INTRODUÇÃO AO R E ANALYTICS

APLICAÇÕES EM ECONOMIA

STEFAN W. D'AMATO - WALLACE MARCELINO PEREIRA

Universidade Federal do Pará
Instituto de Ciências Sociais Aplicadas - ICSA
Faculdade de Ciências Econômicas - FACECON
Programa de Pós-Graduação em Economia – PPGE
Laboratório de Pesquisa em Política Econômica e Desenvolvimento Produtivo - LAPED

Stefan W. D'Amato
Wallace Marcelino Pereira

Introdução ao R e Analytics

Aplicações em Economia

Série Trilhas do Conhecimento – Volume 1

Belém
2025

Imagem da capa obtida sob licença do banco de imagens Shutterstock

Revisão: Stefan W. D'Amato, Wallace Marcelino Pereira

Diagramação: Raquel Botelho

Editora ICSA
Rua Augusto Correa, 01
Cidade Universitária José da Silveira Netto
Instituto de Ciências Sociais Aplicadas
Setor Profissional, Guamá
CEP 66.075-110, Belém, Pará, Brasil
Secretaria: 55+ 91 32017101

Dados Internacionais de Catalogação na
Fonte (CIP) Luiz Otavio Maciel da Silva
Bibliotecário – CRB-2/771

D155 D'Amato, Stefan Wilson
Introdução ao R a analytics [recurso eletrônico]: aplicações em economia / Stevan Wilson D'Amato, Wallace Marcelino Pereira. – Dados eletrônicos. – Belém : ICSA, 2025.

Modo de acesso: Word Wide Web.
ISBN: 978-85-61214-43-2.

1. Economia. 2. Analytics. 3. Estatística. 5. Ciência de dados. I. Pereira, Wallace Marcelino. II. Título.

CDD: 330

Introdução ao R e Analytics

Aplicações em Economia

Série Trilhas do Conhecimento – Volume 1

PREFÁCIO

Esta apostila nasceu das nossas experiências em sala de aula e da convivência com um desafio comum: a curva de aprendizado do R. Embora seja uma linguagem poderosa e gratuita, o primeiro contato pode parecer intimidador — especialmente para quem vem de áreas aplicadas.

A proposta aqui é descomplicar. Ao longo dos tópicos, buscamos apresentar o R de forma progressiva, clara e aplicada, mostrando que é possível construir análises robustas com comandos simples e bem compreendidos. Cada seção foi pensada para reforçar a autonomia do leitor, valorizando a intuição por trás dos procedimentos e evitando o uso automático de comandos sem propósito.

Mais do que ensinar a linguagem, esta apostila pretende ajudar o leitor a se sentir confortável com ela — e, quem sabe, até gostar do processo. Se ao final desta leitura o R deixar de ser um obstáculo e passar a ser uma ferramenta útil no seu cotidiano, teremos alcançado nosso objetivo.

AGRADECIMENTOS

STEFAN W. D'AMATO

Quero expressar minha profunda gratidão a todos que contribuíram direta ou indiretamente para a realização deste trabalho.

À CAPES, pela fundamental apoio financeiro e pela valorização da pesquisa e da pós-graduação no Brasil.

Ao Programa de Pós-Graduação em Economia da UFPA (PPGE/UFPA), pela excelência acadêmica e pelo espaço de construção do conhecimento. Aos meus mestres e orientadores, em especial na figura do Prof. Wallace M. Pereira, cujas orientações, sugestões e incentivos foram essenciais em minha trajetória.

De modo especial, dedico este trabalho à minha família, alicerce de minha vida e motivação diária. À minha esposa, Vanessa, coração do nosso lar, pelo amor, paciência e apoio em todos os momentos. E aos meus filhos, que são minha maior inspiração e razão para seguir em frente.

Que este trabalho reflita não apenas um esforço intelectual, mas também o afeto e a colaboração de todos que caminharam comigo nessa jornada.

WALLACE M. PEREIRA

Expresso aqui minha gratidão à Universidade Federal do Pará - UFPA, e em especial ao Instituto de Ciências Sociais Aplicadas – ICSA, na pessoa do Prof. Dr. Armando Lírio de Souza, à Faculdade de Ciências Econômicas – FACECON e ao Programa de Pós-Graduação em Economia - PPGE.

