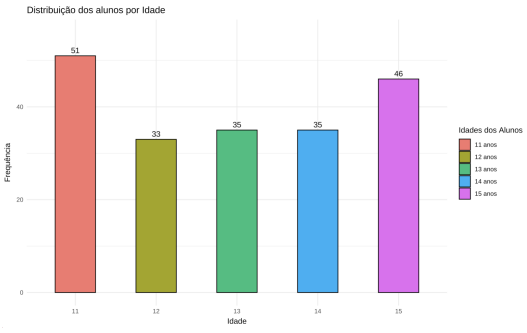
Utiliza-se o argumento `aes(label = after_stat(count))` para definir os rótulos que serão exibidos nas barras. O `aes()` é utilizado para mapear variáveis dos

dados para elementos do gráfico. Neste caso, serão mapeados os valores de label para exibir as contagens de cada categoria. A função `after_stat(count)` é responsável por calcular a contagem das observações de cada categoria (por exemplo, a quantidade de alunos em cada faixa etária). Assim, serão adicionados os valores dessas contagens como rótulos em cada barra do gráfico.

Além disso, o argumento `stat = "count"` é usado para garantir que o `geom_text()` calcule a contagem de ocorrências de cada categoria, pois por padrão o `geom_text()` tentaria usar os valores do eixo y, o que não seria adequado nesse caso. O `stat = "count"` instrui o R a contar as ocorrências nas barras para gerar os rótulos de forma adequada.

Para ajustar a posição vertical dos rótulos, utilizamos o argumento `vjust`. O valor padrão de `vjust` é 0, o que coloca o texto centralizado verticalmente sobre a barra. No entanto, ao definir vjust = -0.5, movemos os rótulos ligeiramente para cima, para que eles não fiquem colados nas barras e fiquem mais visíveis, o que é especialmente útil quando as barras são altas.

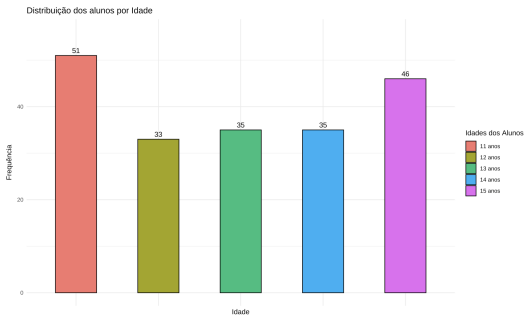
```
ggplot(data = estudantes,
        aes(x = factor(Idade),
        fill = factor(Idade)))+
  geom_bar(width = 0.5,
           color = "black") +
geom_text(aes(
        label = after_stat(count)),
        stat = "count",
        vjust = -0.5) +
  scale_fill_discrete(labels =
                        c("11 anos",
                        "12 anos",
                        "13 anos",
                        "14 anos",
                        "15 anos"))+
  labs(title = "Distribuição dos
                alunos por Idade",
       x = "Idade",
       y = "Frequência",
       fill = "Idades dos Alunos"))+
  theme_minimal() +
  ylim(0,
    max(table(estudantes$Idade)) + 5)
```



Com o uso da legenda lateral, pode-se optar por esconder o nome das categorias que aparecem no eixo x do gráfico de barras, especialmente quando a legenda já está fornecendo a informação necessária. Para isso, utiliza-se a função `theme()` com o argumento `axis.text.x`. O valor que deve ser atribuído a esse argumento para remover o texto do eixo x é `element_blank()`. Essa configuração fará com que os rótulos das

categorias do eixo x desapareçam do gráfico, mantendo apenas a legenda lateral, o que pode ser útil em gráficos com muitos rótulos ou quando se deseja um visual mais limpo e sem repetições de informações.

```
ggplot(data = estudantes,
        aes(x = factor(Idade),
        fill = factor(Idade)))+
  geom_bar(width = 0.5,
           color = "black") +
geom_text(aes(
        label = after_stat(count)),
        stat = "count",
        vjust = -0.5) +
  scale_fill_discrete(labels =
                        c("11 anos",
                        "12 anos",
                        "13 anos",
                        "14 anos",
                        "15 anos"))+
  labs(title = "Distribuição dos
                alunos por Idade",
       x = "Idade",
       y = "Frequência",
       fill = "Idades dos Alunos"))+
  theme_minimal() +
  theme(axis.text.x=element_blank())+
  ylim(0,
    max(table(estudantes$Idade)) + 5)
```



Com isso, foram exploradas as principais funcionalidades para a criação de gráficos de barras no R com o pacote `ggplot2`, incluindo personalizações como cores, espaçamento, limites dos eixos, legendas e a adição de rótulos numéricos. Essas configurações possibilitam a geração de visualizações mais informativas e adequadas ao contexto dos dados analisados.

Na próxima seção, serão apresentados os diferentes posicionamentos das barras, recursos bastante úteis para representar comparações entre categorias de maneira clara e eficiente.

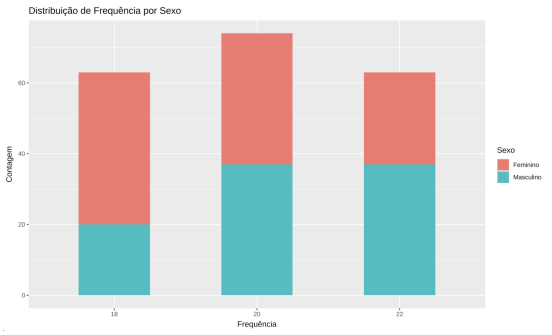
5.3 Personalizando a disposição das barras com `ggplot`

Nesta seção, o foco será apresentar as diferentes disposições das barras que podem ser ajustadas por meio do argumento `position`. As demais personalizações exploradas na seção anterior (5.2) continuam válidas e podem ser aplicadas nos códigos a seguir.

Por padrão, a função `geom_bar` gera um gráfico de barras simples, em que cada barra representa uma categoria do eixo `x`. No entanto, ao incluir o argumento `position`, é possível modificar essa disposição para criar, por exemplo, gráficos de barras empilhadas, barras justapostas (lado a lado) ou gráficos proporcionais. Esse argumento é especialmente útil quando estamos lidando com duas variáveis categóricas: uma no eixo `x` e outra usada para agrupar os dados com o argumento `fill`.

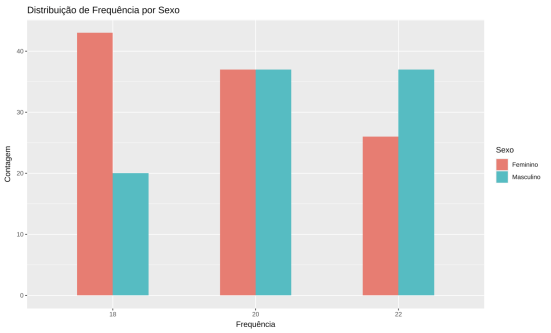
Para criar um gráfico de barras empilhadas, utiliza-se `position = "stack"`. Nesse caso, as barras são empilhadas verticalmente, somando as alturas de cada categoria. Esse tipo de gráfico é útil para visualizar tanto o total de ocorrências quanto a proporção relativa entre os grupos dentro de cada categoria.

```
ggplot(estudantes,
       aes(x = factor(Presencas),
       fill = factor(Sexo))) +
  geom_bar(position = "stack",
           width = 0.5) +
  labs(title = "Distribuição de
       Frequência por Sexo",
      x = "Frequência",
      y = "Contagem",
      fill = "Sexo")
```



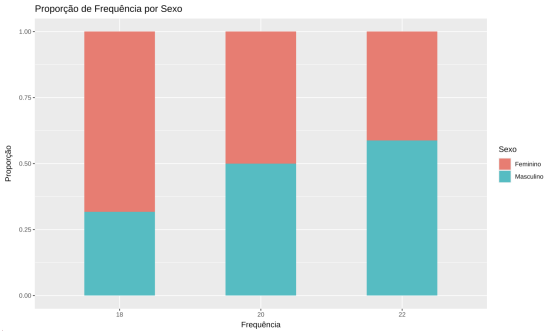
Para criar um gráfico de barras justapostas, utiliza-se `position = "dodge"`. Nesse caso, as barras são separadas por categorias lado a lado, em vez de empilhar. É útil para comparações diretas entre grupos.

```
ggplot(estudantes,
       aes(x = factor(Presencas),
       fill = factor(Sexo))) +
  geom_bar(position = "dodge",
           width = 0.5) +
  labs(title = "Distribuição de
       Frequência por Sexo",
      x = "Frequência",
      y = "Contagem",
      fill = "Sexo")
```



Para criar um gráfico de barras proporcionais, utiliza-se `position = "fill"`. Nesse caso, as barras são empilhadas, mas com uma altura fixa representando 100%. Cada barra representa proporções relativas dos grupos.

```
ggplot(estudantes,
        aes(x = factor(Presencas),
        fill = factor(Sexo))) +
  geom_bar(position = "fill",
           width = 0.5) +
  labs(title = "Distribuição de
        Frequência por Sexo",
       x = "Frequência",
       y = "Contagem",
       fill = "Sexo")
```



Com isso, encerra-se a apresentação das principais funcionalidades do pacote `ggplot2` aplicadas à construção de gráficos de barras. Ao longo das seções, foram abordados desde os elementos básicos da criação desses gráficos até personalizações mais avançadas, como ajustes de cores, rótulos, legendas, limites dos eixos e diferentes disposições das barras. Essas ferramentas permitem que o usuário desenvolva visualizações mais claras, organizadas e alinhadas ao contexto dos dados, facilitando a interpretação e a comunicação dos resultados.

No próximo Capítulo, será abordada a construção de histogramas, um tipo de gráfico especialmente útil para representar a distribuição de variáveis quantitativas. Inicialmente, serão utilizadas as funções nativas do R para esse tipo de visualização e, em seguida, será apresentada a construção de histogramas com o pacote `ggplot2`.

6 HISTOGRAMA

O histograma é uma representação gráfica que permite visualizar e analisar a distribuição de frequências em intervalos, sendo especialmente útil para dados contínuos. Em sala de aula, esse tipo de gráfico pode ser utilizado para analisar o desempenho dos alunos por meio das notas, estudar fenômenos naturais e explorar outros conjuntos de dados contínuos.

Sua principal vantagem é a facilidade de interpretação, auxiliando na compreensão das medidas de tendência central, na identificação da simetria dos dados e na detecção de valores extremos.

Inicialmente, será demonstrado como plotar o histograma utilizando funções nativas do R e, posteriormente, utilizando funções da biblioteca `ggplot2`. Serão apresentadas as linhas de código e, ao lado, o gráfico gerado por essas linhas.

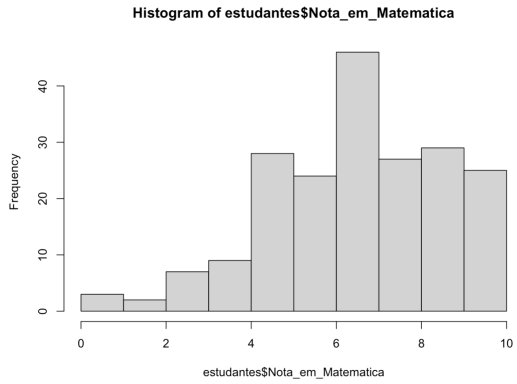
As Figuras apresentadas neste capítulo são de autoria própria, geradas a partir da execução dos códigos apresentados ao longo do texto. Assim, cada imagem corresponde diretamente à saída gráfica produzida pelos scripts indicados na seção.

6.1 Histograma por funções nativas

A função nativa do R para a criação de histogramas é `hist`. Para consultar os parâmetros disponíveis para essa função, utiliza-se o comando `?hist`, que exibirá a documentação correspondente em inglês.

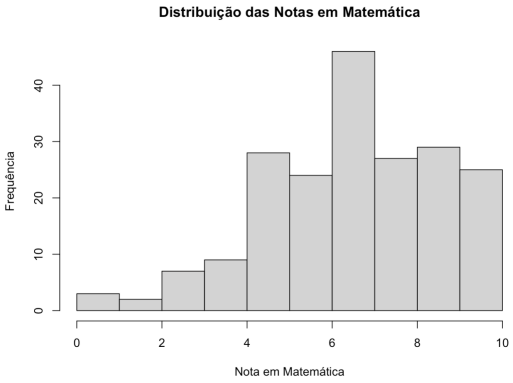
Para gerar um histograma, primeiro é necessário escolher a variável da base de dados que será representada graficamente. Neste exemplo, utiliza-se a variável `Nota_em_Matematica` da base de dados `estudantes`, conforme ilustrado no exemplo abaixo:

```
hist(estudantes$Nota_em_Matematica)
```



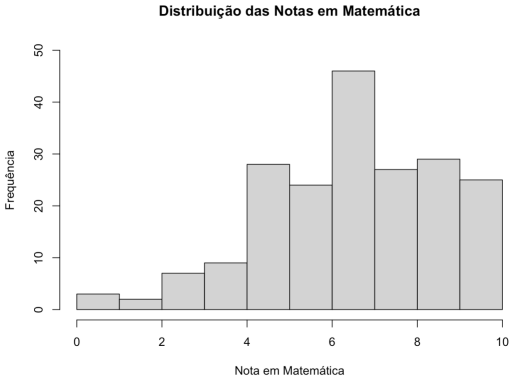
Perceba que, automaticamente, a função insere o título do gráfico e os rótulos dos eixos, baseado na base de dados utilizada. Para personalizar os rótulos do gráfico utiliza-se os seguintes argumentos: `main` que insere o título principal do gráfico, `xlab` para inserir o rótulo do eixo x e `ylab` para inserir o rótulo do eixo y.

```
hist(estudantes$Nota_em_Matematica,
     main = "Distribuição das Notas
             em Matemática",
     xlab = "Nota em Matemática",
     ylab = "Frequência")
```



A função `hist` automaticamente define os limites numéricos dos eixos `x` e `y`, mas esses limites podem ser ajustados utilizando os argumentos `xlim` e `ylim`, passando os valores no formato `(valor_inicio, valor_final)`. No exemplo abaixo, o limite de `x` é alterado para `(0,10)`, ou seja, o valor inicial do eixo `x` é 0 e o valor máximo do eixo `x` é 10. De forma análoga, o limite de `y` é ajustado para `(0, 50)`,conforme os valores da distribuição.

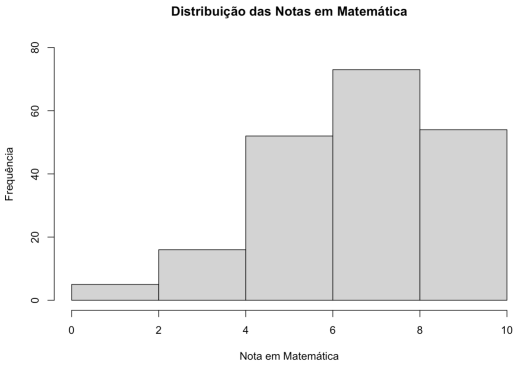
```
hist(estudantes$Nota_em_Matematica,
     main = "Distribuição das Notas
             em Matemática",
     xlab = "Nota em Matemática",
     ylab = "Frequência",
     xlim = c(0, 10),
     ylim = c(0, 50))
```



Também é possível modificar o intervalo das categorias utilizando o argumento `breaks`, que pode ser configurado de duas maneiras: definindo um número fixo de intervalos ou especificando o início, o fim e o tamanho do intervalo.

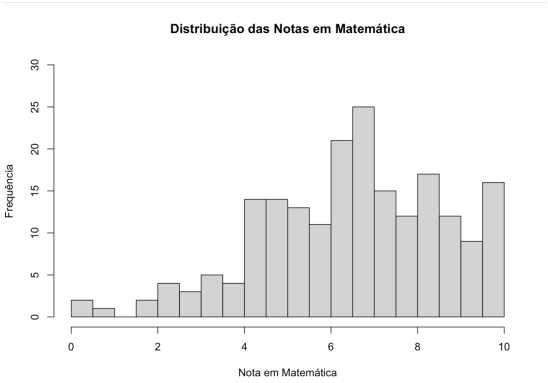
No exemplo a seguir, foram definidos 6 intervalos, mas automaticamente o R ajusta para 5, pois o número total de intervalos deve ser ajustado de acordo com os dados e com a distribuição dos valores.

```
hist(estudantes$Nota_em_Matematica,
     main = "Distribuição das Notas
             em Matemática",
     xlab = "Nota em Matemática",
     ylab = "Frequência",
     xlim = c(0, 10),
     ylim = c(0, 50),
     breaks = 6)
```



A forma alternativa de usar o argumento `breaks` se encontra no exemplo seguinte, seguindo a seguinte estrutura `seq(valor_inicial, valor_final, by = tamanho_intervalo)`. No caso abaixo, o início é 0, o final é 10, e o intervalo é de 0.5 em 0.5:

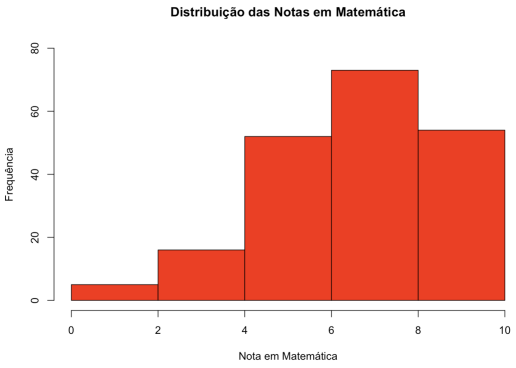
```
hist(estudantes$Nota_em_Matematica,
     main = "Distribuição das Notas
             em Matemática",
     xlab = "Nota em Matemática",
     ylab = "Frequência",
     xlim = c(0, 10),
     ylim = c(0, 50),
     breaks = seq(0, 10, by = 0.5))
```



Para inserir cor ao histograma, utiliza-se o argumento `col`. As cores podem ser editadas de duas formas: aplicando uma única cor para todas os intervalos ou atribuindo uma cor diferente para cada intervalo representado.

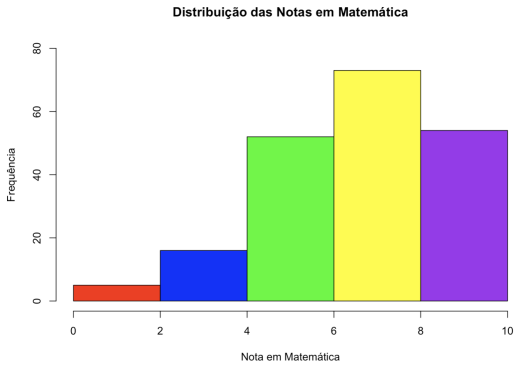
Para definir uma única cor, basta especificá-la em inglês. Além disso, é possível visualizar a lista de cores disponíveis no R utilizando o comando `colors()` no console.

```
hist(estudantes$Nota_em_Matematica,
     main = "Distribuição das Notas
             em Matemática",
     xlab = "Nota em Matemática",
     ylab = "Frequência",
     xlim = c(0, 10),
     ylim = c(0, 80),
     breaks = 5,
     col = "red")
```



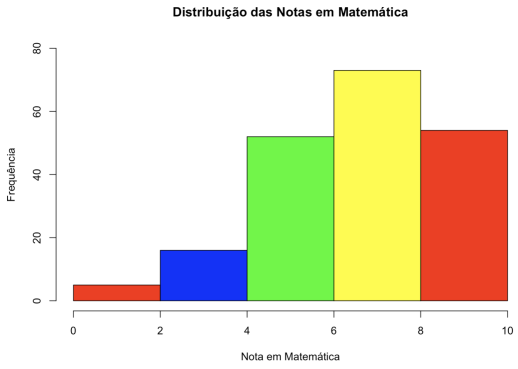
Para definir várias cores no gráfico, atribuindo uma cor diferente para cada intervalo, definimos um vetor com todas as cores desejadas. Para isso, utiliza-se a função `c("cor_1", "cor_2", ...)`, com os nomes das cores em inglês. A cor correspondente a `cor_1` será aplicada ao primeiro intervalo, `cor_2` ao segundo intervalo e assim sucessivamente.

```
hist(estudantes$Nota_em_Matematica,
     main = "Distribuição das Notas
             em Matemática",
     xlab = "Nota em Matemática",
     ylab = "Frequência",
     xlim = c(0, 10),
     ylim = c(0, 80),
     breaks = 5,
     col = c("red", "blue", "green",
             "yellow", "purple"))
```



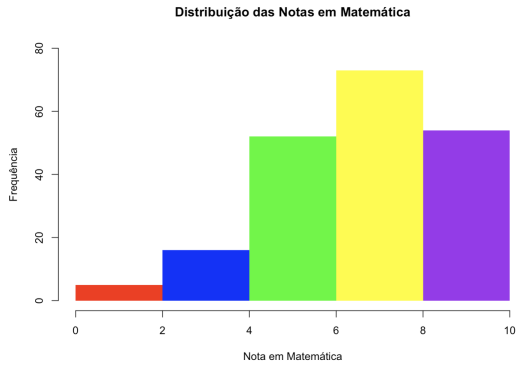
Se o vetor conter menos cores do que a quantidade de intervalos, essas cores serão repetidas sucessivamente para os próximo intervalos, conforme o exemplo abaixo.

```
hist(estudantes$Nota_em_Matematica,
     main = "Distribuição das Notas
             em Matemática",
     xlab = "Nota em Matemática",
     ylab = "Frequência",
     xlim = c(0, 10),
     ylim = c(0, 80),
     breaks = 5,
     col = c("red", "blue", "green",
             "yellow"))
```



Outro elemento gráfico interessante são as bordas das barras do histograma. É possível ajustar a espessura das bordas, modificar seu estilo ou até mesmo removê-las, o que é bastante comum. Para isso, utiliza-se o argumento `border`. Caso deseje remover as bordas, basta atribuir o valor `NA`, indicando ausência de borda.

```
hist(estudantes$Nota_em_Matematica,
     main = "Distribuição das Notas
             em Matemática",
     xlab = "Nota em Matemática",
     ylab = "Frequência",
     xlim = c(0, 10),
     ylim = c(0, 80),
     breaks = 5,
     col = c("red", "blue", "green",
             "yellow", "purple"),
     border = NA)
```



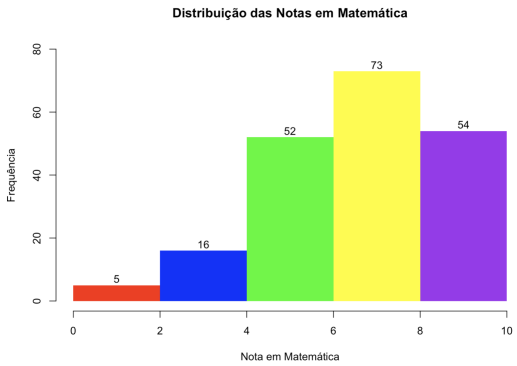
Por fim, é possível adicionar rótulos com os valores numéricos de cada barra do gráfico. Esses valores são independentes do gráfico, ou seja, são sobrepostos ao gráfico já gerado utilizando a função `text`. Para isso, utiliza-se o argumento `labels`, que recebe a tabela de referência com os valores a serem exibidos.

Para posicionar corretamente os valores, é necessário determinar as coordenadas `x` e `y`, onde os rótulos serão colocados. No caso de um gráfico de barras, o idela é que os valores fiquem centralizados sobre cada barra. Para isso, utiliza-se a mesma coordenada `x` de cada barra, e a coordenada `y` deve ser ligeiramente superior ao topo de cada barra, com um pequeno deslocamento para garantir que os rótulos não fiquem colados às barras.

Uma técnica importante para acessar os valores necessários para os rótulos é armazenar o gráfico em uma variável. Isso é feito para poder extrair as informações sobre os intervalos do eixo `x` (`mids`) e as contagens das barras (`counts`). No exemplo abaixo, o gráfico é armazenado na variável `h`, o que permite acessar esses valores de forma simples e clara.

No exemplo abaixo, o código aplica os ajustes ao gráfico já plotado, adicionando os valores de cada barra de forma centralizada e acima delas:

```
h <- hist(estudantes$Nota_em_Matematica,
          main = "Distribuição das
                Notas em Matemática",
          xlab = "Nota em Matemática",
          ylab = "Frequência",
          xlim = c(0, 10),
          ylim = c(0, 80),
          breaks = 5,
          col = c("red", "blue",
                "green", "yellow",
                "purple"),
          border = NA)

text(
  x = h$mids,
  y = h$counts + 2,
  labels = h$counts
)
```



Com isso, as principais funcionalidades da função `hist` para criar histogramas no R foram exploradas, incluindo personalizações como cores, intervalos, limites dos eixos e a adição de rótulos numéricos. Essas configurações permitem gerar visualizações mais informativas e bem ajustadas ao contexto dos dados analisados.

Na próxima seção, será apresentado o processo de criação de histogramas utilizando a biblioteca `ggplot2`, que oferece uma abordagem mais flexível e estilizada para visualizações de dados.

6.2 Histograma pelo ggplot2

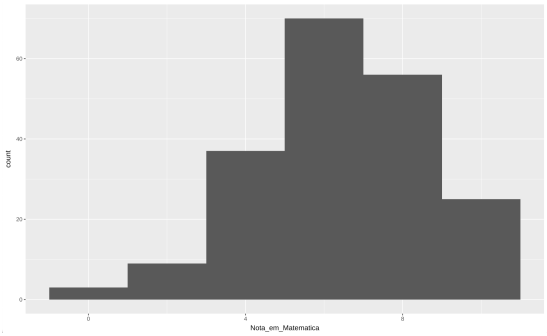
Como mencionado anteriormente no capítulo 4, os gráficos criados a partir do pacote `ggplot2` seguem uma estrutura em camadas, sendo sempre necessário especificar a base de dados utilizada, as variáveis a serem plotadas e o tipo de gráfico desejado.

Dando continuidade ao exemplo apresentado na seção anterior, será construído um histograma para a variável `Notas_em_Matematica`, da base de dados `estudantes`. Para isso, utiliza-se a função `geom_histogram`, que é responsável por gerar esse tipo de gráfico no `ggplot2`.

No caso do histograma, um fator importante é a definição dos intervalos (ou "bins") que compõem as barras. Esses intervalos podem ser ajustados por meio dos argumentos `binwidth` ou `bins`, dentro da função `geom_histogram`.

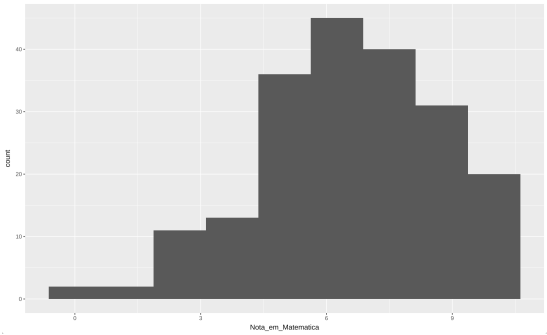
Ao utilizar o argumento `binwidth`, define-se a largura fixa de cada intervalo. Por exemplo, ao definir `binwidth = 2`, determina-se que cada barra do histograma representará um intervalo de 2 unidades.

```
ggplot(data = estudantes,
        aes(x = Nota_em_Matematica)) +
  geom_histogram(binwidth = 2)
```



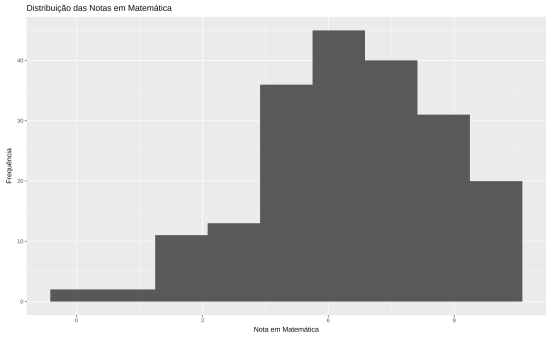
Em contrapartida, ao utilizar o argumento `bins`, define-se a quantidade de intervalos desejada para o histograma. Por exemplo, ao definir `bins = 9`, espera-se que o gráfico apresente 9 barras correspondentes a 9 intervalos. No entanto, vale destacar que a função `geom_histogram` pode ajustar automaticamente os limites dos intervalos para cobrir todo o conjunto de dados. Se as notas vão de 0 a 10 e é definido `bins = 9`, o `ggplot2` tentará criar 9 faixas com largura de aproximadamente 1,11. No entanto, o intervalo pode começar, por exemplo, em -0,05 e terminar em 10,05, a fim de garantir que todos os dados sejam incluídos nas barras.

```
ggplot(data = estudantes,
        aes(x = Nota_em_Matematica)) +
  geom_histogram(bins = 9)
```



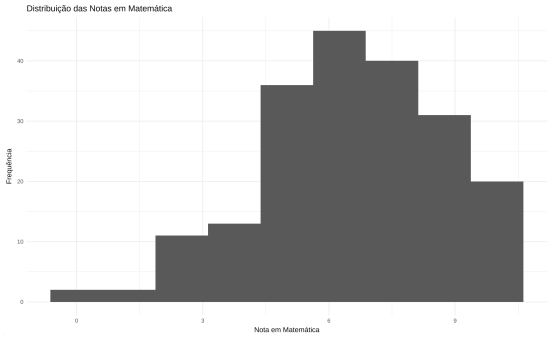
Para inserir os rótulos em um gráfico criado com o pacote `ggplot2`, utiliza-se a função `labs()`, que permite especificar o título principal do gráfico por meio do argumento `title`, o rótulo do eixo x com o argumento `x` e o rótulo do eixo y com o argumento `y`.

```
ggplot(data = estudantes,
       aes(x = Nota_em_Matematica)) +
  geom_histogram(bins = 9) +
  labs(title = "Distribuição das
           Notas em Matemática",
       x = "Nota em Matemática",
       y = "Frequência")
```



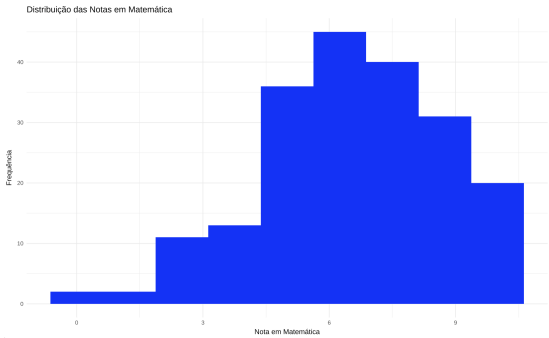
Para deixar o gráfico com um visual mais limpo e agradável, é possível utilizar a função `theme_minimal()`, que aplica um tema com grade (grid) mais discreta e elementos visuais simplificados.

```
ggplot(data = estudantes,
       aes(x = Nota_em_Matematica)) +
  geom_histogram(bins = 9) +
  labs(title = "Distribuição das
           Notas em Matemática",
       x = "Nota em Matemática",
       y = "Frequência") +
  theme_minimal()
```



Para definir uma única cor para todas as barras do gráfico, utiliza-se o argumento `fill` diretamente dentro da função `geom_histogram()`, especificando o nome da cor desejada em inglês. Caso queira explorar outras opções, é possível visualizar a lista completa de nomes de cores disponíveis no R utilizando o comando `colors()` no console.

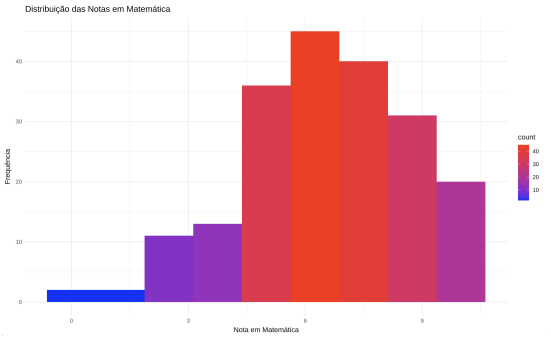
```
ggplot(data = estudantes,
       aes(x = Nota_em_Matematica)) +
  geom_histogram(bins = 9,
                 fill = "blue") +
  labs(title = "Distribuição das
           Notas em Matemática",
       x = "Nota em Matemática",
       y = "Frequência") +
  theme_minimal()
```



Outra opção interessante para cores no histograma é aplicar um gradiente de cores com base na frequência de cada intervalo. Para isso, utiliza-se a função `scale_fill_gradient()` juntamente com os argumentos `low`, que define a cor inicial do gradiente (para os valores mais baixos de frequência), e `high`, que define a cor final do gradiente (para os valores mais altos de frequência).

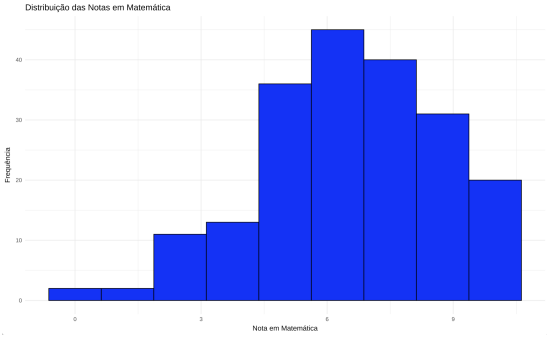
Para que o gradiente funcione corretamente, é necessário informar ao `ggplot2` que a cor de preenchimento das barras deve variar de acordo com a frequência de cada intervalo. Isso é feito com o comando `aes(fill = after_stat(count))` dentro da função `geom_histogram()`. A função `after_stat(count)` serve para usar o valor da frequência calculado automaticamente pelo gráfico como base para a cor. Sem essa linha, o gradiente não será aplicado corretamente.

```
ggplot(data = estudantes,
       aes(x = Nota_em_Matematica)) +
  geom_histogram(bins = 9,
       aes(fill =after_stat(count)))+
  scale_fill_gradient(low = "blue",
                      high = "red") +
  labs(title = "Distribuição das
            Notas em Matemática",
       x = "Nota em Matemática",
       y = "Frequência") +
  theme_minimal()
```



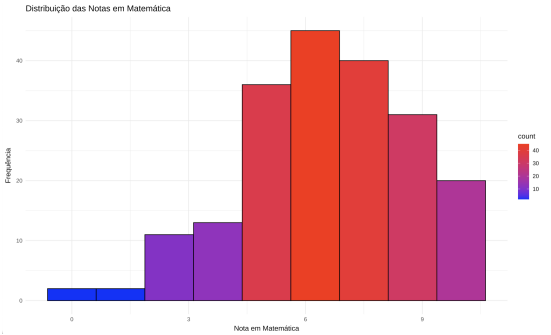
Outro elemento gráfico são as bordas das barras, que podem ser personalizadas com o argumento `color` dentro da função `geom_histogram()`. Esse argumento define a cor do contorno de cada barra. Por exemplo, para adicionar bordas pretas, utiliza-se `color = "black"`. Caso se deseje remover completamente as bordas, basta atribuir o valor `color = NA`, indicando que nenhuma cor deve ser aplicada ao contorno.

```
ggplot(data = estudantes,
       aes(x = Nota_em_Matematica)) +
  geom_histogram(bins = 9,
                 fill = "blue",
                 color = "black") +
  labs(title = "Distribuição das
            Notas em Matemática",
       x = "Nota em Matemática",
       y = "Frequência") +
  theme_minimal()
```



Esse mesmo ajuste pode ser feito em histogramas com preenchimento em gradiente, bastando adicionar o argumento `color` dentro da função `geom_histogram()`, assim como no exemplo anterior.

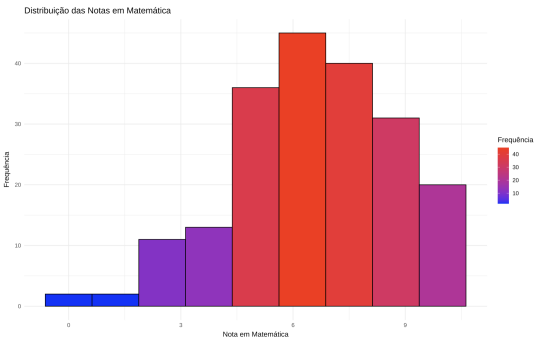
```
ggplot(data = estudantes,
       aes(x = Nota_em_Matematica)) +
  geom_histogram(bins = 9,
       aes(fill =after_stat(count)),
       color = "black") +
  scale_fill_gradient(low = "blue",
                      high = "red")+
  labs(title = "Distribuição das
           Notas em Matemática",
       x = "Nota em Matemática",
       y = "Frequência") +
  theme_minimal()
```



O `ggplot` adiciona automaticamente legendas quando necessário, mas muitas vezes é interessante personalizá-las. No caso do histograma, a legenda é gerada automaticamente para o preenchimento das cores em gradiente.

Para modificar o título exibido na legenda, utiliza-se o argumento `name` dentro da função `scale_fill_gradient()`. Quando o preenchimento é mapeado com `after_stat(count)`, o título padrão da legenda será `count`. Com o argumento `name`, é possível substituir esse título por um rótulo mais descritivo e informativo, facilitando a interpretação do gráfico.

```
ggplot(data = estudantes,
       aes(x = Nota_em_Matematica)) +
  geom_histogram(bins = 9,
       aes(fill =after_stat(count)),
                   color = "black") +
  scale_fill_gradient(low = "blue",
                high = "red",
                name = "Frequência")+
  labs(title = "Distribuição das
           Notas em Matemática",
       x = "Nota em Matemática",
       y = "Frequência") +
  theme_minimal()
```



Com isso, encerra-se a apresentação das principais funcionalidades do pacote `ggplot2` aplicadas à construção de histogramas no R. Ao longo das seções, foram abordados desde os elementos básicos da criação desses gráficos até personalizações

mais avançadas, como ajustes de intervalos, cores, rótulos e legendas. Essas ferramentas permitem que o usuário desenvolva visualizações mais claras, organizadas e alinhadas ao contexto dos dados, facilitando a interpretação e a comunicação dos resultados.

No próximo Capítulo, será abordada a construção de gráficos de setores. Esses gráficos são particularmente úteis para representar a distribuição proporcional de variáveis qualitativas (categóricas), onde cada "setor"representa a proporção de uma categoria em relação ao todo. Inicialmente, serão utilizadas as funções nativas do R para esse tipo de visualização. Em seguida, será apresentada a construção de histogramas com o pacote `ggplot2`, que são mais adequados para representar a distribuição de variáveis quantitativas.

7 GRÁFICO DE SETORES

O gráfico de setores é uma das representações gráficas mais utilizadas no ensino de estatística e análise de dados. Ele permite visualizar a proporção de cada categoria em relação ao total, de forma intuitiva e visualmente atrativa. Em sala de aula, esse tipo de gráfico é amplamente empregado para apresentar a composição percentual de respostas em pesquisas, distribuição de alunos por faixa etária, preferências em questionários, entre outros conjuntos de dados categóricos.

Sua principal vantagem é a facilidade de interpretação, especialmente quando queremos destacar a relação de cada parte em relação ao todo, tornando-se uma ferramenta didática importante para introduzir conceitos de proporção e percentual, além de incentivar a análise crítica dos dados pelos alunos. O gráfico de setores é útil quando tratamos de poucas categorias.

Inicialmente, será demonstrado como plotar o gráfico de setores utilizando funções nativas do R e, posteriormente, utilizando funções da biblioteca `ggplot2`. Serão apresentadas as linhas de código e, ao lado, o gráfico gerado por essas linhas.

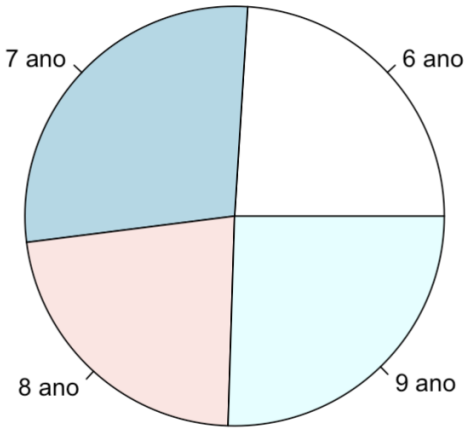
As Figuras apresentadas neste capítulo são de autoria própria, geradas a partir da execução dos códigos apresentados ao longo do texto. Assim, cada imagem corresponde diretamente à saída gráfica produzida pelos scripts indicados na seção.

7.1 Gráfico de Setores por funções nativas

A função nativa do R para a criação de gráficos de barras é `pie`. Para consultar os parâmetros disponíveis para essa função, utiliza-se o comando `?pie`, que exibirá a documentação correspondente em inglês.

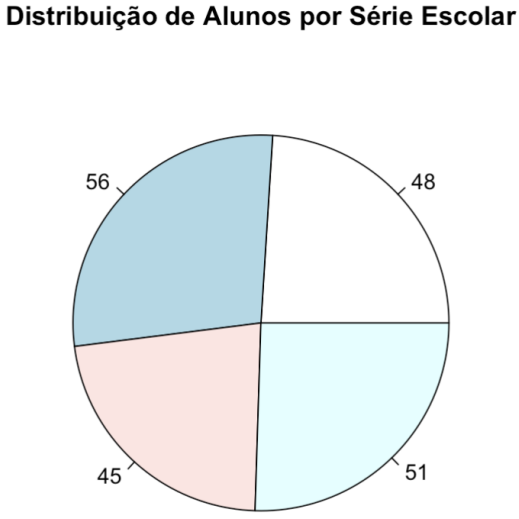
Para gerar um gráfico de setores, isso como nos casos anteriores, primeiro é necessário escolher a variável da base de dados que será representada graficamente. Neste exemplo, utiliza-se a variável `Serie` da base de dados `estudantes`. Para isso, é necessário calcular a tabela de frequências dessa variável e, em seguida, aplicar a função `pie`, conforme ilustrado no exemplo abaixo:

```
pie(table(estudantes$Serie))
```



Para inserir os rótulos do gráfico utiliza-se os seguintes argumentos: `main` que insere o título principal do gráfico e `labels` que define o rótulo de cada setor, ou seja, a informação exibida diretamente em cada fatia. Neste exemplo, os rótulos mostram a frequência absoluta de cada categoria, referenciando diretamente a tabela de frequências.

```
pie(
  table(estudantes$Serie),
  main = "Distribuição de Alunos
        por Série Escolar",
  labels = table(estudantes$Serie))
```



No entanto, é bastante comum, principalmente em análises exploratórias e apresentações de resultados, que os gráficos de setores apresentem as categorias em percentuais, ao invés de valores absolutos. Para isso, é necessário calcular esses percentuais e utilizá-los como rótulos.

A construção dos rótulos percentuais pode ser feita diretamente dentro do argumento `labels`, utilizando a função `paste` combinada com `round`, conforme exemplo abaixo:

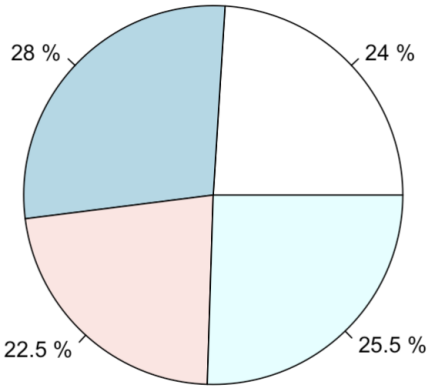
```
labels = paste(round(100 * table(estudantes$Serie)/sum(table(estudantes$Serie)), 1), "%"))
```

Aqui, temos:

- `table(estudantes$Serie)`: tabela de frequências com a contagem de cada série.
- `sum(table(estudantes$Serie))`: total geral, ou seja, número total de estudantes.
- `100 * ...`: cálculo da porcentagem.
- `round(..., 1)`: arredonda os valores para uma casa decimal.
- `paste(...)`: concatena os números percentuais com o símbolo "%", para que o rótulo fique completo.

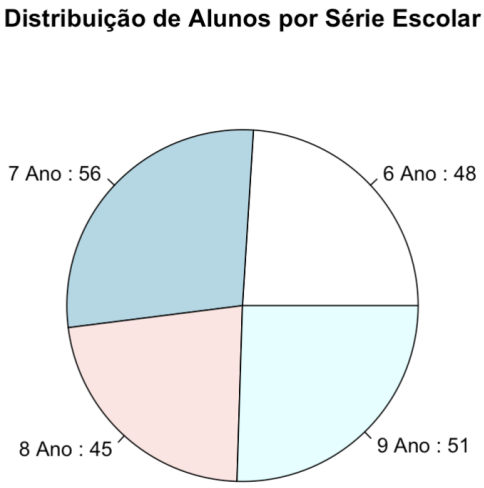
Assim, o código do gráfico fica conforme abaixo:

```
pie(
  table(estudantes$Serie),
  main = "Distribuição de Alunos
        por Série Escolar",
  labels = paste(round(
    100 * table(estudantes$Serie) /
    sum(table(estudantes$Serie)),
    1), "%"))
```



Para melhor visualização, como não é utilizada uma legenda, é útil incluir no rótulo tanto o nome da categoria quanto o valor que ela representa. No exemplo abaixo, é utilizado o valor absoluto de cada categoria, mas esse rótulo pode ser facilmente adaptado para exibir o valor percentual, conforme o código apresentado anteriormente.

```
pie(
  table(estudantes$Serie),
  main = "Distribuição de Alunos
        por Série Escolar",
  labels = paste(c("6 Ano",
        "7 Ano", "8 Ano", "9 Ano"),
        ":", table(estudantes$Serie)))
```

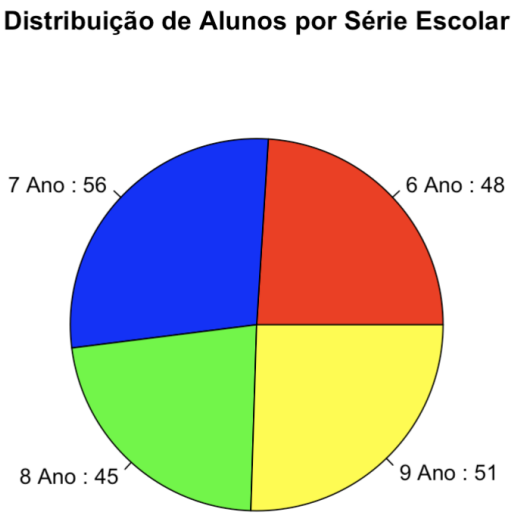


Para inserir cor aos setores, utiliza-se o argumento `col`. No geral, o gráfico de setores não faz sentido com uma cor única para todas as categorias, a não ser que o objetivo seja, visualmente, agrupar categorias.