Desde minha chegada à cidade de Belém tenho aprendido muito. Essa cultura fantástica e o suporte ao desenvolvimento de minha pesquisa tem contribuído para que eu possa pensar e repensar a teoria e prática enquanto professor.

Contribuir para a formação dos alunos está para além da sala de aula. É pensar no longo prazo e tentar ajudar para que sonhos possam ser realizados, apesar das adversidades da vida. Esta publicação é uma pequena contribuição para quebrar barreiras e criar pontes para o futuro dos jovens paraenses.

Por fim, de modo especial, dedico este trabalho aos meus pais, Antônio Luciano (in memoriam) e Marta e à minha esposa Erodiana.

Nulla tenaci invia est via

INTRODUÇÃO: R E ANALYTICS NAS CIÊNCIAS ECONÔMICAS

A crescente complexidade dos fenômenos econômicos, aliada à ampla disponibilidade de dados, tornou essencial o uso de ferramentas analíticas robustas na pesquisa empírica e na formulação de políticas. Nesse cenário, a linguagem **R** e os métodos de **Analytics** têm se tornado cada vez mais relevantes na formação e na prática profissional de economistas.

O R é uma linguagem de programação **gratuita e open source**, voltada para estatística, modelagem e ciência de dados. Amplamente utilizada nas ciências econômicas, permite manipular grandes volumes de dados, aplicar modelos econométricos e gerar visualizações de alta qualidade. Sua força está também em sua comunidade ativa: espaços como o fórum **RStudio Community** (https://forum.posit.co/) e o site de perguntas e respostas **Stack Overflow**, na seção de R (www.stackoverflow.com/questions/tagged/r), reúnem milhares de usuários que compartilham soluções, tutoriais e boas práticas, facilitando o aprendizado contínuo.

Nesta apostila, a introdução a **Analytics** será focada em seus fundamentos aplicados à Economia, com ênfase na **estatística descritiva** e na **visualização de dados**. Essas ferramentas são essenciais para explorar bases de dados, identificar padrões e comunicar evidências de forma clara e eficaz.

A integração entre R e Analytics representa, portanto, uma competência central para o economista contemporâneo: pensar os dados como parte do raciocínio econômico.

SUMÁRIO

INSTALAÇÃO E PRIMEIROS PASSOS

O primeiro passo para trabalhar com R e aplicar métodos analíticos em Economia é garantir que o ambiente esteja corretamente instalado em seu computador. Esta seção apresenta o processo de instalação do R e do RStudio, além de uma introdução às principais funcionalidades da interface.

➷ Instalando o R

Para instalá-lo:

1. Acesse o site oficial do projeto R:
https://www.r-project.org/

2. Clique em "CRAN" (Comprehensive R Archive Network) e escolha um espelho (mirror) próximo da sua localização (Brasil).

3. Selecione seu sistema operacional (Windows, macOS ou Linux) e baixe o instalador correspondente.

4. Siga os passos do instalador padrão.

📣 **Importante:** O R é gratuito, open source e de rápida instalação.

➷ Instalando o RStudio

O RStudio é um ambiente de desenvolvimento integrado (IDE) que facilita o uso do R com uma interface amigável e recursos adicionais como editor de scripts, visualização de gráficos e painel de variáveis.

Para instalar:

1. Acesse:
https://posit.co/download/rstudio-desktop/

2. Baixe a versão gratuita do RStudio Desktop.

3. Instale normalmente em seu computador.

 O RStudio depende do R. Portanto, certifique-se de instalar o R antes de instalar o RStudio.

1. EXPLORANDO A INTERFACE DO RSTUDIO

Após a instalação, abra o RStudio. A interface é dividida em quatro painéis principais:

- **Console (Inferior esquerdo):** onde os comandos são digitados e executados diretamente.
- **Script Editor (Superior esquerdo):** espaço para escrever, salvar e executar scripts de código.
- **Environment / History (Superior direito):** mostra os objetos ativos (variáveis, data frames e etc.) e o histórico de comandos.
- **Files / Plots / Packages / Help (Inferior direito):** permite navegar pelas pastas, visualizar gráficos gerados, gerenciar pacotes instalados e acessar a ajuda.