Para definir várias cores no gráfico, atribuindo uma cor diferente para cada categoria, defini-se um vetor com todas as cores desejadas. Para isso, emprega-se a função `c("cor_1", "cor_2", ...)`, onde cada elemento representa uma cor, com os nomes das cores escritos em inglês.

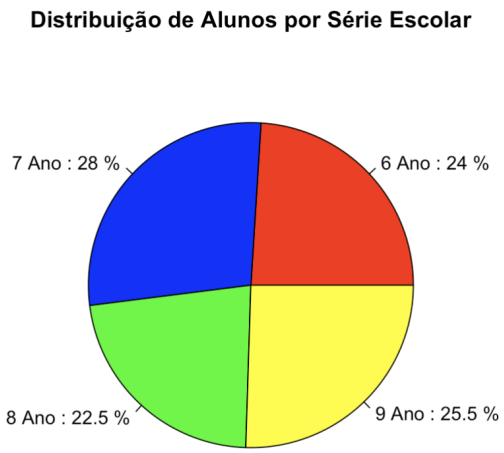
A cor correspondente a `cor_1` será aplicada à primeira categoria, `cor_2` à segunda categoria e assim sucessivamente. É possível, inclusive, repetir cores dentro do vetor, caso isso faça sentido para a análise que está sendo apresentada.

```
pie(
  table(estudantes$Serie),
  main = "Distribuição de Alunos
        por Série Escolar",
  labels = paste(c("6 Ano",
        "7 Ano", "8 Ano", "9 Ano"),
        ":", table(estudantes$Serie)),
  col = c("red", "blue", "green",
        "yellow"))
```



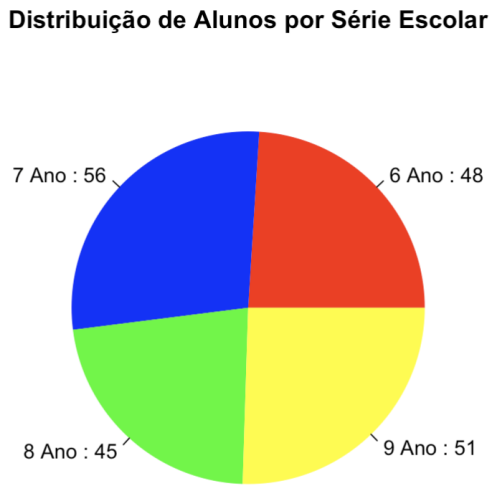
O mesmo pode ser aplicado ao gráfico com os rótulos em porcentagem, como a figura abaixo exemplifica.

```
pie(
  table(estudantes$Serie),
  main = "Distribuição de Alunos
        por Série Escolar",
  labels = paste(c("6 Ano",
        "7 Ano", "8 Ano", "9 Ano"),
        ":", round(100 *
        table(estudantes$Serie)/
        sum(table(estudantes$Serie)),
        1), "%"),
  col = c("red", "blue", "green",
        "yellow"))
```



Outro elemento gráfico interessante são as bordas dos setores. Pode-se ajustar a espessura das bordas, modificar seu estilo ou até mesmo removê-las, o que é bastante comum. Para isso, utiliza-se o argumento `border`. Caso deseje remover as bordas, basta atribuir o valor `NA`, indicando ausência de borda. O exemplo abaixo contém os rótulos com valores numéricos, podendo ser adaptados para os valores percentuais, como visto acima.

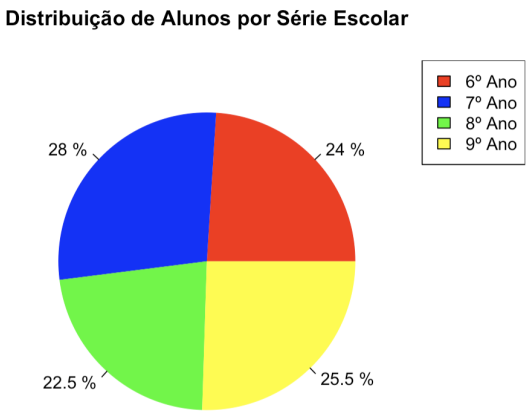
```
pie(
  table(estudantes$Serie),
  main = "Distribuição de Alunos
        por Série Escolar",
  labels = paste(c("6 Ano",
        "7 Ano", "8 Ano", "9 Ano"),
        ":", table(estudantes$Serie)),
  col = c("red", "blue", "green",
        "yellow"),
  border = NA)
```



Ao invés de mostrar o nome da categoria diretamente no gráfico, pode-se inserir uma legenda, utilizando o argumento `legend`, cuja estrutura é: `legend(posição, nomes_categorias, cores_correspondentes)`.

É importante destacar que a legenda é sobreposta ao gráfico já existente. Assim, o procedimento correto é primeiro gerar o gráfico `(pie)` e, em seguida, adicionar a legenda `(legend)`.

```
pie(
  table(estudantes$Serie),
  main = "Distribuição de Alunos
        por Série Escolar",
 labels = paste(c("6 Ano",
        "7 Ano", "8 Ano", "9 Ano"),
        ":", round(100 *
        table(estudantes$Serie)/
        sum(table(estudantes$Serie)),
        1), "%"),
  col = c("red", "blue", "green",
        "yellow")),
  border = NA)

legend(
  "topright",
  legend = c("6º Ano", "7º Ano",
                "8º Ano", "9º Ano"),
  fill = c("red", "blue",
        "green", "yellow")
)
```



Com isso, as principais funcionalidades da função `pie` para criar gráficos de setores no R foram exploradas, incluindo personalizações como cores, bordas, legendas e a adição de rótulos numéricos, tanto no valor absoluto como percentual. Essas configurações permitem gerar visualizações mais informativas e bem ajustadas ao contexto dos dados analisados.

Na próxima seção, será apresentado o processo de criação de gráficos de setores utilizando a biblioteca `ggplot2`, que oferece uma abordagem mais flexível e estilizada para visualizações de dados.

7.2 Gráfico de Setores pelo ggplot2

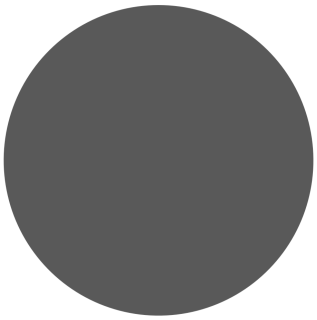
Para gerar um gráfico de setores com o `ggplot2`, é necessário seguir alguns passos simples. Primeiro, deve-se escolher a variável que será representada graficamente. Assim como apresentado na seção anterior, o gráfico será da variável `Serie`, da base de dados `estudantes`.

O gráfico de setores é construído a partir do gráfico de barras, mas utilizando coordenadas polares para obter um gráfico circular. Para isso, utiliza-se a função `geom_bar()`, com o argumento `width = 1` para que as barras ocupem toda a largura disponível. Em seguida, aplica-se a função `coord_polar()`, com o argumento `theta = "y"`, para que o eixo Y seja a referência para os ângulos dos setores.

Além disso, o código começa com a função `ggplot(data = estudantes, aes(x = ¨¨))`, que define a base do gráfico. O `x = ¨¨` indica que não há um eixo X a ser representado, uma vez que o gráfico de pizza não precisa dessa dimensão. O preenchimento das barras/setores é feito pela variável `Serie`, alocada no argumento `fill`.

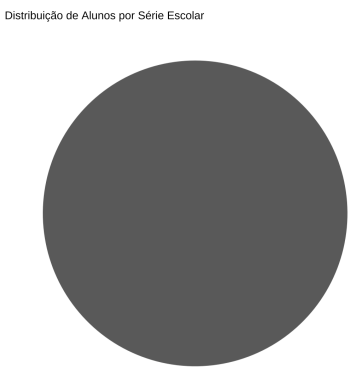
Por fim, a função `theme_void()` é utilizada para remover elementos desnecessários, como eixos e grades, deixando o gráfico limpo, com apenas os setores e as legendas, o que é característico de um gráfico de pizza.

```
ggplot(data = estudantes,
        aes(x = ""),
        fill = factor(Serie))+
   geom_bar(width = 1) +
   coord_polar(theta = "y") +
   theme_void()
```



Para inserir os rótulos em um gráfico criado com o pacote `ggplot2`, utiliza-se a função `labs()`, que permite especificar o título principal do gráfico por meio do argumento `title`, o rótulo do eixo x com o argumento `x` e o rótulo do eixo y com o argumento `y`. No caso de gráficos de setores, não faz sentido atribuir rótulos aos eixos x e y, uma vez que esses eixos não são utilizados na interpretação do gráfico. Por isso, atribui-se o valor `NULL` a esses argumentos.

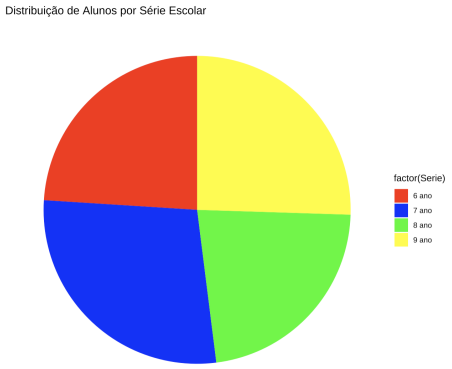
```
ggplot(data = estudantes,
        aes(x = ""),
        fill = factor(Serie)) +
  geom_bar(width = 1) +
  coord_polar(theta = "y") +
  labs(title = "Distribuição de
        Alunos por Série Escolar",
        x = NULL,
        y = NULL) +
  theme_void()
```



Para inserir cor aos setores, utiliza-se a função `scale_fill_manual` juntamente com o argumento `values`, onde é informado um vetor com as cores desejadas. No geral, o gráfico de setores não faz sentido com uma cor única para todas as categorias, a não ser que o objetivo seja, visualmente, agrupar categorias.

No vetor passado ao argumento `values`, foram especificadas quais categorias receberão quais cores. As categorias que não forem explicitamente mencionadas manterão a cor padrão (geralmente um tom de cinza).

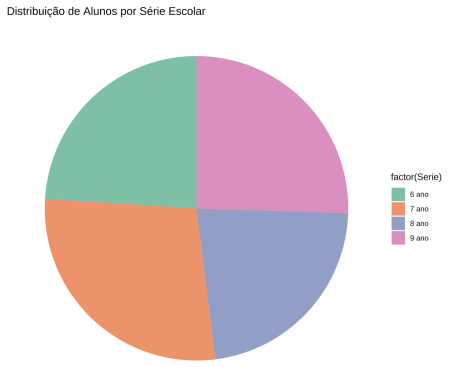
```
ggplot(data = estudantes,
       aes(x = ""),
       fill = factor(Serie)) +
  geom_bar(width = 1) +
  coord_polar(theta = "y") +
  scale_fill_manual(values =
                c("6 ano" = "red",
                "7 ano" = "blue",
                "8 ano" = "green",
                "9 ano" = "yellow")) +
  labs(title = "Distribuição de
       Alunos por Série Escolar",
       x = NULL,
       y = NULL) +
  theme_void()
```



A função `scale_fill_brewer()`, juntamente com o argumento `palette`, permite aplicar paletas de cores pré-definidas disponíveis no pacote `RColorBrewer`. Essas paletas são especialmente úteis para garantir uma boa harmonia visual e acessibilidade nos gráficos.

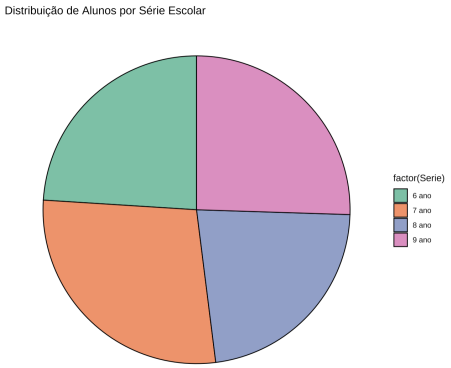
Para utilizá-la, é necessário instalar e carregar o pacote com o comando `library(RColorBrewer)` no início do código. Para visualizar todas as paletas disponíveis e obter mais informações, digite `?RColorBrewer` no console do R.

```
ggplot(data = estudantes,
       aes(x = ""),
       fill = factor(Serie)) +
  geom_bar(width = 1) +
  coord_polar(theta = "y") +
  scale_fill_brewer(
       palette = "Set2")+
  labs(title = "Distribuição de
       Alunos por Série Escolar",
       x = NULL,
       y = NULL) +
  theme_void()
```



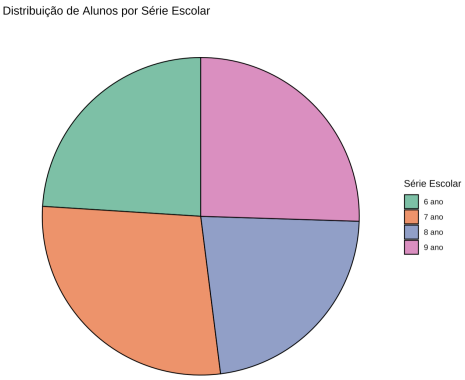
Outro elemento gráfico interessante são as bordas dos setores. Pode-se ajustar a espessura das bordas, modificar seu estilo ou até mesmo removê-las, o que é bastante comum. Para isso, utiliza-se o argumento `border`. Caso deseje remover as bordas, basta atribuir o valor `NA`, indicando ausência de borda. O exemplo abaixo contém os rótulos com valores numéricos, podendo ser adaptados para os valores percentuais, como visto acima.

```
ggplot(data = estudantes,
        aes(x = ""),
        fill = factor(Serie)) +
  geom_bar(width = 1,
           color = "black") +
  coord_polar(theta = "y") +
  scale_fill_brewer(
        palette = "Set2")+
  labs(title = "Distribuição de
        Alunos por Série Escolar",
        x = NULL,
        y = NULL) +
  theme_void()
```



Para modificar o título exibido na legenda, utiliza-se o argumento `fill` dentro da função `labs()`. Assim, é possível substituir o nome padrão — que normalmente corresponde ao nome da variável utilizada no `fill` — por um título mais descritivo.

```
ggplot(data = estudantes,
        aes(x = ""),
        fill = factor(Serie)) +
  geom_bar(width = 1,
           color = "black") +
  coord_polar(theta = "y") +
  scale_fill_brewer(
        palette = "Set2")+
  labs(title = "Distribuição de
        Alunos por Série Escolar",
        x = NULL,
        y = NULL,
        fill = "Série Escolar") +
  theme_void()
```



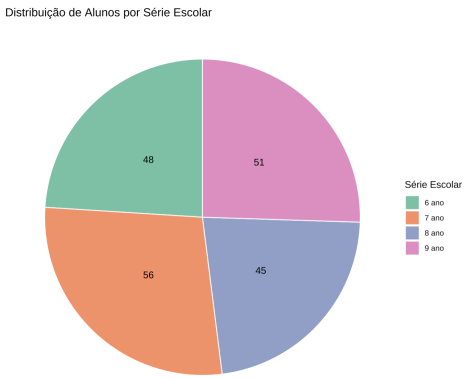
Por fim, é possível adicionar rótulos com os valores numéricos de setor do gráfico, utilizando a função `geom_text` com o argumento `label`, que recebe os valores a serem exibidos.

Utiliza-se o argumento `aes(label = after_stat(count))` para definir os rótulos que serão exibidos nos setores. O `aes()` é utilizado para mapear variáveis dos dados para elementos do gráfico. Neste caso, serão mapeados os valores de label para exibir as contagens de cada categoria. A função `after_stat(count)` é responsável por calcular a contagem das observações de cada categoria (por exemplo, a quantidade de alunos em cada faixa etária). Assim, serão adicionados os valores dessas contagens como rótulos em cada barra do gráfico.

Além disso, o argumento `stat = "count"` é usado para garantir que o `geom_text()` calcule a contagem de ocorrências de cada categoria, pois por padrão o `geom_text()` tentaria usar os valores do eixo y, o que não seria adequado nesse caso. O `stat = "count"` instrui o R a contar as ocorrências nas barras para gerar os rótulos de forma adequada. O argumento `color = "black"` define a cor do rótulo como preto.

Para ajustar a posição dos rótulos em um gráfico de setores, utiliza-se o argumento `position = position_stack(vjust = 0.5)` dentro da função `geom_text()`. Esse argumento permite que os rótulos sejam posicionados no centro de cada setor, utilizando a lógica de empilhamento da variável categórica. O valor `vjust = 0.5` garante que o texto fique centralizado verticalmente dentro de cada fatia do gráfico, melhorando a legibilidade e a estética da visualização.

```
ggplot(data = estudantes,
       aes(x = ""),
       fill = factor(Serie)) +
  geom_bar(width = 1,
           color = "black") +
  coord_polar(theta = "y") +
  scale_fill_brewer(
           palette = "Set2")+
  geom_text(
    aes(label = after_stat(count)),
    stat = "count",
    color = "black",
    position = position_stack(
                vjust = 0.5)) +
  labs(title = "Distribuição de
       Alunos por Série Escolar",
       x = NULL,
       y = NULL,
       fill = "Série Escolar") +
  theme_void()
```



No entanto, é bastante comum, principalmente em análises exploratórias e apresentações de resultados, que os gráficos de setores apresentem as categorias em percentuais, ao invés de valores absolutos. Para isso, é necessário calcular esses percentuais e utilizá-los como rótulos.

A construção dos rótulos percentuais pode ser feita diretamente dentro do argumento `labels`, utilizando a função `paste` combinada com `round`, conforme exemplo abaixo:

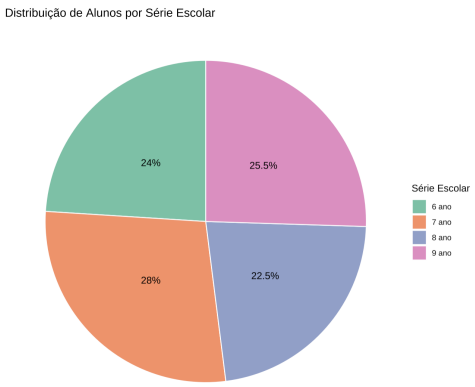
```
labels = paste0(round(...count.../sum(...count...)*100), 1), "%"))
```

Aqui, temos:

- `...count...`: contagem de cada série.
- `sum(...count..)`: total geral, ou seja, número total de estudantes.
- `...*100`: cálculo da porcentagem.
- `round(..., 1)`: arredonda os valores para uma casa decimal.
- `paste(...)`: concatena os números percentuais com o símbolo "%", para que o rótulo fique completo, sem espaço entre os símbolos.

Assim, o código do gráfico fica conforme abaixo:

```
ggplot(data = estudantes,
       aes(x = ""),
       fill = factor(Serie)) +
  geom_bar(width = 1,
           color = "black") +
  coord_polar(theta = "y") +
  scale_fill_brewer(
           palette = "Set2")+
  geom_text(
    aes(label = paste0(
        round(..count.. /
        sum(..count..)* 100, 1),
        "%")),
    stat = "count",
    color = "black",
    position = position_stack(
                vjust = 0.5) +
  labs(title = "Distribuição de
        Alunos por Série Escolar",
        x = NULL,
        y = NULL,
        fill = "Série Escolar") +
  theme_void()
```



Com isso, encerra-se a apresentação das principais funcionalidades do pacote `ggplot2` aplicadas à construção de gráfico de setores no R. Ao longo das seções, foram abordados desde os elementos básicos da criação desses gráficos até personalizações mais avançadas, como cores, rótulos e legendas. Essas ferramentas permitem que o usuário desenvolva visualizações mais claras, organizadas e alinhadas ao contexto dos dados, facilitando a interpretação e a comunicação dos resultados.

No próximo Capítulo, será abordada a construção de gráficos de linhas. Esses gráficos são particularmente úteis para representar a evolução de uma variável ao longo do tempo ou de algum outro eixo contínuo. São indicados principalmente para variáveis quantitativas, permitindo visualizar tendências, oscilações e padrões ao longo

de um intervalo. Inicialmente, serão utilizadas as funções nativas do R para esse tipo de visualização. Em seguida, será apresentada a construção de gráficos de linhas com o pacote `ggplot2`, que oferece maior flexibilidade e qualidade gráfica para análises mais sofisticadas.

8 GRÁFICO DE LINHAS

O gráfico de linhas é uma representação gráfica que permite visualizar e analisar a evolução de uma variável ao longo de um intervalo, sendo especialmente útil para dados temporais ou sequenciais. Em sala de aula, esse tipo de gráfico pode ser utilizado para analisar a evolução das notas dos alunos ao longo do tempo, estudar tendências em fenômenos naturais e explorar outros conjuntos de dados sequenciais.

Sua principal vantagem é a clareza na visualização de tendências e padrões ao longo do tempo, permitindo identificar rapidamente variações, ciclos e comportamentos dos dados. Além disso, o gráfico de linhas facilita a compreensão da relação entre duas variáveis, mostrando como uma pode influenciar ou depender da outra ao longo do período analisado.

Inicialmente, será demonstrado como plotar o gráfico de linhas utilizando funções nativas do R e, posteriormente, utilizando funções da biblioteca `ggplot2`. Serão apresentadas as linhas de código e, ao lado, o gráfico gerado por essas linhas.

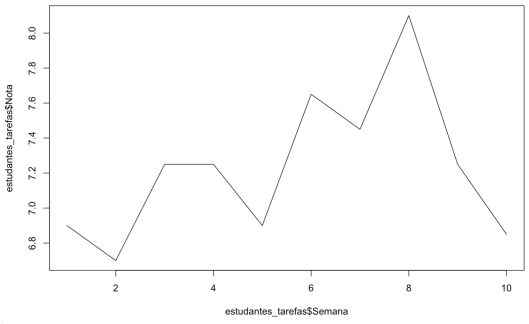
As Figuras apresentadas neste capítulo são de autoria própria, geradas a partir da execução dos códigos apresentados ao longo do texto. Assim, cada imagem corresponde diretamente à saída gráfica produzida pelos scripts indicados na seção.

8.1 Gráfico de linhas por funções nativas

A função nativa do R para a criação de gráficos de linhas é `plot`. Para consultar os parâmetros disponíveis para essa função, utiliza-se o comando `?plot`, que exibirá a documentação correspondente em inglês. A função `plot` pode ser usada para diversos gráficos, como pontos (`"p"`), linhas (`"l"`), pontos e linhas (`"b"`), degraus (`"s"` e `"S"`), ou histograma com linhas verticais (`"h"`), entre outros.

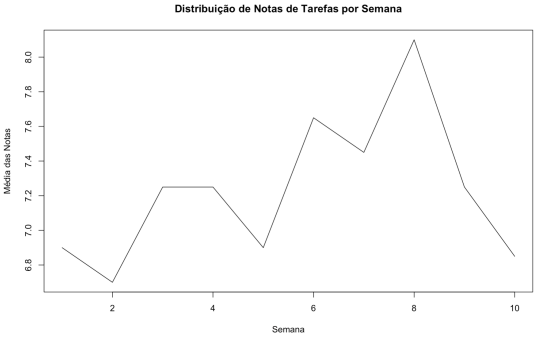
O foco deste capítulo são os gráficos de linhas, então será utilizada a função `plot` com o argumento `type = "l"`. Para gerar o gráfico de linhas, primeiro determina-se a variável da base de dados para o eixo `x`, e em seguida, a variável para o eixo `y`, além do tipo de gráfico conforme mencionado. Será usada a base de dados `estudantes_tarefas`, que apresenta as semanas e a média da turma nas notas de tarefas de Matemática. No exemplo a seguir, utiliza-se a variável `Semana` para o eixo `x` e a variável `Nota` para o eixo `y`:

```
plot(estudantes_tarefas$Semana,
    estudantes_tarefas$Nota,
    type = "l")
```



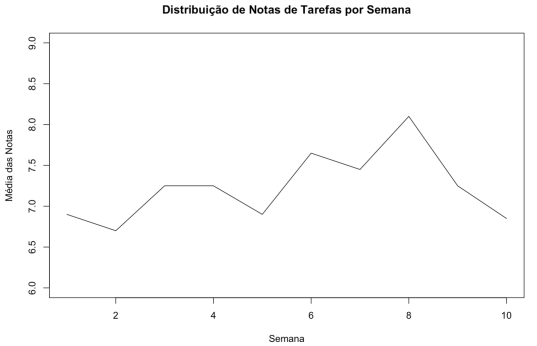
Perceba que, automaticamente, a função insere os rótulos dos eixos, baseado na base de dados utilizada. Para personalizar os rótulos do gráfico utiliza-se os seguintes argumentos: `main` que insere o título principal do gráfico, `xlab` para inserir o rótulo do eixo x e `ylab` para inserir o rótulo do eixo y.

```
plot(estudantes_tarefas$Semana,
     estudantes_tarefas$Nota,
     type = "l",
     main = "Distribuição de Notas
            de Tarefas por Semana",
     xlab = "Semana",
     ylab = "Média das Notas")
```



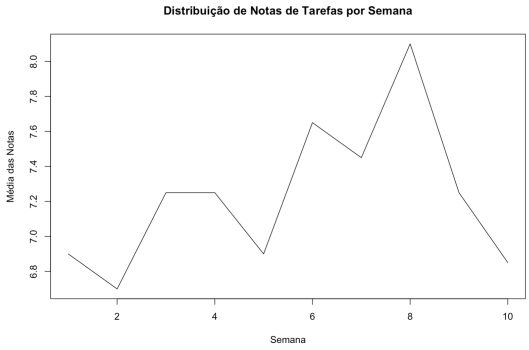
A função `plot` automaticamente define os limites numéricos dos eixos `x` e `y`, mas é possível alterar esses limites utilizando os argumentos `xlim` e `ylim`, passando os valores no formato `(valor_inicio, valor_final)`. No exemplo abaixo, o limite de `x` foi ajustado para `(1,10)`, ou seja, o valor inicial do eixo `x` é 1 e o valor máximo do eixo `x` é 10. De forma análoga, o limite de `y` foi alterado para `(6, 9)`,conforme os valores da distribuição.

```
plot(estudantes_tarefas$Semana,
     estudantes_tarefas$Nota,
     type = "l",
     main = "Distribuição de Notas
            de Tarefas por Semana",
     xlab = "Semana",
     ylab = "Média das Notas",
     xlim = c(1, 10),
     ylim = c(6, 9))
```



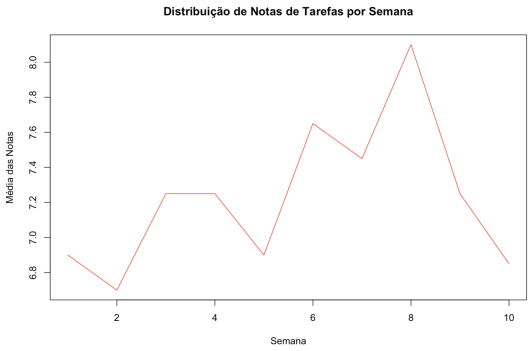
Também é possível modificar esses limites conforme os valores das variáveis. Por exemplo, no caso abaixo, define-se que o limite inferior do eixo `y` é o valor mínimo das notas e o limite superior é o valor máximo das notas, utilizando as funções `min` e `max`, respectivamente.

```
plot(
    estudantes_tarefas$Semana,
    estudantes_tarefas$Nota,
    type = "l",
    main = "Distribuição de Notas
            de Tarefas por Semana",
    xlab = "Semana",
    ylab = "Média das Notas",
    xlim = c(1,10),
    ylim =
        c(min(estudantes_tarefas$Nota),
        max(estudantes_tarefas$Nota)))
```



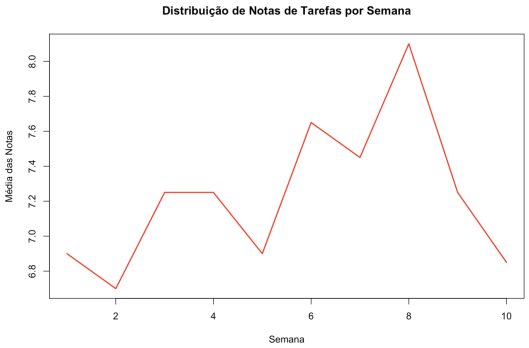
Para inserir cor à linha, utiliza-se o argumento `col`. Para definir a cor da linha, basta especificá-la em inglês. Além disso, é possível visualizar a lista de cores disponíveis no R utilizando o comando `colors()` no console.

```
plot(
    estudantes_tarefas$Semana,
    estudantes_tarefas$Nota,
    type = "l",
    main = "Distribuição de Notas
            de Tarefas por Semana",
    xlab = "Semana",
    ylab = "Média das Notas",
    xlim = c(1,10),
    ylim =
        c(min(estudantes_tarefas$Nota),
        max(estudantes_tarefas$Nota)),
    col = "red")
```



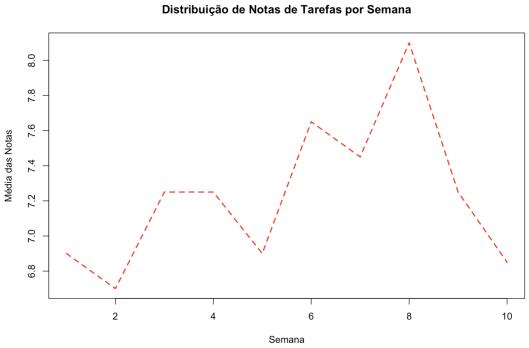
Com o argumento `lwd`, pode-se alterar a espessura da linha no gráfico, o que é útil para destacar ou suavizar a aparência da linha, dependendo do objetivo visual. O valor de `lwd` é um número que define a espessura da linha, sendo 1 o valor padrão. Valores maiores que 1 tornam a linha mais espessa, enquanto valores menores que 1 tornam a linha mais fina.

```
plot(
    estudantes_tarefas$Semana,
    estudantes_tarefas$Nota,
    type = "l",
    main = "Distribuição de Notas
            de Tarefas por Semana",
    xlab = "Semana",
    ylab = "Média das Notas",
    xlim = c(1,10),
    ylim =
        c(min(estudantes_tarefas$Nota),
        max(estudantes_tarefas$Nota)),
    col = "red",
    lwd = 2)
```



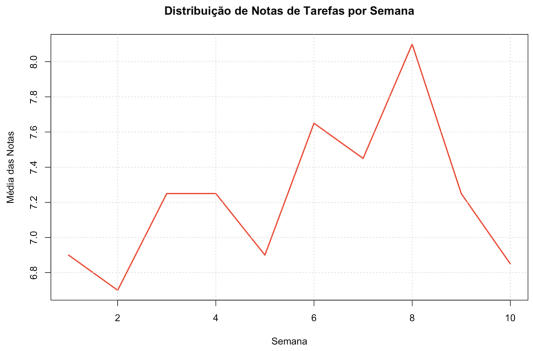
Além disso, pode-se customizar a linha com o argumento `lty`, que define o tipo de linha no gráfico. O valor padrão (`lty = 1`) gera uma linha sólida, mas é possível escolher entre diversas opções, como linhas tracejadas (`lty = 2`), pontilhadas (`lty = 3`) e outras variações.

```
plot(
    estudantes_tarefas$Semana,
    estudantes_tarefas$Nota,
    type = "l",
    main = "Distribuição de Notas
            de Tarefas por Semana",
    xlab = "Semana",
    ylab = "Média das Notas",
    xlim = c(1,10),
    ylim =
        c(min(estudantes_tarefas$Nota),
        max(estudantes_tarefas$Nota)),
    col = "red",
    lwd = 2,
    lty = 2)
```



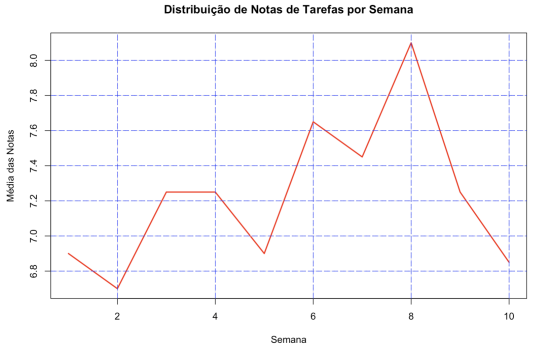
Outro elemento que pode ser útil no gráfico de linhas é a presença do grid de orientação. O `grid` é um conjunto de linhas horizontais e verticais que ajudam a visualizar a escala dos eixos e a localização exata dos pontos no gráfico. Ele é sobreposto ao gráfico, ou seja, deve ser adicionado após a plotagem do gráfico.

```
plot(
    estudantes_tarefas$Semana,
    estudantes_tarefas$Nota,
    type = "l",
    main = "Distribuição de Notas
            de Tarefas por Semana",
    xlab = "Semana",
    ylab = "Média das Notas",
    xlim = c(1,10),
    ylim =
        c(min(estudantes_tarefas$Nota),
        max(estudantes_tarefas$Nota)),
    col = "red",
    lwd = 2,
    lty = 1)

grid()
```



Assim como as linhas do gráfico, o grid também pode ser personalizado com cor, espessura e tipo de linha. No exemplo abaixo, o grid foi personalizado com a cor azul, espessura de 0.8 e tipo de linha = 5, ou seja, uma linha pontilhada. A personalização dessas características é feita diretamente nos argumentos da função `grid()`.

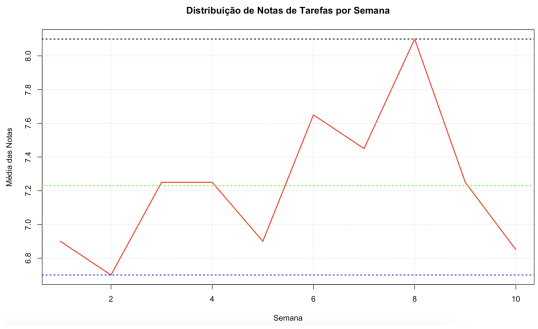
```
plot(
    estudantes_tarefas$Semana,
    estudantes_tarefas$Nota,
    type = "l",
    main = "Distribuição de Notas
            de Tarefas por Semana",
    xlab = "Semana",
    ylab = "Média das Notas",
    xlim = c(1,10),
    ylim =
        c(min(estudantes_tarefas$Nota),
        max(estudantes_tarefas$Nota)),
    col = "red",
    lwd = 2,
    lty = 1)

grid(col = "blue", lwd = 0.8, lty = 5)
```



Por fim, é possível adicionar linhas de referência no gráfico, como as linhas de mínimo, máximo e média. Para isso, utiliza-se a função `abline`, que determina a altura `h` das linhas a serem inseridas no gráfico. Essas linhas podem ser personalizadas, como as linhas do próprio gráfico, alterando suas cores, espessura e tipo de linha. Para inserir as linhas de mínimo, usa-se a função `min`, para a média a função `mean`, e para o máximo utiliza-se a função `max`.

No exemplo abaixo, as linhas foram diferenciadas por cores: a linha do mínimo foi plotada em azul, a linha da média em verde e a linha do máximo em preto. O uso dessas linhas de referência ajuda a destacar informações importantes no gráfico e facilita a visualização de dados relevantes.

```
plot(
    estudantes_tarefas$Semana,
    estudantes_tarefas$Nota,
    type = "l",
    main = "Distribuição de Notas
            de Tarefas por Semana",
    xlab = "Semana",
    ylab = "Média das Notas",
    xlim = c(1,10),
    ylim =
        c(min(estudantes_tarefas$Nota),
        max(estudantes_tarefas$Nota)),
    col = "red",
    lwd = 2,
    lty = 1)

grid()

abline(h=min(estudantes_tarefas$Nota),
    col = "blue", lty = 3, lwd = 2)
abline(h=mean(estudantes_tarefas$Nota),
    col = "green", lty = 3, lwd = 2)
abline(h =
↪max(estudantes_tarefas$Nota),
    col = "black", lty = 3, lwd = 2)
```



Com isso, as principais funcionalidades da função `plot` para criar gráfico de linhas no R foram exploradas, incluindo personalizações como cores, espessuras, tipos de linha, limites dos eixos e a adição de linhas de referência. Essas configurações permitem gerar visualizações mais informativas e ajustadas ao contexto dos dados analisados.

Na próxima seção, será apresentado o processo de criação de gráfico de linhas utilizando a biblioteca `ggplot2`, que oferece uma abordagem mais flexível e estilizada para visualizações de dados.

8.2 Gráfico de linhas pelo ggplot2

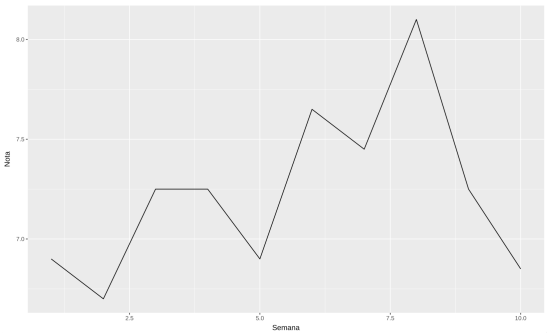
Como mencionado anteriormente no capítulo 4, os gráficos criados a partir do `ggplot2` seguem uma estrutura em camadas, sendo sempre necessário especificar a

base de dados utilizada, as variáveis a serem plotadas no gráfico, bem como o tipo de gráfico desejado.

Dando continuidade ao exemplo apresentado na seção anterior, será construído um gráfico de linhas para as variáveis `Semana` e `Nota` da base de dados `estudantes_tarefas`, que apresenta as semanas e a média da turma nas notas de tarefas de Matemática. Para a construção do gráfico de linhas, será utilizada a função `geom_line()`.

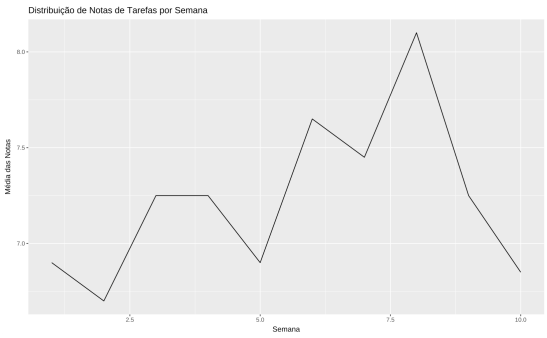
No gráfico de linhas, é necessário identificar qual variável será representada em cada eixo. No exemplo a seguir, utiliza-se a variável `Semana` para o eixo `x` e a variável `Nota` para o eixo `y`:

```
ggplot(data = estudantes_tarefas,
        aes(x = Semana, y = Nota))+
  geom_line()
```



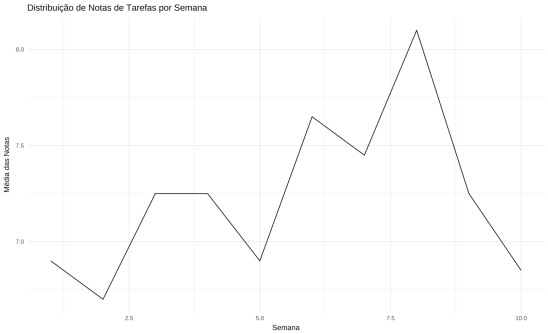
Para inserir os rótulos em um gráfico criado com o pacote `ggplot2`, utiliza-se a função `labs()`, que permite especificar o título principal do gráfico por meio do argumento `title`, o rótulo do eixo x com o argumento `x` e o rótulo do eixo y com o argumento `y`.

```
ggplot(data = estudantes_tarefas,
        aes(x = Semana, y = Nota))+
  geom_line()  +
  labs(title = "Distribuição de Notas
            de Tarefas por Semana",
       x = "Semana",
       y = "Média das Notas")
```



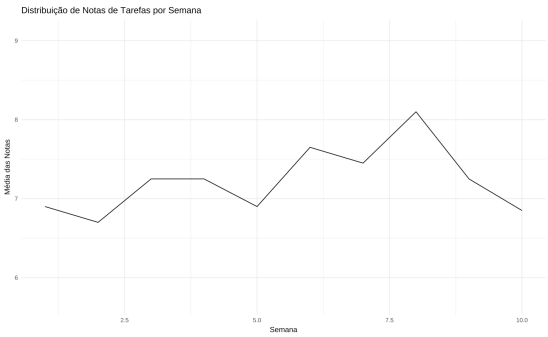
Para deixar o gráfico com um visual mais limpo e agradável, é possível utilizar a função `theme_minimal()`, que aplica um tema com grade (grid) mais discreta e elementos visuais simplificados.

```
ggplot(data = estudantes_tarefas,
        aes(x = Semana, y = Nota))+
  geom_line()  +
  labs(title = "Distribuição de Notas
           de Tarefas por Semana",
       x = "Semana",
       y = "Média das Notas") +
  theme_minimal()
```



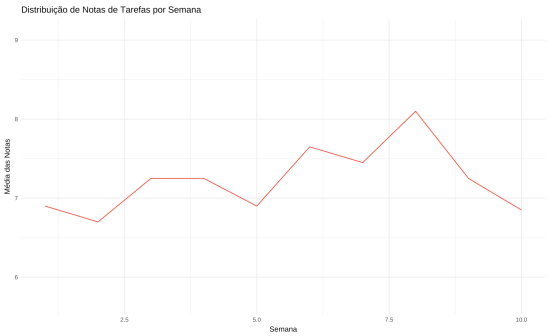
O pacote `ggplot2` define automaticamente os limites numéricos dos eixos `x` e `y`, mas é possível alterá-los utilizando a função `coord_cartesian()`, juntamente com os argumentos `xlim` e `ylim`. Esses argumentos recebem vetores no formato `c(valor_inicial, valor_final)`, que podem ser definidos de forma numérica ou por meio de funções. No exemplo a seguir, define-se que o limite inferior do eixo `y` corresponde ao valor mínimo das notas, e o limite superior, ao valor máximo das notas, utilizando, respectivamente, as funções `min()` e `max()`.

```
ggplot(data = estudantes_tarefas,
        aes(x = Semana, y = Nota))+
  geom_line()  +
  labs(title = "Distribuição de Notas
           de Tarefas por Semana",
       x = "Semana",
       y = "Média das Notas") +
  coord_cartesian(ylim = c(
    min(estudantes_tarefas$Nota)-1,
    max(estudantes_tarefas$Nota)+1))+
  theme_minimal()
```



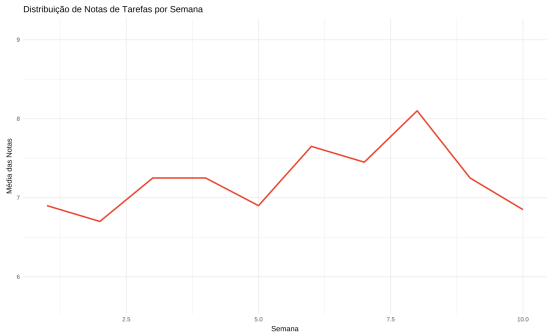
Para inserir cor à linha, utiliza-se o argumento `color` aplicado à função `geom_line()`. Para definir a cor da linha, basta especificá-la em inglês. Além disso, é possível visualizar a lista de cores disponíveis no R utilizando o comando `colors()` no console.

```
ggplot(data = estudantes_tarefas,
        aes(x = Semana, y = Nota))+
  geom_line(color = "red")  +
  labs(title = "Distribuição de Notas
           de Tarefas por Semana",
       x = "Semana",
       y = "Média das Notas") +
  coord_cartesian(ylim = c(
    min(estudantes_tarefas$Nota)-1,
    max(estudantes_tarefas$Nota)+1))+
  theme_minimal()
```



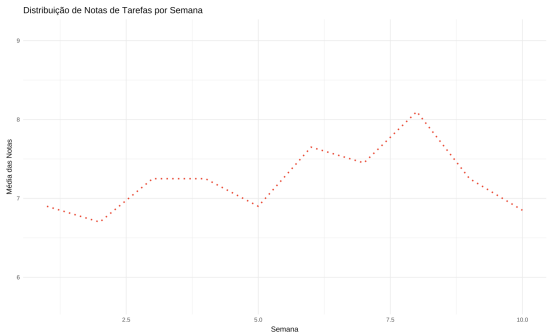
Com o argumento `size`, também na função `geom_line()`, pode-se alterar a espessura da linha no gráfico, o que é útil para destacar ou suavizar sua aparência, dependendo do objetivo visual. O valor de `size` é um número que define a espessura da linha, sendo aproximadamente 0,5 o valor padrão. Valores maiores que 0,5 tornam a linha mais espessa, enquanto valores menores que 0,5 a tornam mais fina.

```
ggplot(data = estudantes_tarefas,
       aes(x = Semana, y = Nota))+
  geom_line(color = "red",
            size = 1)  +
  labs(title = "Distribuição de Notas
            de Tarefas por Semana",
       x = "Semana",
       y = "Média das Notas") +
  coord_cartesian(ylim = c(
    min(estudantes_tarefas$Nota)-1,
    max(estudantes_tarefas$Nota)+1))+
  theme_minimal()
```



Além disso, pode-se customizar a linha com o argumento `linetype`, aplicado à função `geom_line()`, que define o tipo de linha no gráfico. O valor padrão (`linetype = "solid"`) gera uma linha contínua, mas é possível escolher entre diversas opções, como linhas tracejadas (`"dashed"`), pontilhadas (`"dotted"`), traço-ponto (`"dotdash"`), entre outras variações.

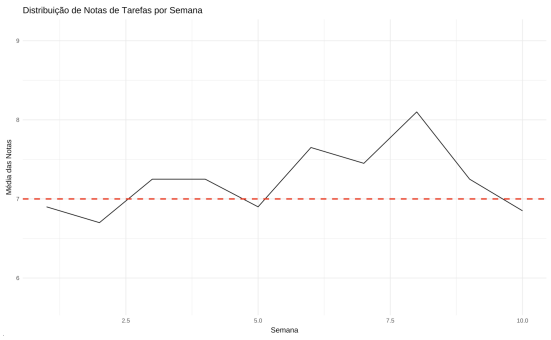
```
ggplot(data = estudantes_tarefas,
       aes(x = Semana, y = Nota))+
  geom_line(color = "red",
            size = 1,
            linetype = "dotted")  +
  labs(title = "Distribuição de Notas
            de Tarefas por Semana",
       x = "Semana",
       y = "Média das Notas") +
  coord_cartesian(ylim = c(
    min(estudantes_tarefas$Nota)-1,
    max(estudantes_tarefas$Nota)+1))+
  theme_minimal()
```



Por fim, é possível adicionar linhas de referência ao gráfico utilizando a função `geom_hline()`, que insere linhas horizontais com base no valor especificado no argumento `yintercept`. Essas linhas podem ser personalizadas de maneira semelhante às

linhas principais do gráfico, permitindo a modificação de cor, espessura e estilo, por meio dos argumentos `color`, `size` e `linetype`, respectivamente. No exemplo abaixo, é inserida uma linha horizontal no valor 7, representando a média mínima aceitável na escola. Isso permite identificar visualmente em quais semanas os alunos obtiveram notas abaixo dessa média.

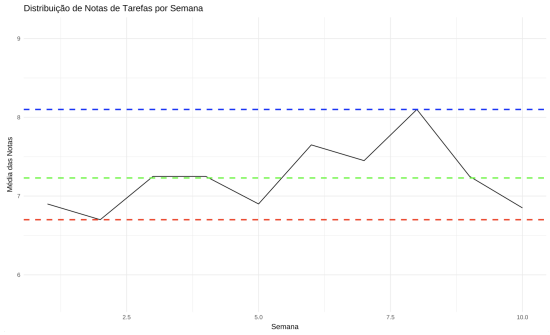
```
ggplot(data = estudantes_tarefas,
        aes(x = Semana, y = Nota))+
  geom_line(color = "black")  +
  geom_hline(yintercept = 7,
             color = "red",
             size = 1,
             linetype = "dashed") +
  labs(title = "Distribuição de Notas
            de Tarefas por Semana",
       x = "Semana",
       y = "Média das Notas") +
  coord_cartesian(ylim = c(
    min(estudantes_tarefas$Nota)-1,
    max(estudantes_tarefas$Nota)+1))+
  theme_minimal()
```



As linhas de referência também podem ser definidas a partir de funções estatísticas aplicadas aos dados, como o mínimo, a média e o máximo. Para isso, utiliza-se a função `geom_hline()` em conjunto com as funções `min()`, `mean()` e `max()`, respectivamente.

No exemplo abaixo, as linhas foram diferenciadas por cores: a linha do valor mínimo foi plotada em vermelho, a linha da média em verde e a linha do valor máximo em azul. O uso dessas linhas de referência contribui para destacar informações importantes no gráfico e facilita a interpretação de dados relevantes.

```
ggplot(data = estudantes_tarefas,
       aes(x = Semana, y = Nota))+
  geom_line(color = "black")  +
  geom_hline(yintercept =
       min(estudantes_tarefas$Nota),
       color = "red",
       linetype = "dashed",
       size = 1) +
  geom_hline(yintercept =
       mean(estudantes_tarefas$Nota),
       color = "green",
       linetype = "dashed",
       size = 1) +
  geom_hline(yintercept =
       max(estudantes_tarefas$Nota),
       color = "blue",
       linetype = "dashed",
       size = 1) +
  labs(title = "Distribuição de Notas
           de Tarefas por Semana",
       x = "Semana",
       y = "Média das Notas") +
  coord_cartesian(ylim = c(
    min(estudantes_tarefas$Nota)-1,
    max(estudantes_tarefas$Nota)+1))+
  theme_minimal()
```



Com isso, encerra-se a apresentação das principais funcionalidades do pacote `ggplot2` aplicadas à construção de gráficos de linhas no R. Ao longo das seções, foram abordados desde os elementos básicos da criação desses gráficos até personalizações mais avançadas, como limites dos eixos `x` e `y`, cores, estilos e linhas de referência. Essas ferramentas permitem que o usuário desenvolva visualizações mais claras, organizadas e alinhadas ao contexto dos dados, facilitando a interpretação e a comunicação dos resultados.

No próximo Capítulo, será abordada a construção de gráficos de dispersão. Esses gráficos são particularmente úteis para representar a relação entre duas variáveis quantitativas, permitindo observar padrões, tendências e possíveis correlações entre elas. Inicialmente, serão utilizadas as funções nativas do R para esse tipo de visualização. Em seguida, será apresentada a construção desses gráficos com o pacote `ggplot2`, que oferece recursos adicionais de personalização e análise gráfica.

9 GRÁFICO DE DISPERSÃO

O gráfico de dispersão é uma representação gráfica que permite visualizar e analisar a relação entre duas variáveis numéricas. Em sala de aula, esse tipo de gráfico pode ser utilizado para investigar se há uma relação entre as horas de estudo e as notas dos alunos, por exemplo, ou para explorar outros conjuntos de dados que envolvem pares de variáveis.

Sua principal vantagem é a capacidade de evidenciar padrões de associação, como correlações positivas, negativas ou inexistentes. O gráfico de dispersão facilita a identificação de tendências, agrupamentos ou valores atípicos, contribuindo para a compreensão da relação entre as variáveis analisadas.

Inicialmente, será demonstrado como plotar o gráfico de dispersão utilizando funções nativas do R e, posteriormente, utilizando funções da biblioteca `ggplot2`. Serão apresentadas as linhas de código e, ao lado, o gráfico gerado por essas linhas.

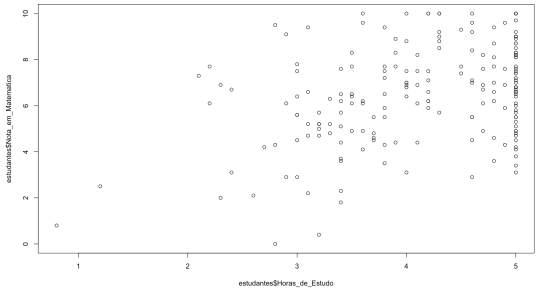
As Figuras apresentadas neste capítulo são de autoria própria, geradas a partir da execução dos códigos apresentados ao longo do texto. Assim, cada imagem corresponde diretamente à saída gráfica produzida pelos scripts indicados na seção.

9.1 Gráfico de dispersão por funções nativas

A função nativa do R para a criação de gráficos de dispersão é `plot`. Para consultar os parâmetros disponíveis para essa função, pode-se utilizar o comando `?plot`, que exibirá a documentação correspondente em inglês. A função `plot` pode ser usada para diversos gráficos, como pontos (`"p"`), linhas (`"l"`), pontos e linhas (`"b"`), degraus (`"s"` e `"S"`), ou histograma com linhas verticais (`"h"`), entre outros.

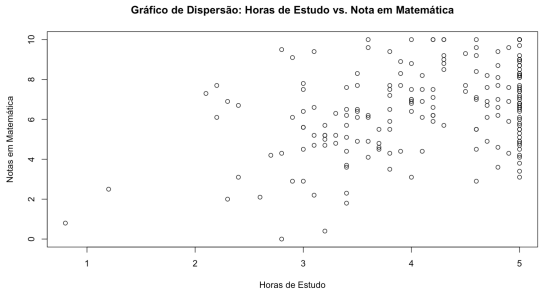
O foco deste capítulo são os gráficos de dispersão, portanto será utilizada a função `plot`. Por padrão, essa função utiliza o argumento `type = "p"`, não sendo necessário explicitá-lo. Para gerar o gráfico de dispersão, primeiro determina-se a variável da base de dados para o eixo `x` e, em seguida, a variável para o eixo `y`. Será utilizada a base de dados `estudantes`, que apresenta, entre outras variáveis, as horas de estudo dos estudantes e a nota obtida em Matemática. No exemplo a seguir, utilizamos a variável `Horas_de_Estudo` para o eixo `x` e a variável `Nota_em_Matematica` para o eixo `y`:

```
plot(estudantes$Horas_de_Estudo,
    estudantes$Nota_em_Matematica)
```



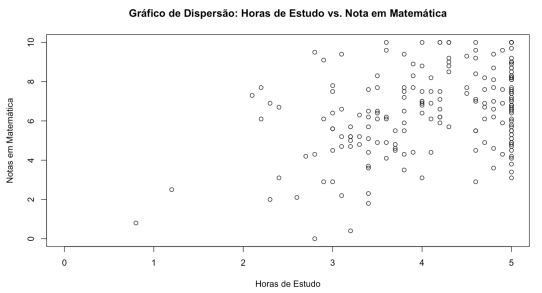
Perceba que, automaticamente, a função insere os rótulos dos eixos, baseado na base de dados utilizada. Para personalizar os rótulos do gráfico utiliza-se os seguintes argumentos: `main` que insere o título principal do gráfico, `xlab` para inserir o rótulo do eixo x e `ylab` para inserir o rótulo do eixo y.

```
plot(estudantes$Horas_de_Estudo,
    estudantes$Nota_em_Matematica,
     main = "Gráfico de Dispersão:
             Horas de Estudo vs.
             Nota em Matemática",
     xlab = "Horas de Estudo",
     ylab = "Notas em Matemática")
```



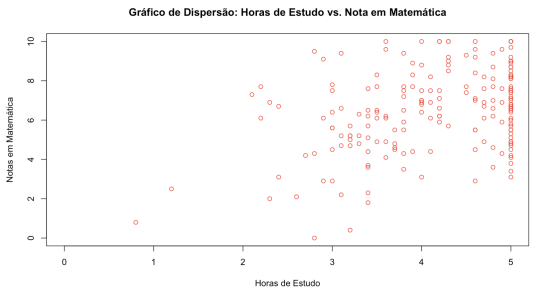
A função `plot` automaticamente define os limites numéricos dos eixos `x` e `y`, mas é possível alterar esses limites utilizando os argumentos `xlim` e `ylim`, passando os valores no formato `(valor_inicio, valor_final)`. No exemplo abaixo, o limite de `x` foi ajustado para `(0,5)`, ou seja, o valor inicial do eixo `x` é 0 e o valor máximo do eixo `x` é 5. De forma análoga, o limite de `y` foi alterado para `(0, 10)`,conforme os valores da distribuição.

```
plot(estudantes$Horas_de_Estudo,
    estudantes$Nota_em_Matematica,
     main = "Gráfico de Dispersão:
             Horas de Estudo vs.
             Nota em Matemática",
     xlab = "Horas de Estudo",
     ylab = "Notas em Matemática",
     xlim = c(0, 5),
     ylim = c(0, 10))
```



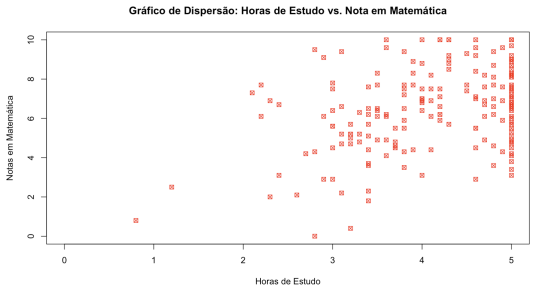
Para inserir cor aos pontos, utiliza-se o argumento `col`. Para definir a cor dos pontos, basta especificá-la em inglês. Além disso, é possível visualizar a lista de cores disponíveis no R utilizando o comando `colors()` no console.

```
plot(estudantes$Horas_de_Estudo,
    estudantes$Nota_em_Matematica,
     main = "Gráfico de Dispersão:
             Horas de Estudo vs.
             Nota em Matemática",
     xlab = "Horas de Estudo",
     ylab = "Notas em Matemática",
     xlim = c(0, 5),
     ylim = c(0, 10),
     col = "red")
```



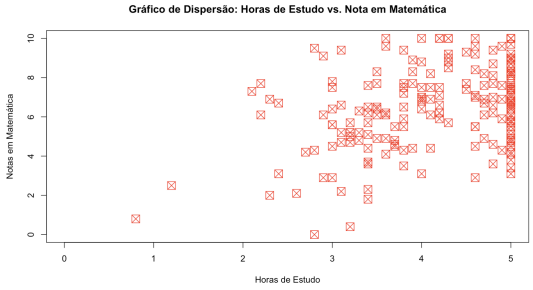
O argumento `pch` permite alterar o símbolo utilizado para representar os pontos no gráfico de dispersão. Essa personalização é útil tanto para destacar informações específicas quanto para ajustar a estética do gráfico, tornando-o mais claro e adequado ao contexto da análise. O valor de `pch` é um número inteiro que define qual símbolo será exibido. O valor padrão é 1, que corresponde a uma bolinha sem preenchimento. Alguns outros valores que podem ser utilizados são: 2 (triângulo), 3 (cruz), 4 (xis), 5 (losango), 6 (triângulo invertido), 7 (quadrado com xis) e 8 (asterisco), entre várias outras opções disponíveis.

```
plot(estudantes$Horas_de_Estudo,
    estudantes$Nota_em_Matematica,
     main = "Gráfico de Dispersão:
             Horas de Estudo vs.
             Nota em Matemática",
     xlab = "Horas de Estudo",
     ylab = "Notas em Matemática",
     xlim = c(0, 5),
     ylim = c(0, 10),
     col = "red",
     pch = 7)
```



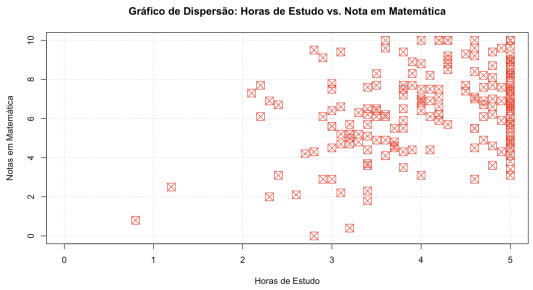
Além disso, é possível alterar o tamanho dos pontos no gráfico com o argumento `cex`, que controla o fator de escala aplicado ao tamanho padrão. O valor padrão é `cex = 1`, que corresponde ao tamanho base definido pelo R. Valores maiores que 1 aumentam o tamanho dos pontos, enquanto valores menores que 1 os reduzem.

```
plot(estudantes$Horas_de_Estudo,
    estudantes$Nota_em_Matematica,
     main = "Gráfico de Dispersão:
             Horas de Estudo vs.
             Nota em Matemática",
     xlab = "Horas de Estudo",
     ylab = "Notas em Matemática",
     xlim = c(0, 5),
     ylim = c(0, 10),
     col = "red",
     pch = 7,
     cex = 2)
```



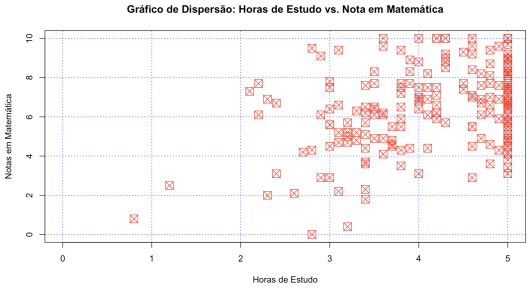
Outro elemento que pode ser útil no gráfico de dispersão é a presença do grid de orientação. O `grid` é um conjunto de linhas horizontais e verticais que ajudam a visualizar a escala dos eixos e a localização exata dos pontos no gráfico. Ele é sobreposto ao gráfico, ou seja, deve ser adicionado após a plotagem do gráfico.

```
plot(estudantes$Horas_de_Estudo,
    estudantes$Nota_em_Matematica,
     main = "Gráfico de Dispersão:
             Horas de Estudo vs.
             Nota em Matemática",
     xlab = "Horas de Estudo",
     ylab = "Notas em Matemática",
     xlim = c(0, 5),
     ylim = c(0, 10),
     col = "red",
     pch = 7,
     cex = 2)

grid()
```



Assim como os pontos do gráfico, o grid também pode ser personalizado. Essa personalização é semelhante à customização das linhas do gráfico, permitindo definir cor, espessura e tipo de linha. No exemplo abaixo, o grid foi configurado com a cor azul, espessura de 0.8 e tipo de linha igual a 5, que corresponde a uma linha pontilhada (para mais detalhes sobre os argumentos de personalização de linha, consulte o capítulo 8). Todas essas características são ajustadas diretamente por meio dos argumentos da função `grid()`.

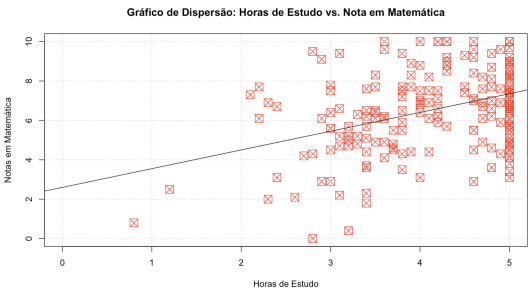
```
plot(estudantes$Horas_de_Estudo,
    estudantes$Nota_em_Matematica,
     main = "Gráfico de Dispersão:
             Horas de Estudo vs.
             Nota em Matemática",
     xlab = "Horas de Estudo",
     ylab = "Notas em Matemática",
     xlim = c(0, 5),
     ylim = c(0, 10),
     col = "red",
     pch = 7,
     cex = 2)

grid(col = "blue", lwd = 0.8, lty = 5)
```



Por fim, é possível adicionar linhas de referência ao gráfico, sendo a linha de regressão uma das mais utilizadas em gráficos de dispersão. Essa linha é especialmente útil para representar a tendência central da relação entre as duas variáveis analisadas, facilitando a identificação de padrões e correlações existentes nos dados, auxiliando na interpretação da força e direção da relação entre as variáveis.

Para adicionar a linha de regressão, utiliza-se a função `abline`, em conjunto com a função `lm()`, que ajusta um modelo linear entre as variáveis. No exemplo abaixo, a linha de regressão representa a relação entre as variáveis `Nota_em_Matematica` e `Horas_de_Estudo`, com base na base de dados `estudantes`. A linha foi plotada na cor preta, destacando-se sobre o gráfico de dispersão previamente gerado.

```
plot(estudantes$Horas_de_Estudo,
    estudantes$Nota_em_Matematica,
     main = "Gráfico de Dispersão:
             Horas de Estudo vs.
             Nota em Matemática",
     xlab = "Horas de Estudo",
     ylab = "Notas em Matemática",
     xlim = c(0, 5),
     ylim = c(0, 10),
     col = "red",
     pch = 7,
     cex = 2)

grid()

abline(lm(estudantes$Nota_em_Matematica ↲
↪~

↪estudantes$Horas_de_Estudo,
            data = estudantes),
       col = "black")
```



Com isso, as principais funcionalidades da função `plot` para criar gráfico de dispersão no R foram exploradas, incluindo personalizações como cores, tipos de símbolos, tamanho, limites dos eixos e a adição de linha de regressão. Essas configurações permitem gerar visualizações mais informativas e ajustadas ao contexto dos dados analisados.

Na próxima seção, será apresentado o processo de criação de gráfico de linhas utilizando a biblioteca `ggplot2`, que oferece uma abordagem mais flexível e estilizada para visualizações de dados.

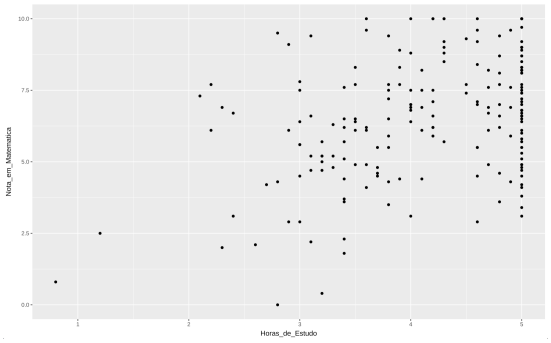
9.2 Gráfico de dispersão pelo ggplot2

Como mencionado anteriormente no capítulo 4, os gráficos criados a partir do `ggplot2` seguem uma estrutura em camadas, sendo sempre necessário especificar a base de dados utilizada, as variáveis a serem plotadas no gráfico, bem como o tipo de gráfico desejado.

Dando continuidade ao exemplo apresentado na seção anterior, será construído um gráfico de dispersão para as variáveis `Nota_em_Matematica` e `Horas_de_Estudo`, da base de dados `estudantes`, a qual apresenta as notas em Matemática e a quantidade de horas de estudo dedicadas à disciplina. Para a construção do gráfico de dispersão, será utilizada a função `geom_point()`.

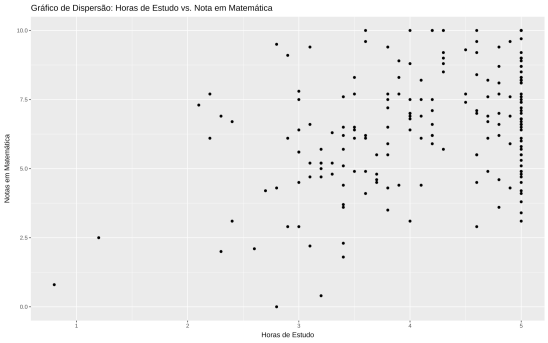
No gráfico de dispersão, é necessário identificar qual variável será representada em cada eixo. No exemplo a seguir, utiliza-se a variável `Horas_de_Estudo` no eixo `x` e a variável `Nota_em_Matematica` no eixo `y`:

```
ggplot(data = estudantes,
        aes(x = Horas_de_Estudo,
        y = Nota_em_Matematica)) +
  geom_point()
```



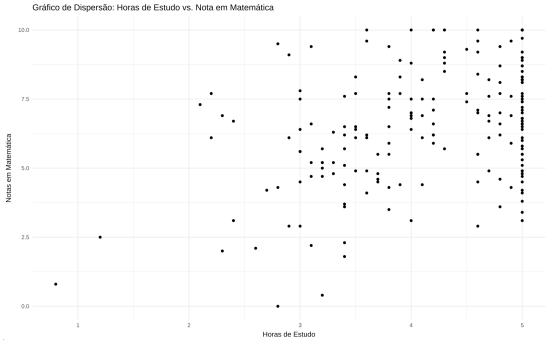
Para inserir os rótulos em um gráfico criado com o pacote `ggplot2`, utiliza-se a função `labs()`, que permite especificar o título principal do gráfico por meio do argumento `title`, o rótulo do eixo x com o argumento `x` e o rótulo do eixo y com o argumento `y`.

```
ggplot(data = estudantes,
        aes(x = Horas_de_Estudo,
        y = Nota_em_Matematica)) +
  geom_point()
  labs(title = "Gráfico de Dispersão:
            Horas de Estudo vs.
            Nota em Matemática",
       x = "Horas de Estudo",
       y = "Notas em Matemática")
```



Para deixar o gráfico com um visual mais limpo e agradável, é possível utilizar a função `theme_minimal()`, que aplica um tema com grade (grid) mais discreta e elementos visuais simplificados.

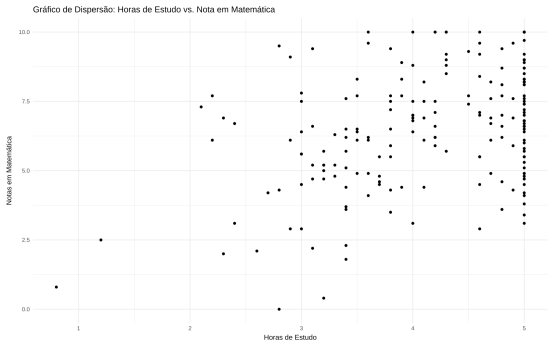
```
ggplot(data = estudantes,
        aes(x = Horas_de_Estudo,
        y = Nota_em_Matematica)) +
  geom_point()
  labs(title = "Gráfico de Dispersão:
            Horas de Estudo vs.
            Nota em Matemática",
       x = "Horas de Estudo",
       y = "Notas em Matemática") +
  theme_minimal()
```



O pacote `ggplot2` define automaticamente os limites numéricos dos eixos `x` e `y`, mas é possível personalizá-los utilizando a função `coord_cartesian()`, em conjunto com os argumentos `xlim` e `ylim`. Esses argumentos recebem vetores no formato `c(valor_inicial, valor_final)`, que podem ser definidos de forma numérica ou por meio de funções.

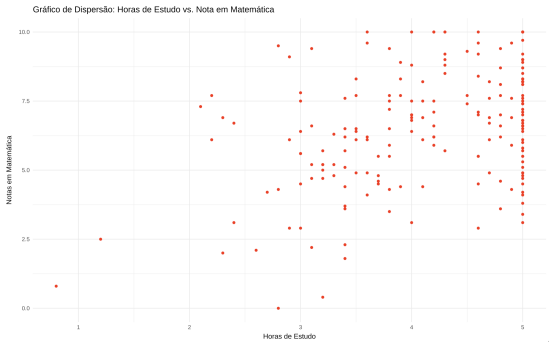
No exemplo a seguir, o limite inferior do eixo `x` é definido como o valor mínimo das horas estudadas, utilizando a função `min()`, enquanto o limite superior é fixado em 5. Para o eixo `y`, os limites vão de 0 a 10, correspondendo à faixa possível de notas em Matemática.

```
ggplot(data = estudantes,
        aes(x = Horas_de_Estudo,
        y = Nota_em_Matematica)) +
  geom_point()
  labs(title = "Gráfico de Dispersão:
            Horas de Estudo vs.
            Nota em Matemática",
       x = "Horas de Estudo",
       y = "Notas em Matemática") +
    coord_cartesian(xlim =
        c(min(estudantes$Horas_de_Estudo),5),
        ylim = c(0, 10)) +
  theme_minimal()
```



Para inserir cor aos pontos, utiliza-se o argumento `color` aplicado à função `geom_point()`. Para definir a cor do ponto, basta especificá-la em inglês. Além disso, é possível visualizar a lista de cores disponíveis no R utilizando o comando `colors()` no console.

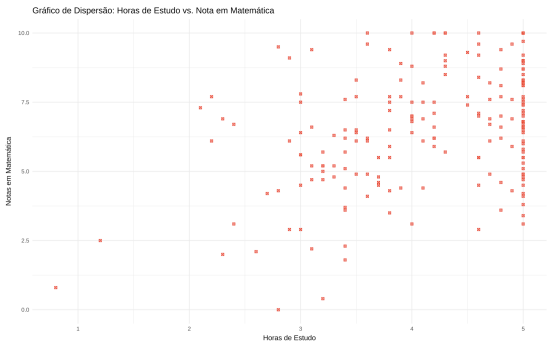
```
ggplot(data = estudantes,
        aes(x = Horas_de_Estudo,
        y = Nota_em_Matematica)) +
  geom_point(color = "red")
  labs(title = "Gráfico de Dispersão:
            Horas de Estudo vs.
            Nota em Matemática",
       x = "Horas de Estudo",
       y = "Notas em Matemática") +
    coord_cartesian(xlim =
        c(min(estudantes$Horas_de_Estudo),5),
        ylim = c(0, 10)) +
  theme_minimal()
```



Com o argumento `shape`, também na função `geom_point()`, pode-se customizar a forma dos pontos no gráfico. O valor padrão (`shape = 16`) gera pontos circulares preenchidos, mas é possível escolher entre diversas opções numéricas, entre 0 e 25, como: `0` (quadrado vazio), `1` (círculo vazio), `2` (triângulo apontando para cima), `15` (quadrado preenchido), `17` (triângulo preenchido) e `25` (triângulo apontando para baixo com borda).

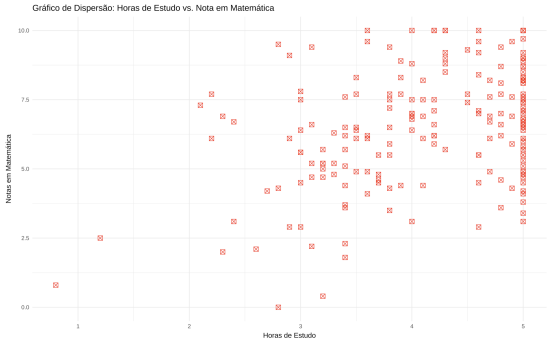
Essas opções permitem destacar diferentes grupos ou categorias no gráfico de maneira visualmente diferenciada. Para visualizar todos os formatos disponíveis, pode-se consultar a documentação do pacote `ggplot2` ou procurar uma tabela com os formatos de pontos (*shape chart*).

```
ggplot(data = estudantes,
        aes(x = Horas_de_Estudo,
        y = Nota_em_Matematica)) +
  geom_point(color = "red",
             shape = 7) +
  labs(title = "Gráfico de Dispersão:
            Horas de Estudo vs.
            Nota em Matemática",
       x = "Horas de Estudo",
       y = "Notas em Matemática") +
    coord_cartesian(xlim =
        c(min(estudantes$Horas_de_Estudo),5),
        ylim = c(0, 10)) +
  theme_minimal()
```



Além disso, pode-se customizar o ponto com o argumento `size`, aplicado à função `geom_point()`, que define o tamanho do ponto no gráfico. Esse parâmetro permite alterar a espessura visual dos pontos, o que é útil para destacar ou suavizar sua aparência, dependendo do objetivo visual. O valor de `size` é um número, sendo aproximadamente 1.5 o valor padrão. Valores maiores tornam os pontos mais espessos, enquanto valores menores os tornam mais sutis.

```
ggplot(data = estudantes,
        aes(x = Horas_de_Estudo,
        y = Nota_em_Matematica)) +
  geom_point(color = "red",
             shape = 7,
             size = 3) +
  labs(title = "Gráfico de Dispersão:
            Horas de Estudo vs.
            Nota em Matemática",
       x = "Horas de Estudo",
       y = "Notas em Matemática") +
    coord_cartesian(xlim =
        c(min(estudantes$Horas_de_Estudo),5),
        ylim = c(0, 10)) +
  theme_minimal()
```



Por fim, é possível adicionar linhas de tendência ao gráfico utilizando a função `geom_smooth()`, que ajusta uma curva baseada nos dados, facilitando a visualização de padrões ou relações entre as variáveis. Essa linha pode ser personalizada com os argumentos `color`, `size` e `linetype`, que definem, respectivamente, a cor, a espessura e o estilo da linha - para dúvidas, veja a seção 8.2. Por padrão, a função utiliza um método de suavização local (`loess`), mas também é possível especificar métodos como `lm`, que ajusta uma linha de regressão linear aos dados. O método `lm` é útil quando

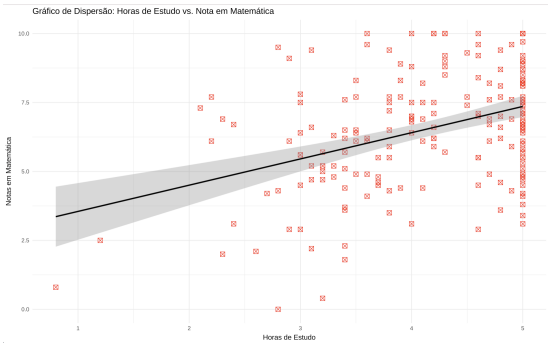
queremos verificar uma relação linear entre as variáveis. Ele ajusta uma linha reta que minimiza a diferença entre os pontos e a linha de tendência.

A inclusão dessa linha de tendência permite identificar, por exemplo, se há uma relação crescente entre as horas de estudo e as notas em matemática.

```
ggplot(data = estudantes,
       aes(x = Horas_de_Estudo,
       y = Nota_em_Matematica)) +
  geom_point(color = "red",
             shape = 7,
             size = 3) +
  geom_smooth(method = "lm",
              color = "black",
              se = FALSE) +
  labs(title = "Gráfico de Dispersão:
            Horas de Estudo vs.
            Nota em Matemática",
       x = "Horas de Estudo",
       y = "Notas em Matemática") +
    coord_cartesian(xlim =
       c(min(estudantes$Horas_de_Estudo),5),
       ylim = c(0, 10)) +
  theme_minimal()
```

Além disso, ao usar `se = TRUE`, podemos adicionar uma área sombreada ao redor da linha de tendência, que indica a incerteza do modelo, ou seja, a faixa onde a linha de tendência pode variar. Isso ajuda a visualizar o grau de confiança na linha ajustada e se o modelo linear é adequado para os dados.

```
ggplot(data = estudantes,
       aes(x = Horas_de_Estudo,
       y = Nota_em_Matematica)) +
  geom_point(color = "red",
             shape = 7,
             size = 3) +
  geom_smooth(method = "lm",
              color = "black",
              se = TRUE) +
  labs(title = "Gráfico de Dispersão:
            Horas de Estudo vs.
            Nota em Matemática",
       x = "Horas de Estudo",
       y = "Notas em Matemática") +
    coord_cartesian(xlim =
       c(min(estudantes$Horas_de_Estudo),5),
       ylim = c(0, 10)) +
  theme_minimal()
```



Com isso, encerra-se a apresentação das principais funcionalidades do pacote `ggplot2` aplicadas à construção de gráficos de dispersão no R. Ao longo das seções, foram abordados desde os elementos básicos da criação desses gráficos até personalizações mais avançadas, como limites dos eixos `x` e `y`, cores, estilos e linhas de referência. Essas ferramentas permitem que o usuário desenvolva visualizações mais claras, organizadas e alinhadas ao contexto dos dados, facilitando a interpretação e a comunicação dos resultados.

No próximo Capítulo, será abordada a replicabilidade dos materiais expostos neste manual com orientações práticas para que os professores possam aplicar os conhecimentos adquiridos, adaptando os scripts e atividades ao seu contexto escolar e promovendo uma aprendizagem ativa e significativa.

10 REPLICABILIDADE

Este manual tem como propósito não apenas apresentar códigos prontos, mas também apoiar a formação de professores capazes de compreender, adaptar e aplicar a construção de gráficos no ambiente R em diferentes contextos escolares. Mais do que dominar comandos específicos, o docente deve desenvolver segurança para explorar, autonomia para experimentar e a capacidade de relacionar os gráficos produzidos com situações de ensino, permitindo que os alunos compreendam conceitos matemáticos, estatísticos ou científicos de forma concreta e contextualizada.

Com o objetivo de assegurar transparência, reprodutibilidade e acesso permanente ao material, os *scripts* desenvolvidos e utilizados neste recurso educacional estão disponíveis em dois formatos: (1) em versão integral nos Apêndices deste trabalho e (2) em repositório público no GitHub, no link: https://github.com/isasterza/manual-graficos-rstudio (Butzen; Marins; Nava, 2026). Dessa forma, o leitor pode consultar o código no próprio texto acadêmico e, ao mesmo tempo, obter a versão digital para execução, atualização e adaptação em diferentes contextos.

O objetivo deste capítulo é fornecer orientações práticas para que os professores possam aplicar os conhecimentos adquiridos, estruturando atividades em que os alunos participem da coleta, manipulação e interpretação dos dados, promovendo aprendizagem significativa e o desenvolvimento do pensamento crítico.

Ao dominar os fundamentos da linguagem R, o professor conquista maior independência para trabalhar com dados, reduzindo a dependência de planilhas prontas ou recursos externos. Essa autonomia permite a elaboração de visualizações personalizadas, ajustadas à realidade da turma e aos objetivos pedagógicos de cada aula.

Além disso, o uso do R amplia o repertório didático e favorece abordagens interdisciplinares. Gráficos podem ser empregados para explorar conteúdos de matemática, ciências, geografia, entre outras áreas, promovendo conexões entre saberes e estimulando a análise crítica de informações.

Dessa forma, o R é apresentado não apenas como uma ferramenta técnica, mas como um recurso que permite integrar análise de dados à prática pedagógica, possibilitando que o professor articule conceitos, visualize padrões e promova atividades contextualizadas de ensino e aprendizagem.

10.1 Do exemplo à prática

Os *scripts* apresentados ao longo deste manual foram organizados para que possam ser reproduzidos e adaptados conforme o contexto educacional. A seguir, são sugeridas etapas para aplicar os gráficos em sala de aula:

1. **Escolher o conjunto de dados:** A coleta e o uso de dados reais dos próprios alunos tornam a atividade mais significativa, permitindo que eles percebam a utilidade da análise de dados no cotidiano e se envolvam de maneira ativa no processo de aprendizagem. É possível utilizar um dos datasets fornecidos neste manual ou criar um novo, a partir de informações coletadas com os alunos, como questionários, contagens ou experimentos. No Capítulo 3, foram apresentadas diversas formas de obter esses dados, possibilitando ao professor escolher aqueles mais adequados ao contexto escolar e aos interesses da turma.
2. **Selecionar o tipo de gráfico:** A escolha do gráfico deve refletir o objetivo pedagógico da aula. Por exemplo, gráficos de barras podem ser usados para comparar categorias e identificar diferenças entre grupos, enquanto gráficos de dispersão possibilitam a exploração de correlações e relações entre variáveis. A seleção adequada do gráfico contribui para a compreensão dos conceitos abordados e para a construção de uma narrativa visual consistente.
3. **Localizar e adaptar o código:** Os *scripts* apresentados ao longo deste manual podem ser consultados integralmente nos Apêndices e também estão disponíveis no repositório do GitHub (`https://github.com/isasterza/manual-graficos-rstudio`) (Butzen; Marins; Nava, 2026). Recomenda-se ajustar nomes de conjuntos de dados, variáveis, cores, títulos e legendas de acordo com a realidade da turma. Essa etapa permite que os alunos se familiarizem progressivamente com a construção de gráficos, compreendendo a função de cada elemento visual e desenvolvendo autonomia para propor alterações ou melhorias.
4. **Aplicar em sala:** Ao apresentar os gráficos, o professor deve conduzir discussões que envolvam análise, interpretação e questionamentos sobre os dados. Essa prática estimula o pensamento crítico, a capacidade de argumentação e a compreensão dos conceitos matemáticos ou científicos

de forma contextualizada, promovendo engajamento e participação ativa dos alunos.

10.2 Possibilidades de aplicação em diferentes contextos

A criação de gráficos com R pode ser integrada a diversas disciplinas, oferecendo aos professores oportunidades de explorar dados de forma visual e significativa. A seguir, são apresentadas algumas ideias de aplicação que podem ser adaptadas conforme o nível de ensino e os objetivos pedagógicos.

Na Matemática, os gráficos podem ser utilizados para analisar os resultados de avaliações, permitindo ao professor identificar padrões de desempenho entre os alunos. Além disso, é possível trabalhar o conceito de função por meio de gráficos de linhas, relacionando variáveis como tempo e distância, ou temperatura e hora do dia, por exemplo. Essa abordagem ajuda os estudantes a visualizarem o comportamento das funções e compreenderem suas propriedades.

Em aulas de Ciências, os gráficos podem representar dados coletados em experimentos, como a variação da temperatura em diferentes ambientes ou o crescimento de plantas sob diferentes condições. Gráficos de dispersão e de linhas são especialmente úteis para comparar variáveis e observar tendências.

Na Geografia, é possível utilizar gráficos de setores para representar a distribuição populacional por região, ou gráficos de linhas para acompanhar séries históricas de temperatura, precipitação ou crescimento urbano. Essas visualizações ajudam os alunos a interpretar dados espaciais e temporais com mais clareza.

A História também se beneficia da visualização de dados, especialmente ao representar eventos ao longo do tempo. Gráficos de linhas podem ser usados para mostrar a evolução de indicadores sociais, econômicos ou políticos, como taxas de alfabetização, crescimento populacional ou variações no PIB ao longo dos séculos.

Na Educação Física, o acompanhamento do desempenho dos alunos pode ser feito por meio de gráficos de barras ou de linhas, mostrando a evolução em testes físicos, frequência nas aulas ou participação em atividades. Essa visualização pode ser usada como ferramenta motivacional e de análise pedagógica.

Essas são apenas algumas possibilidades. O uso de gráficos em sala de aula não se limita à visualização de dados: ele pode ser um ponto de partida para discussões,

investigações e projetos interdisciplinares que envolvam os alunos na construção do conhecimento.

11 CONCLUSÃO E PRÓXIMOS PASSOS

Este manual foi elaborado como recurso de apoio à formação docente, com foco na utilização da linguagem R e do ambiente RStudio para a construção de gráficos e o ensino de estatística. Ao longo dos capítulos, foram apresentados comandos, *scripts* e exemplos aplicados.

A abordagem adotada busca favorecer a autonomia do professor no uso de tecnologias educacionais, permitindo que ele explore, modifique e aplique os recursos do R conforme as demandas de sua realidade escolar. O domínio dos fundamentos da linguagem e da estrutura dos gráficos contribui para a ampliação do repertório didático e para o desenvolvimento de práticas que envolvam análise de dados em sala de aula.

A aplicação dos *scripts* com dados reais, a adaptação dos exemplos às características das turmas e o aprofundamento em temas como manipulação de dados e visualizações interativas são caminhos para consolidar o uso do R como ferramenta pedagógica. A coleta e interpretação de dados pelos próprios alunos, mediada por gráficos, favorece o desenvolvimento de competências relacionadas ao letramento estatístico e ao uso crítico de tecnologias.

O uso do R no contexto educacional permite integrar a análise de dados ao ensino de estatística, promovendo a leitura e interpretação de informações quantitativas. Essa prática contribui para que os alunos compreendam fenômenos por meio de representações visuais, estabeleçam relações entre variáveis e reconheçam padrões, utilizando ferramentas computacionais como suporte à construção do conhecimento, formando cidadãos capazes de compreender e interpretar o mundo por meio dos dados.

Para continuar explorando

Se você se interessou pela criação de gráficos com R e deseja conhecer outras possibilidades além das abordadas neste manual, visite o site:

https://r-graph-gallery.com/

Esse repositório reúne uma ampla variedade de gráficos produzidos com R, organizados por tipo e finalidade. Cada exemplo é acompanhado do código correspon-

dente, permitindo que você estude, adapte e experimente novas visualizações em seus próprios projetos pedagógicos.

REFERÊNCIAS

BUTZEN, Isabella Sterza de Oliveira. **Recurso Educacional (PROFMAT): Configurando o R e o RStudio para construção de gráficos**. Vídeo no YouTube. Disponível em: https://youtu.be/P0ckTM_zMBw. Acesso em: 19 jan. 2026.

BUTZEN, Isabella Sterza de Oliveira; MARINS, Araceli Ciotti de; NAVA, Daniela Trentin. **isasterza/manual-graficos-rstudio**. Repositório de scripts do recurso educacional. Disponível em: https://github.com/isasterza/manual-graficos-rstudio/tree/main. Acesso em: 19 jan. 2026.

CRAN. **Comprehensive R Archive Network**. Disponível em: https://cran.r-project.org/. Acesso em: 19 jan. 2026.

FARIA, Pedro Duarte. **Introdução à Linguagem R**: seus fundamentos e sua prática. 5. ed. Belo Horizonte: [s.n.], 2024. ISBN 978-65-01-03954-1. Disponível em: https://pedro-faria.netlify.app/pt/publication/book/introducao_linguagem_r/. Acesso em: 19 jan. 2026.

POSIT. **RStudio Desktop**. Disponível em: https://posit.co/download/rstudio-desktop/. Acesso em: 19 jan. 2026.

R CORE TEAM. **R: A language and environment for statistical computing**. Vienna, Austria, 2024. Disponível em: https://www.R-project.org/. Acesso em: 19 jan. 2026.

WICKHAM, Hadley. **ggplot2: Elegant Graphics for Data Analysis**. Cham: Springer, 2016.

WILKINSON, Leland. **The Grammar of Graphics**. 2. ed. New York, NY: Springer, 2005.

APÊNDICE A — SCRIPTS DO RECURSO EDUCACIONAL

Os scripts utilizados e desenvolvidos no recurso educacional estão disponíveis neste apêndice e também no repositório do GitHub: https://github.com/isasterza/manual-graficos-rstudio/tree/main (Butzen; Marins; Nava, 2026).

Apêndice A.1 — Script 1 — Gráficos com funções nativas (graficos_funcoes_nativas.R)

```
# ============================================================
# SCRIPT: graficos_funcoes_nativas.R
#
# Autoria: Isabella Sterza de Oliveira Butzen
# Vínculo acadêmico: Dissertação (Mestrado Profissional em Matemática em Rede Nacional - PROFMAT)
# Produto educacional: MANUAL DIGITAL PARA CONSTRUÇÃO DE GRÁFICOS NO RSTUDIO: UM GUIA PRÁTICO EM LINGUAGEM R PARA DOCENTES
#
# Finalidade do script:
# - Reunir e documentar, de forma reprodutível, os comandos utilizados no produto educacional.
# - Servir como material de apoio (apêndice e repositório) para execução, adaptação e estudo.
#
# Observação sobre reprodutibilidade:
# - Este script foi organizado para fins didáticos. Alguns trechos podem exigir ajuste de caminhos
#   (setwd, leitura de arquivos) conforme o ambiente do(a) usuário(a).
#
# Dados de entrada (exemplos utilizados no produto):
# - "estudantes.csv"
# - "estudantes_tarefas_media.csv"
#
# Como citar (modelo sugerido):
# BUTZEN, Isabella Sterza de Oliveira. MANUAL DIGITAL PARA CONSTRUÇÃO DE GRÁFICOS NO RSTUDIO:
# um guia prático em linguagem R para docentes. Toledo: Universidade Tecnológica Federal do
# Paraná (UTFPR), 2026. Scripts e materiais suplementares disponíveis em: <https://github.com/isasterza/manual-graficos-rs
#
# Última atualização: <18-01-2026>
# ============================================================


### CARREGAR BIBLIOTECAS ###
library(readr)
library(dplyr)

### EXPORTAR DADOS ###

estudantes <- read_csv("estudantes.csv")
estudantes_tarefas <- read_csv("estudantes_tarefas_media.csv")

### CONHECENDO A BASE DE DADOS ###

View(estudantes)
str(estudantes)
```

```
estudantes$Horas_de_Estudo <- as.numeric(estudantes$Horas_de_Estudo) #Convertendo váriaveis necessarias

ncol(estudantes)
nrow(estudantes)
colnames(estudantes)


estudantes <- estudantes %>%
  rename(
    Nome = Nome,
    Serie = Serie,
    Idade = Idade,
    Horas_de_Estudo = "Horas de Estudo",
    Nota_em_Matematica = "Nota em Matemática",
    Extracurricular = Extracurricular,
    Presencas = Presenças,
    Sexo = Sexo,
    Musica_Favorita = "Música Favorita",
    Distancia_Escola_km = "Distância Escola (km)"
  )

####################################
### GRÁFICOS COM FUNÇÕES NATIVAS ###
####################################

# Gráfico de barras: barplot (usando ?barplot)
#####
barplot(table(estudantes$Idade)) # grafico em cima de uma tabela de frequencia

# argumentos:
# main: título principal do gráfico
# xlab: rótulo do eixo x
# ylab: rótulo do eixo y

barplot(table(estudantes$Idade),
        main = "Distribuição dos alunos por Idade",
        xlab = "Idade",
        ylab = "Frequência")

# space: define o espaço entre as barras proporcional a largura da barra.

barplot(table(estudantes$Idade),
        main = "Distribuição dos alunos por Idade",
        xlab = "Idade",
        ylab = "Frequência",
        space = 1)

# xlim e ylim: define o limite para os eixos x e y (inicio, fim)
barplot(table(estudantes$Idade),
```

```
        main = "Distribuição dos alunos por Idade",
        xlab = "Idade",
        ylab = "Frequência",
        space = 1,
        ylim = c(0, 60))

# col:  define as cores do gráfico
barplot(table(estudantes$Idade),
        main = "Distribuição dos alunos por Idade",
        xlab = "Idade",
        ylab = "Frequência",
        space = 1,
        ylim = c(0, 60),
        col = "red")

barplot(table(estudantes$Idade),
        main = "Distribuição dos alunos por Idade",
        xlab = "Idade",
        ylab = "Frequência",
        space = 1,
        ylim = c(0, 60),
        col = c("red", "blue", "green", "yellow", "purple"))

barplot(table(estudantes$Idade),
        main = "Distribuição dos alunos por Idade",
        xlab = "Idade",
        ylab = "Frequência",
        space = 1,
        ylim = c(0, 60),
        col = c("red", "blue", "green")) #Quando tem menos cores é preenchido sozinho

# border: define a cor das bordas das barras (ou remove com NA)
barplot(table(estudantes$Idade),
        main = "Distribuição dos alunos por Idade",
        xlab = "Idade",
        ylab = "Frequência",
        space = 1,
        ylim = c(0, 60),
        col = c("red", "blue", "green", "yellow", "purple"),
        border = NA)

# legend.text: adiciona uma legenda no gráfico
barplot(table(estudantes$Idade),
                          main = "Distribuição dos alunos por Idade",
                          xlab = "Idade",
                          ylab = "Frequência",
                          space = 1,
                          ylim = c(0, 60),
                          col = c("red", "blue", "green", "yellow", "purple"),
```

```
                        border = NA,
                        legend.text = c("11 anos", "12 anos", "13 anos", "14 anos", "15 anos"))

# Se precisar ajudar o tamanho da legenda, utilizar args.legend (cex altera o tamanho do texto, x  e y controlam a posicao
barplot(table(estudantes$Idade),
        main = "Distribuição dos alunos por Idade",
        xlab = "Idade",
        ylab = "Frequência",
        space = 1,
        ylim = c(0, 60),
        col = c("red", "blue", "green", "yellow", "purple"),
        border = NA,
        legend.text = c("11 anos", "12 anos", "13 anos", "14 anos", "15 anos"),
        args.legend = list(cex = 0.8, # Reduz tamanho do texto
                           x = "topright", inset = -0.05))


# Adicionar os valores das frequências no gráfico (posicao_x, posicao_y, valores)
text(
  x = grafico_barras,             # Posições das barras no eixo x
  y = table(estudantes$Idade) + 2,     # Valores das barras + um deslocamento para cima
  labels = table(estudantes$Idade)      # Valores a serem exibidos
)

#####
# Histograma: hist (usando ?hist)
#####
hist(estudantes$Nota_em_Matematica)

# argumentos:
# main: título principal do gráfico
# xlab: rótulo do eixo x
# ylab: rótulo do eixo y
hist(estudantes$Nota_em_Matematica,
     main = "Distribuição das Notas em Matemática",
     xlab = "Nota em Matemática",
     ylab = "Frequência")

# xlim e ylim: define o limite para os eixos x e y (inicio, fim)
hist(estudantes$Nota_em_Matematica,
     main = "Distribuição das Notas em Matemática",
     xlab = "Nota em Matemática",
     ylab = "Frequência",
     xlim = c(0, 10),
     ylim = c(0, 50))

# breaks: define a quantidade de intervalos
hist(estudantes$Nota_em_Matematica,
     main = "Distribuição das Notas em Matemática",
```

```
     xlab = "Nota em Matemática",
     ylab = "Frequência",
     xlim = c(0, 10),
     ylim = c(0, 80),
     breaks = 6) # o R ajusta automaticamente para o valor mais adequado (nesse caso, ajustou para 5)

hist(estudantes$Nota_em_Matematica,
     main = "Distribuição das Notas em Matemática",
     xlab = "Nota em Matemática",
     ylab = "Frequência",
     xlim = c(0, 10),
     ylim = c(0, 30),
     breaks = seq(0, 10, by = 0.5)) #intervalos de 0 a 10 de 0.5 em 0.5

# col: define as cores do gráfico
hist(estudantes$Nota_em_Matematica,
     main = "Distribuição das Notas em Matemática",
     xlab = "Nota em Matemática",
     ylab = "Frequência",
     xlim = c(0, 10),
     ylim = c(0, 80),
     breaks = 5,
     col = "red")

hist(estudantes$Nota_em_Matematica,
     main = "Distribuição das Notas em Matemática",
     xlab = "Nota em Matemática",
     ylab = "Frequência",
     xlim = c(0, 10),
     ylim = c(0, 80),
     breaks = 5,
     col = c("red", "blue", "green", "yellow", "purple"))

hist(estudantes$Nota_em_Matematica,
     main = "Distribuição das Notas em Matemática",
     xlab = "Nota em Matemática",
     ylab = "Frequência",
     xlim = c(0, 10),
     ylim = c(0, 80),
     breaks = 5,
     col = c("red", "blue", "green", "yellow")) #Se a quantidade de breaks e cor for diferente, ele vai recomeçar a colora

# border: define a cor das bordas das barras (ou remove com NA)
hist(estudantes$Nota_em_Matematica,
     main = "Distribuição das Notas em Matemática",
     xlab = "Nota em Matemática",
     ylab = "Frequência",
     xlim = c(0, 10),
     ylim = c(0, 80),
```

```
     breaks = 5,
     col = c("red", "blue", "green", "yellow", "purple"),
     border = NA)

# Colocar valores no histograma: hist para obter as contagens sem plotar
h <- hist(estudantes$Nota_em_Matematica,
          main = "Distribuição das Notas em Matemática",
          xlab = "Nota em Matemática",
          ylab = "Frequência",
          xlim = c(0, 10),
          ylim = c(0, 80),
          breaks = 5,
          col = c("red", "blue", "green", "yellow", "purple"),
          border = NA)

text(
  x = h$mids,                       # Posições dos intervalos no eixo x
  y = h$counts + 2,               # Valores das barras + um deslocamento para cima
  labels = h$counts                 # Valores a serem exibidos
)



#####
# Gráfico de setores: pie (usando ?pie)
#####
pie(table(estudantes$Serie))

# argumentos:
# main: título do gráfico
pie(
  table(estudantes$Serie),
  main = "Distribuição de Alunos por Série Escolar")

# labels: adiciona o valor e o nome da categoria
pie(
  table(estudantes$Serie),
  main = "Distribuição de Alunos por Série Escolar",
  labels = table(estudantes$Serie))

pie(
  table(estudantes$Serie),
  main = "Distribuição de Alunos por Série Escolar",
  labels = paste(round(100 * table(estudantes$Serie) / sum(table(estudantes$Serie)), 1), "%"))

pie(
  table(estudantes$Serie),
  main = "Distribuição de Alunos por Série Escolar",
  labels = paste(c("6 Ano", "7 Ano", "8 Ano", "9 Ano"), ":", table(estudantes$Serie)))
```

```
# col: define as cores do gráfico
pie(
  table(estudantes$Serie),
  main = "Distribuição de Alunos por Série Escolar",
  labels = paste(c("6 Ano", "7 Ano", "8 Ano", "9 Ano"), ":", table(estudantes$Serie)),
  col = c("red", "blue", "green", "yellow"))

pie(
  table(estudantes$Serie),
  main = "Distribuição de Alunos por Série Escolar",
  labels = paste(c("6 Ano", "7 Ano", "8 Ano", "9 Ano"), ":", round(100 * table(estudantes$Serie) / sum(table(estudantes$Se
  col = c("red", "blue", "green", "yellow"))


# border: define a cor das bordas das barras (ou remove com NA)
pie(
  table(estudantes$Serie),
  main = "Distribuição de Alunos por Série Escolar",
  labels = paste(c("6 Ano", "7 Ano", "8 Ano", "9 Ano"), ":", table(estudantes$Serie)),
  col = c("red", "blue", "green", "yellow"),
  border = NA)

pie(
  table(estudantes$Serie),
  main = "Distribuição de Alunos por Série Escolar",
  labels = paste(c("6 Ano", "7 Ano", "8 Ano", "9 Ano"), ":", round(100 * table(estudantes$Serie) / sum(table(estudantes$Se
  col = c("red", "blue", "green", "yellow"),
  border = NA)

# legend: (posicao, nome_categorias, cores_correspondentes)
pie(
  table(estudantes$Serie),
  main = "Distribuição de Alunos por Série Escolar",
  labels = paste(round(100 * table(estudantes$Serie) / sum(table(estudantes$Serie)), 1), "%"),
  col = c("red", "blue", "green", "yellow"),
  border = NA)

legend(
  "topright",
  legend = c("6º Ano", "7º Ano", "8º Ano", "9º Ano"),
  fill = c("red", "blue", "green", "yellow")
)

#####
# Gráfico de linhas: plot type: l
#####
plot(estudantes_tarefas$Semana,estudantes_tarefas$Nota, type = "l")

# argumentos:
```

```
# main: título principal do gráfico
# xlab: rótulo do eixo x
# ylab: rótulo do eixo y
plot(estudantes_tarefas$Semana,
     estudantes_tarefas$Nota,
     type = "l",
     main = "Distribuição de Notas de Tarefas por Semana",
     xlab = "Semana",
     ylab = "Média das Notas")

# xlim e ylim: define o limite para os eixos x e y (inicio, fim)
plot(estudantes_tarefas$Semana,
     estudantes_tarefas$Nota,
     type = "l",
     main = "Distribuição de Notas de Tarefas por Semana",
     xlab = "Semana",
     ylab = "Média das Notas",
     xlim = c(1, 10),
     ylim = c(6, 9))

plot(
    estudantes_tarefas$Semana,
    estudantes_tarefas$Nota,
    type = "l",
    main = "Distribuição de Notas de Tarefas por Semana",
    xlab = "Semana",
    ylab = "Média das Notas",
    xlim = c(1,10),
    ylim = c(min(estudantes_tarefas$Nota), max(estudantes_tarefas$Nota))) #limites de acordo com os valores da tabela


# col: define a cor da linha
plot(
    estudantes_tarefas$Semana,
    estudantes_tarefas$Nota,
    type = "l",
    main = "Distribuição de Notas de Tarefas por Semana",
    xlab = "Semana",
    ylab = "Média das Notas",
    xlim = c(1,10),
    ylim = c(min(estudantes_tarefas$Nota), max(estudantes_tarefas$Nota)),
    col = "red")

# lwd: define a largura da linha
plot(
    estudantes_tarefas$Semana,
    estudantes_tarefas$Nota,
    type = "l",
    main = "Distribuição de Notas de Tarefas por Semana",
```

```
    xlab = "Semana",
    ylab = "Média das Notas",
    xlim = c(1,10),
    ylim = c(min(estudantes_tarefas$Nota), max(estudantes_tarefas$Nota)),
    col = "red",
    lwd = 2)

# lty: define o tipo de linha (1 sólida (padrao), 2 tracejada, 3 pontinhada, entre outros)
plot(
    estudantes_tarefas$Semana,
    estudantes_tarefas$Nota,
    type = "l",
    main = "Distribuição de Notas de Tarefas por Semana",
    xlab = "Semana",
    ylab = "Média das Notas",
    xlim = c(1,10),
    ylim = c(min(estudantes_tarefas$Nota), max(estudantes_tarefas$Nota)),
    col = "red",
    lwd = 2,
    lty = 2)

# grid: adiciona linhas de grade ao grafico
plot(
  estudantes_tarefas$Semana,
  estudantes_tarefas$Nota,
  type = "l",
  main = "Distribuição de Notas de Tarefas por Semana",
  xlab = "Semana",
  ylab = "Média das Notas",
  xlim = c(1,10),
  ylim = c(min(estudantes_tarefas$Nota), max(estudantes_tarefas$Nota)),
  col = "red",
  lwd = 2,
  lty = 1)
grid()

grid(col = "blue", lty = 5, lwd = 0.8) # personalizando o grid

# Adicionando linhas de referencia no grid
abline(h = min(estudantes_tarefas$Nota), col = "blue", lty = 3, lwd = 2)
abline(h = mean(estudantes_tarefas$Nota), col = "green", lty = 3, lwd = 2)
abline(h = max(estudantes_tarefas$Nota), col = "black", lty = 3, lwd = 2)



#####
# Gráfico  de dispersão: plot
#####
plot(estudantes$Horas_de_Estudo, estudantes$Nota_em_Matematica) # por padrão, variavel que representa o eixo x primeiro
```

```
# argumentos:
# main: título principal do gráfico
# xlab: rótulo do eixo x
# ylab: rótulo do eixo y
plot(estudantes$Horas_de_Estudo, estudantes$Nota_em_Matematica,
     main = "Gráfico de Dispersão: Horas de Estudo vs. Nota em Matemática",
     xlab = "Horas de Estudo",
     ylab = "Notas em Matemática")

# xlim e ylim: define o limite para os eixos x e y (inicio, fim)
plot(estudantes$Horas_de_Estudo, estudantes$Nota_em_Matematica,
     main = "Gráfico de Dispersão: Horas de Estudo vs. Nota em Matemática",
     xlab = "Horas de Estudo",
     ylab = "Notas em Matemática",
     xlim = c(0, 5),
     ylim = c(0, 10))

# col: define a cor dos pontos
plot(estudantes$Horas_de_Estudo, estudantes$Nota_em_Matematica,
     main = "Gráfico de Dispersão: Horas de Estudo vs. Nota em Matemática",
     xlab = "Horas de Estudo",
     ylab = "Notas em Matemática",
     xlim = c(0, 5),
     ylim = c(0, 10),
     col = "red")

# pch: define o simbolo usado para os pontos (pode ser letras)
plot(estudantes$Horas_de_Estudo, estudantes$Nota_em_Matematica,
     main = "Gráfico de Dispersão: Horas de Estudo vs. Nota em Matemática",
     xlab = "Horas de Estudo",
     ylab = "Notas em Matemática",
     xlim = c(0, 5),
     ylim = c(0, 10),
     col = "red",
     pch = 9)

#cex: define o tamanho do simbolo
plot(estudantes$Horas_de_Estudo, estudantes$Nota_em_Matematica,
     main = "Gráfico de Dispersão: Horas de Estudo vs. Nota em Matemática",
     xlab = "Horas de Estudo",
     ylab = "Notas em Matemática",
     xlim = c(0, 5),
     ylim = c(0, 10),
     col = "red",
     pch = 7,
     cex = 2)

# grid: define as linhas de grade
grid()
```

```
grid(col = "blue", lty = 3, lwd = 0.8) # personalizando o grid

# Adicionando linhas de referencia no grid (linha de regressão)
abline(lm(Nota_em_Matematica ~ Horas_de_Estudo, data = estudantes), col = "black")
```

Apêndice A.2 — Script 2 — Gráficos com ggplot (`graficos_ggplot.R`)

```
# ============================================================
# SCRIPT: graficos_ggplot.R
#
# Autoria: Isabella Sterza de Oliveira Butzen
# Vínculo acadêmico: Dissertação (Mestrado Profissional em Matemática em Rede Nacional - PROFMAT)
# Produto educacional: MANUAL DIGITAL PARA CONSTRUÇÃO DE GRÁFICOS NO RSTUDIO: UM GUIA PRÁTICO EM LINGUAGEM R PARA DOCENTES
#
# Finalidade do script:
# - Reunir e documentar, de forma reprodutível, os comandos utilizados no produto educacional.
# - Servir como material de apoio (apêndice e repositório) para execução, adaptação e estudo.
#
# Observação sobre reprodutibilidade:
# - Este script foi organizado para fins didáticos. Alguns trechos podem exigir ajuste de caminhos
#   (setwd, leitura de arquivos) conforme o ambiente do(a) usuário(a).
#
# Dados de entrada (exemplos utilizados no produto):
# - "estudantes.csv"
# - "estudantes_tarefas_media.csv"
#
# Como citar (modelo sugerido):
# BUTZEN, Isabella Sterza de Oliveira. MANUAL DIGITAL PARA CONSTRUÇÃO DE GRÁFICOS NO RSTUDIO:
# um guia prático em linguagem R para docentes. Toledo: Universidade Tecnológica Federal do
# Paraná (UTFPR), 2026. Scripts e materiais suplementares disponíveis em: <https://github.com/isasterza/manual-graficos-rs
#
# Última atualização: <18-01-2026>
# ============================================================


### CARREGAR BIBLIOTECAS ###
library(readr)
library(ggplot2)
library(RColorBrewer)

### EXPORTAR DADOS ###

estudantes <- read_csv("estudantes.csv")
estudantes_tarefas <- read_csv("estudantes_tarefas_media.csv")

### CONHECENDO A BASE DE DADOS ###
```

```
View(estudantes)
str(estudantes)
ncol(estudantes)
nrow(estudantes)
colnames(estudantes)


# Estrutura básica do ggplot
# ggplot (data = df, aes(x = <variável_x> , y = <variável_y> )) + quais variaveis estarão no gráfico
#   tipo_grafico() + (geom_bar, geom_histogram, geom_line, geom_point)
#   labs(title = , x = , y = ) + quais são os rótulos que deseja mostrar nas variáveis


#####
# Gráfico de barras
#####
ggplot(data = estudantes, aes(x = factor(Idade))) +
  geom_bar()

# Inserindo rótulos
ggplot(data = estudantes, aes(x = factor(Idade))) +
  geom_bar() +
  labs(title = "Distribuição dos alunos por Idade",
       x = "Idade",
       y = "Frequência")

# Deixando o gráfico visualmente mais leve (theme_minimal deixo o eixo cartesiano mais simples, theme_void deixa tudo em b
ggplot(data = estudantes, aes(x = factor(Idade))) +
  geom_bar() +
  labs(title = "Distribuição dos alunos por Idade",
       x = "Idade",
       y = "Frequência") +
  theme_minimal()

# width: Ajusta a largura das barras
ggplot(data = estudantes, aes(x = factor(Idade))) +
  geom_bar(width = 0.5) +
  labs(title = "Distribuição dos alunos por Idade",
       x = "Idade",
       y = "Frequência") +
  theme_minimal()

# xlim e ylim: define o limite para os eixos x e y (inicio, fim)
ggplot(data = estudantes, aes(x = factor(Idade))) +
  geom_bar(width = 0.5) +
  labs(title = "Distribuição dos alunos por Idade",
       x = "Idade",
       y = "Frequência") +
```

```
  theme_minimal() +
  ylim(0, max(table(estudantes$Idade)) + 5) #Define que a altura do eixo y é o maximo + 5 unidades

# fill: define a cor do gráfico - quando você pinta por categoria, diferenciando as cores, o ggplot gera legenda

# aplicar a cor em cada categoria, mapeando a váriavel - aes
ggplot(data = estudantes, aes(x = factor(Idade), fill = factor(Idade))) +
  geom_bar(width = 0.5) +
  labs(title = "Distribuição dos alunos por Idade",
       x = "Idade",
       y = "Frequência") +
  theme_minimal() +
  ylim(0, max(table(estudantes$Idade)) + 5)

# aplicar a cor geral no gráfico - geom_bar
ggplot(data = estudantes, aes(x = factor(Idade))) +
  geom_bar(width = 0.5, fill = "darkgreen") +
  labs(title = "Distribuição dos alunos por Idade",
       x = "Idade",
       y = "Frequência") +
  theme_minimal() +
  ylim(0, max(table(estudantes$Idade)) + 5)

# fill + scale_fill_manual: escolher uma paleta de cores de forma manual
ggplot(data = estudantes, aes(x = factor(Idade), fill = factor(Idade))) +
  geom_bar(width = 0.5) +
  scale_fill_manual(values = c("11" = "red", "12" = "blue", "13" = "green")) +
  labs(title = "Distribuição dos alunos por Idade",
       x = "Idade",
       y = "Frequência") +
  theme_minimal() +
  ylim(0, max(table(estudantes$Idade)) + 5)

# fill + scale_fill_brewer: escolher uma paleta de cores pré-definida com o pacote RColorBrewer (ver ?RColorBrewer)
ggplot(data = estudantes, aes(x = factor(Idade), fill = factor(Idade))) +
  geom_bar(width = 0.5) +
  scale_fill_brewer(palette = "Set2") +
  labs(title = "Distribuição dos alunos por Idade",
       x = "Idade",
       y = "Frequência") +
  theme_minimal() +
  ylim(0, max(table(estudantes$Idade)) + 5)

# color: define as bordas das barras
ggplot(data = estudantes, aes(x = factor(Idade), fill = factor(Idade))) +
  geom_bar(width = 0.5, color = "black") +
  labs(title = "Distribuição dos alunos por Idade",
       x = "Idade",
       y = "Frequência") +
```

```
  theme_minimal() +
  ylim(0, max(table(estudantes$Idade)) + 5)

# legenda: vem automatica mas pode ser editada
# scale_fill_manual(quando ja usado) ou scale_fill_discrete (labels = c("variavel_1", "variavel_2")) editar os rótulos de
ggplot(data = estudantes, aes(x = factor(Idade), fill = factor(Idade))) +
  geom_bar(width = 0.5, color = "black") +
  scale_fill_discrete(labels = c("11 anos", "12 anos", "13 anos", "14 anos", "15 anos")) +
  labs(title = "Distribuição dos alunos por Idade",
       x = "Idade",
       y = "Frequência") +
  theme_minimal() +
  ylim(0, max(table(estudantes$Idade)) + 5)

# labs (fill = "Titulo da Legenga")
ggplot(data = estudantes, aes(x = factor(Idade), fill = factor(Idade))) +
  geom_bar(width = 0.5, color = "black") +
  scale_fill_discrete(labels = c("11 anos", "12 anos", "13 anos", "14 anos", "15 anos")) +
  labs(title = "Distribuição dos alunos por Idade",
       x = "Idade",
       y = "Frequência",
       fill = "Idades dos Alunos") +
  theme_minimal() +
  ylim(0, max(table(estudantes$Idade)) + 5)

# geom_text: Adicionar o valor das frequências e outros textos
# aes(label =   after_stat(count)): substitui pelos valores da contagem
# stat: qual é o tipo de método estatistico aplicando (nesse caso contagem - count)
# vjust: define a posição (por padrão o valor fica na metade da linha superior do gráfico)
ggplot(data = estudantes, aes(x = factor(Idade), fill = factor(Idade))) +
  geom_bar(width = 0.5, color = "black") +
  geom_text(aes(label =   after_stat(count)), stat = "count", vjust = -0.5) +
  scale_fill_discrete(labels = c("11 anos", "12 anos", "13 anos", "14 anos", "15 anos")) +
  labs(title = "Distribuição dos alunos por Idade",
       x = "Idade",
       y = "Frequência",
       fill = "Idades dos Alunos") +
  theme_minimal() +
  ylim(0, max(table(estudantes$Idade)) + 5)

# theme: Modificar elementos do gráfico
# axis.text.x: Controla o texto do eixo X (categorias das barras)
# element_blank(): Remove o elemento do gráfico
ggplot(data = estudantes, aes(x = factor(Idade), fill = factor(Idade))) +
  geom_bar(width = 0.5, color = "black") +
  geom_text(aes(label =   after_stat(count)), stat = "count", vjust = -0.5) +
  scale_fill_discrete(labels = c("11 anos", "12 anos", "13 anos", "14 anos", "15 anos")) +
  labs(title = "Distribuição dos alunos por Idade",
       x = "Idade",
```

```
       y = "Frequência",
       fill = "Idades dos Alunos") +
  theme_minimal() +
  theme(axis.text.x = element_blank()) +
  ylim(0, max(table(estudantes$Idade)) + 5)


# No geom_bar podemos usar vários argumentos que vão ajustar o gráfico de barras

# Distribuição de Presenças por Sexo (empilhadas)
ggplot(estudantes, aes(x = factor(Presencas), fill = factor(Sexo))) +
  geom_bar(position = "stack", width = 0.5) +
  labs(title = "Distribuição de Frequência por Sexo",
       x = "Frequência",
       y = "Contagem",
       fill = "Sexo")

# Distribuição de Presenças por Sexo (justapostas)
ggplot(estudantes, aes(x = factor(Presencas), fill = factor(Sexo))) +
  geom_bar(position = "dodge", width = 0.5) +
  labs(title = "Distribuição de Frequência por Sexo",
       x = "Frequência",
       y = "Contagem",
       fill = "Sexo")

# Proporção de Presenças por Sexo (proporcionais)
ggplot(estudantes, aes(x = factor(Presencas), fill = factor(Sexo))) +
  geom_bar(position = "fill", width = 0.5) +
  labs(title = "Proporção de Frequência por Sexo",
       x = "Frequência",
       y = "Proporção",
       fill = "Sexo")

# #####
# # Gráfico de setores: geom_bar + coord_polar
# #####
#
# # O gráfico de setores é um gráfico de barras com coordenadas polares.
# # Então repetimos os parâmetros.
# # A coordenada polar deve ser em relação a altura do grafico de barras (contagem). Por isso é usado o y
 # fill: define a cor do gráfico - quando você pinta por categoria, diferenciando as cores, o ggplot gera legenda
 # aplicar a cor em cada categoria, mapeando a váriavel - aes
ggplot(data = estudantes, aes(x = ""), fill = factor(Serie)) +
   geom_bar(width = 1) +
   coord_polar(theta = "y") +
   theme_void()

ggplot(data = estudantes, aes(x = ""), fill = factor(Serie)) +
  geom_bar(width = 1) +
```

```
  coord_polar(theta = "y") +
  labs(title = "Distribuição de Alunos por Série Escolar", x = NULL, y = NULL) +
  theme_void()

# # Aplicar uma cor só no gráfico não faz sentido, como vimos anteriormente
# # fill + scale_fill_manual: escolher uma paleta de cores de forma manual
# nesse caso, ja adaptamos a legenda junto com as cores para ficar mais facil
ggplot(data = estudantes, aes(x = "", fill = factor(Serie))) +
  geom_bar(width = 1) +
  coord_polar(theta = "y") +
  scale_fill_manual(values = c("6 ano" = "red", "7 ano" = "blue", "8 ano" = "green")) +
  labs(title = "Distribuição de Alunos por Série Escolar", x = NULL, y = NULL) +
  theme_void()

# # fill + scale_fill_brewer: escolher uma paleta de cores pré-definida com o pacote RColorBrewer (ver ?RColorBrewer)
ggplot(data = estudantes, aes(x = "", fill = factor(Serie))) +
  geom_bar(width = 1) +
  coord_polar(theta = "y") +
   scale_fill_brewer(palette = "Set2") +
  labs(title = "Distribuição de Alunos por Série Escolar", x = NULL, y = NULL) +
  theme_void()
#
# # color: define as bordas das barras
ggplot(data = estudantes, aes(x = "", fill = factor(Serie))) +
  geom_bar(width = 1, color = "black") +
  coord_polar(theta = "y") +
  scale_fill_brewer(palette = "Set2") +
  labs(title = "Distribuição de Alunos por Série Escolar", x = NULL, y = NULL) +
  theme_void()


# # labs (fill = "Titulo da Legenga")
ggplot(data = estudantes, aes(x = "", fill = factor(Serie))) +
  geom_bar(width = 1, color = "black") +
  coord_polar(theta = "y") +
  scale_fill_brewer(palette = "Set2") +
  labs(title = "Distribuição de Alunos por Série Escolar", x = NULL, y = NULL, fill = "Série Escolar") +
  theme_void()

#
# # geom_text: Adicionar o valor das frequências e outros textos
# # aes(label =   after_stat(count)): substitui pelos valores da contagem
# # stat: qual é o tipo de método estatistico aplicando (nesse caso contagem - count)
# # vjust: define a posição (por padrão o valor fica na metade da linha superior do gráfico)
#
ggplot(estudantes, aes(x = "", fill = factor(Serie))) +
  geom_bar(width = 1, color = "white") +
  coord_polar(theta = "y") +
  scale_fill_brewer(palette = "Set2") +
```

```
  geom_text(
    aes(label = after_stat(count)),
    stat = "count",
    position = position_stack(vjust = 0.5),
    color = "black"
  ) +
  labs(
    title = "Distribuição de Alunos por Série Escolar",
    fill = "Série Escolar",
    x = NULL, y = NULL
  ) +
  theme_void()



# # Mostrando a porcentagem
ggplot(estudantes, aes(x = "", fill = factor(Serie))) +
  geom_bar(width = 1, color = "white") +
  coord_polar(theta = "y") +
  scale_fill_brewer(palette = "Set2") +
  geom_text(
    aes(label = paste0(round(..count.. / sum(..count..) * 100, 1), "%")),
    stat = "count",
    position = position_stack(vjust = 0.5),
    color = "black"
  ) +
  labs(
    title = "Distribuição de Alunos por Série Escolar",
    fill = "Série Escolar",
    x = NULL, y = NULL
  ) +
  theme_void()


#####
# Histograma: geom_histogram
#####

# O histograma depende dos intervalos.
# geom_histogram(binwidth = <largura_intervalo> ou bins = <quant_intervalos>)
ggplot(data = estudantes, aes(x = Nota_em_Matematica)) +
  geom_histogram(binwidth = 2)

ggplot(data = estudantes, aes(x = Nota_em_Matematica)) +
  geom_histogram(bins = 9)

# Inserindo rótulos
ggplot(data = estudantes, aes(x = Nota_em_Matematica)) +
  geom_histogram(bins = 9) +
```

```
  labs(title = "Distribuição das Notas em Matemática",
       x = "Nota em Matemática",
       y = "Frequência")

# Deixando o gráfico visualmente mais leve (theme_minimal deixo o eixo cartesiano mais simples, theme_void deixa tudo em b
ggplot(data = estudantes, aes(x = Nota_em_Matematica)) +
  geom_histogram(bins = 9) +
  labs(title = "Distribuição das Notas em Matemática",
       x = "Nota em Matemática",
       y = "Frequência") +
  theme_minimal()

# fill: define a cor do gráfico

# aplicar a cor geral no gráfico - geom_histogram
ggplot(data = estudantes, aes(x = Nota_em_Matematica)) +
  geom_histogram(bins = 9, fill = "blue") +
  labs(title = "Distribuição das Notas em Matemática",
       x = "Nota em Matemática",
       y = "Frequência") +
  theme_minimal()

# scale_fill_gradient: aplicar uma cor gradiente pela quantidade de elementos em cada intervalo
ggplot(data = estudantes, aes(x = Nota_em_Matematica)) +
  geom_histogram(bins = 9, aes(fill =after_stat(count))) +
  scale_fill_gradient(low = "blue", high = "red") +
  labs(title = "Distribuição das Notas em Matemática",
       x = "Nota em Matemática",
       y = "Frequência") +
  theme_minimal()



# color: define as bordas das barras
ggplot(data = estudantes, aes(x = Nota_em_Matematica)) +
  geom_histogram(bins = 9, fill = "blue", color = "black") +
  labs(title = "Distribuição das Notas em Matemática",
       x = "Nota em Matemática",
       y = "Frequência") +
  theme_minimal()

ggplot(data = estudantes, aes(x = Nota_em_Matematica)) +
  geom_histogram(bins = 9, aes(fill =after_stat(count)), color = "black") +
  scale_fill_gradient(low = "blue", high = "red") +
  labs(title = "Distribuição das Notas em Matemática",
       x = "Nota em Matemática",
       y = "Frequência") +
  theme_minimal()

# legenda: no caso do histograma faz sentido apenas para a coloração em gradiente
```

```
ggplot(data = estudantes, aes(x = Nota_em_Matematica)) +
  geom_histogram(bins = 9, aes(fill =after_stat(count)), color = "black") +
  scale_fill_gradient(low = "blue", high = "red", name = "Frequência") +
  labs(title = "Distribuição das Notas em Matemática",
       x = "Nota em Matemática",
       y = "Frequência") +
  theme_minimal()


#####
# GRÁFICO DE LINHAS: geom_line
#####
# O gráfico de linhas precisa de x e y

ggplot(data = estudantes_tarefas, aes(x = Semana, y = Nota)) +
  geom_line()

# Inserindo rótulos
ggplot(data = estudantes_tarefas, aes(x = Semana, y = Nota)) +
  geom_line()  +
  labs(title = "Distribuição de Notas de Tarefas por Semana",
       x = "Semana",
       y = "Média das Notas")

# Deixando o gráfico visualmente mais leve (theme_minimal deixo o eixo cartesiano mais simples, theme_void deixa tudo em b
ggplot(data = estudantes_tarefas, aes(x = Semana, y = Nota)) +
  geom_line()  +
  labs(title = "Distribuição de Notas de Tarefas por Semana",
       x = "Semana",
       y = "Média das Notas") +
  theme_minimal()

# Definindo limites de x e y:
ggplot(data = estudantes_tarefas, aes(x = Semana, y = Nota)) +
  geom_line()  +
  labs(title = "Distribuição de Notas de Tarefas por Semana",
       x = "Semana",
       y = "Média das Notas") +
  coord_cartesian(ylim = c(min(estudantes_tarefas$Nota) -1, max(estudantes_tarefas$Nota) + 1)) +
  theme_minimal()


# color: define a cor da linha
ggplot(data = estudantes_tarefas, aes(x = Semana, y = Nota)) +
  geom_line(color = "red")  +
  labs(title = "Distribuição de Notas de Tarefas por Semana",
       x = "Semana",
       y = "Média das Notas") +
  coord_cartesian(ylim = c(min(estudantes_tarefas$Nota) -1, max(estudantes_tarefas$Nota) + 1)) +
```

```
  theme_minimal()

# size: define a largura da linha
ggplot(data = estudantes_tarefas, aes(x = Semana, y = Nota)) +
  geom_line(color = "red", size = 1)  +
  labs(title = "Distribuição de Notas de Tarefas por Semana",
       x = "Semana",
       y = "Média das Notas") +
  coord_cartesian(ylim = c(min(estudantes_tarefas$Nota) -1, max(estudantes_tarefas$Nota) + 1)) +
  theme_minimal()

# linetype: define o tipo de linha
# 0 = blank (em branco), 1 = solid (solida), 2 = dashed (tracejada), 3 = dotted (pontilhada),
# 4 = dotdash (ponto e traço), 5 = longdash (traço longo), 6 = twodash (dois traços)
ggplot(data = estudantes_tarefas, aes(x = Semana, y = Nota)) +
  geom_line(color = "red", size = 1, linetype = "dotted")  +
  labs(title = "Distribuição de Notas de Tarefas por Semana",
       x = "Semana",
       y = "Média das Notas") +
  coord_cartesian(ylim = c(min(estudantes_tarefas$Nota) -1, max(estudantes_tarefas$Nota) + 1)) +
  theme_minimal()

# Adicionando linhas de referencia no grid
# geom_hline é uma linha horizontal, utilizando os outros parâmetros ja conhecidos de linhas
ggplot(data = estudantes_tarefas, aes(x = Semana, y = Nota)) +
  geom_line(color = "black") +
  geom_hline(yintercept = min(estudantes_tarefas$Nota),
             color = "red", linetype = "dashed", size = 1) +
  geom_hline(yintercept = mean(estudantes_tarefas$Nota),
             color = "green", linetype = "dashed", size = 1) +
  geom_hline(yintercept = max(estudantes_tarefas$Nota),
             color = "blue", linetype = "dashed", size = 1) +
  labs(title = "Distribuição de Notas de Tarefas por Semana",
       x = "Semana",
       y = "Média das Notas") +
  coord_cartesian(ylim = c(min(estudantes_tarefas$Nota) -1, max(estudantes_tarefas$Nota) + 1)) +
  theme_minimal()

ggplot(data = estudantes_tarefas, aes(x = Semana, y = Nota)) +
  geom_line(color = "black") +
  geom_hline(yintercept = 7,
             color = "red", linetype = "dashed", size = 1) +
  labs(title = "Distribuição de Notas de Tarefas por Semana",
       x = "Semana",
       y = "Média das Notas") +
  coord_cartesian(ylim = c(min(estudantes_tarefas$Nota) -1, max(estudantes_tarefas$Nota) + 1)) +
  theme_minimal()

#####
```

```
# Gráfico  de dispersão: geom_point
#####
ggplot(data = estudantes, aes(x = Horas_de_Estudo, y = Nota_em_Matematica)) +
  geom_point()

# Inserindo rótulos
ggplot(data = estudantes, aes(x = Horas_de_Estudo, y = Nota_em_Matematica)) +
  geom_point() +
  labs(title = "Gráfico de Dispersão: Horas de Estudo vs. Nota em Matemática",
       x = "Horas de Estudo",
       y = "Notas em Matemática")

# Deixando o gráfico visualmente mais leve
ggplot(data = estudantes, aes(x = Horas_de_Estudo, y = Nota_em_Matematica)) +
  geom_point() +
  labs(title = "Gráfico de Dispersão: Horas de Estudo vs. Nota em Matemática",
       x = "Horas de Estudo",
       y = "Notas em Matemática") +
  theme_minimal()

# xlim e ylim: define o limite para os eixos x e y (inicio, fim)
ggplot(data = estudantes, aes(x = Horas_de_Estudo, y = Nota_em_Matematica)) +
  geom_point() +
  labs(title = "Gráfico de Dispersão: Horas de Estudo vs. Nota em Matemática",
       x = "Horas de Estudo",
       y = "Notas em Matemática") +
  coord_cartesian(xlim = c(min(estudantes$Horas_de_Estudo), 5), ylim = c(0, 10)) +
  theme_minimal()

# color: define a cor dos pontos
ggplot(data = estudantes, aes(x = Horas_de_Estudo, y = Nota_em_Matematica)) +
  geom_point(color = "red") +
  labs(title = "Gráfico de Dispersão: Horas de Estudo vs. Nota em Matemática",
       x = "Horas de Estudo",
       y = "Notas em Matemática") +
  coord_cartesian(xlim = c(min(estudantes$Horas_de_Estudo), 5), ylim = c(0, 10)) +
  theme_minimal()

# shape: define o simbolo usado para os pontos (pode ser letras)

ggplot(data = estudantes, aes(x = Horas_de_Estudo, y = Nota_em_Matematica)) +
  geom_point(color = "red", shape = 7) +
  labs(title = "Gráfico de Dispersão: Horas de Estudo vs. Nota em Matemática",
       x = "Horas de Estudo",
       y = "Notas em Matemática") +
  coord_cartesian(xlim = c(min(estudantes$Horas_de_Estudo), 5), ylim = c(0, 10)) +
  theme_minimal()
```

```
#size: define o tamanho do simbolo
ggplot(data = estudantes, aes(x = Horas_de_Estudo, y = Nota_em_Matematica)) +
  geom_point(color = "red", shape = 7, size = 3) +
  labs(title = "Gráfico de Dispersão: Horas de Estudo vs. Nota em Matemática",
       x = "Horas de Estudo",
       y = "Notas em Matemática") +
  coord_cartesian(xlim = c(min(estudantes$Horas_de_Estudo), 5), ylim = c(0, 10)) +
  theme_minimal()

# Adicionando linhas de referencia no grid (linha de regressão)
# method = lm (linear method - regressao linear).
# se: áre de incerteza da linha - FALSE PARA TIRAR
ggplot(data = estudantes, aes(x = Horas_de_Estudo, y = Nota_em_Matematica)) +
  geom_point(color = "red", shape = 7, size = 3) +
  geom_smooth(method = "lm", color = "black", se = FALSE) +
  labs(title = "Gráfico de Dispersão: Horas de Estudo vs. Nota em Matemática",
       x = "Horas de Estudo",
       y = "Notas em Matemática") +
  coord_cartesian(xlim = c(min(estudantes$Horas_de_Estudo), 5), ylim = c(0, 10)) +
  theme_minimal()

ggplot(data = estudantes, aes(x = Horas_de_Estudo, y = Nota_em_Matematica)) +
  geom_point(color = "red", shape = 7, size = 3) +
  geom_smooth(method = "lm", color = "black", se =TRUE) +
  labs(title = "Gráfico de Dispersão: Horas de Estudo vs. Nota em Matemática",
       x = "Horas de Estudo",
       y = "Notas em Matemática") +
  coord_cartesian(xlim = c(min(estudantes$Horas_de_Estudo), 5), ylim = c(0, 10)) +
  theme_minimal()
```

APÊNDICE B — BASES DE DADOS UTILIZADAS NOS EXEMPLOS

As bases de dados utilizadas ao longo dos exemplos estão disponibilizadas em formato `.csv` na pasta `dados/` do repositório no GitHub: https://github.com/isasterza/manual-graficos-rstudio/tree/main (Butzen; Marins; Nava, 2026). Este apêndice apresenta a descrição dos arquivos, um dicionário resumido de variáveis e uma amostra das bases, de modo a apoiar a reprodutibilidade.

Apêndice B.1 — Arquivos disponibilizados

- `dados/estudantes.csv`: base principal utilizada nos exemplos de gráficos (barras, setores, histograma e dispersão).
- `dados/estudantes_tarefas_media.csv`: base utilizada nos exemplos de gráfico de linhas (nota por semana).

Apêndice B.2 — Dicionário de variáveis (`estudantes.csv`)

Variável	Descrição
Nome	Identificador do estudante (rótulo didático; ex.: Estudante_1)
Serie	Série escolar
Idade	Idade (anos)
Horas_de_Estudo	Horas de estudo
Nota_em_Matematica	Nota em Matemática
Extracurricular	Atividade extracurricular (categoria; ex.: Teatro, Música, Esportes, Nenhum)
Presencas	Quantidade de presenças (contagem no período considerado)
Sexo	Categoria de sexo
Musica_Favorita	Preferência musical (categoria)
Distancia_Escola_km	Distância até a escola (km)

Apêndice B.3 — Dicionário de variáveis (`estudantes_tarefas_media.csv`)

Variável	Descrição
Semana	Identificador da semana
Nota	Média das notas de tarefas na semana

Apêndice B.4 — Amostra das bases (primeiras linhas)

A seguir apresenta-se uma amostra das bases utilizadas nos exemplos para referência e verificação de estrutura. A versão integral encontra-se no repositório do GitHub. Conforme ilustrado nas Figuras 17 e 18, a base `estudantes.csv` contém variáveis categóricas e numéricas, enquanto a base `estudantes_tarefas_media.csv` organiza a média de notas por semana.

Nome	Serie	Idade	Horas_de_Estudo	Nota_em_Matematica	Extracurricular	Presencas	Sexo	Musica_Favorita	Distancia_Escola_km
Estudante_1	6 ano	11	3.0	7.8	Teatro	22	Masculino	Rock	4.1
Estudante_2	7 ano	11	4.0	7.0	Esportes	22	Feminino	Pop	9.5
Estudante_3	6 ano	11	5.0	8.2	Música	18	Masculino	Clássica	7.5
Estudante_4	7 ano	15	3.8	5.5	Música	20	Feminino	Clássica	6.2
Estudante_5	8 ano	11	2.7	4.2	Nenhum	18	Feminino	Sertanejo	2.0
Estudante_6	7 ano	12	5.0	6.6	Esportes	20	Masculino	Eletrônica	2.0
Estudante_7	6 ano	13	5.0	8.3	Teatro	20	Masculino	Eletrônica	1.1
Estudante_8	6 ano	14	5.0	6.7	Teatro	18	Feminino	Clássica	8.7
Estudante_9	7 ano	11	3.4	6.5	Música	18	Feminino	Pop	6.2
Estudante_10	7 ano	11	2.8	9.5	Teatro	20	Feminino	Sertanejo	7.2

Figura 17 – Amostra da base `estudantes.csv` (primeiras linhas).

Semana	Nota
1	6.9
2	6.7
3	7.25
4	7.25
5	6.9
6	7.65
7	7.45
8	8.1
9	7.25
10	6.85

Figura 18 – Base `estudantes_tarefas_media.csv` (todas as linhas).