1.1 CRIANDO E SALVANDO SCRIPTS

Definir diretório de trabalho[1]:

```
setwd("C:/Users/Aluno/Documents/MeuProjetoR")
```

OBS: Altere o caminho acima para a pasta onde estão seus arquivos.

Atenção ao uso de barra / **ou** \\ (use sempre / no R, mesmo no Windows).

Verificar o diretório atual:

```
getwd()
```

Ao trabalhar com análises mais longas, é recomendável escrever o código em um script (arquivo .R) em vez de digitar tudo no Console. Para isso:

1. Clique em **File > New File > R Script.**

1. Os trechos apresentados na cor laranja ao longo do material correspondem a comandos e códigos escritos na linguagem R.

2. Digite os comandos no editor de texto (painel superior esquerdo).

3. Para executar um trecho de código, selecione-o e pressione **Ctrl + Enter.**

4. Para salvar o script, clique em **File > Save As...** e escolha o local desejado (caso não tenha definido o diretório de trabalho).

Importante: Organizar seu código em scripts facilita a reprodutibilidade das análises e a documentação do processo de trabalho.

Baixe o script acessando o QR Cod abaixo

2.FUNDAMENTOS DA LINGUAGEM R²

---

2. Como pedir ajuda ao R:
1. Ajuda sobre uma função específica:
?nome_da_funcao ou help(nome_da_funcao)
2. Buscar função por palavra-chave (quando você não lembra o nome exato):
help.search("palavra") ou ??palavra
3. Ajuda sobre pacotes:
help(package = "nome_do_pacote")
4.Obter estrutura e argumentos de uma função:
args(nome_da_funcao)

Antes de avançar para análises estatísticas e gráficos, é essencial compreender a estrutura básica da linguagem R. Nesta seção, apresentamos os principais tipos de dados, estruturas fundamentais e operações iniciais que compõem a base do trabalho com R.

2.1 TIPOS DE DADOS EM R

O R reconhece diferentes **tipos de dados**, e saber identificá-los é essencial para evitar erros e aplicar funções corretamente. Os principais tipos são:

- **Numérico:** números com ou sem casas decimais (ex: 2, 3.14)
- **Inteiro:** números inteiros declarados com L (ex: 10L)
- **Caractere (texto):** palavras ou frases entre aspas (ex: “PIB”)
- **Lógico:** valores booleanos (TRUE ou FALSE)
- **Fator:** categorias ordenadas ou não ordenadas (ex: sexo = c(“M”, “F”) transformado em fator)
- **Data:** datas no formato padrão (as.Date(“2024-01-01”)) – Ano/Mês/dia

2.2 CRIANDO OBJETOS

Em R, qualquer valor pode ser armazenado em um **objeto**, atribuindo-se um nome a ele:

```
x <- 5

nome <- “Economia”

crescimento <- TRUE
```

Importante: A seta <- é o símbolo mais comum para atribuição, mas o sinal = também pode ser usado.

Estruturas básicas de dados

O R trabalha com diferentes **estruturas de dados**. As mais utilizadas são:

- **Vetores:** sequência de elementos do mesmo tipo

```
v <- c(1, 2, 3, 4)
```

- **Matrizes:** tabelas com linhas e colunas, com elementos do mesmo tipo

```
m <- matrix(1:6, nrow = 2, ncol = 3)
```

- **Listas:** coleção de objetos de tipos diferentes

```
lista <- list(nome = "PIB", valor = 2034.5, crescimento = TRUE)
```

- **Data frames:** estruturas semelhantes a planilhas, ideais para bases de dados

```
df <- data.frame(ano = c(2020, 2021), pib = c(7.5, 4.6))
```

2.3 OPERAÇÕES BÁSICAS

O R permite realizar operações matemáticas simples com facilidade:

```
a <- 10
b <- 2
a + b  # soma[3]
a - b  # subtração
a * b  # multiplicação
a / b  # divisão
a^b    # potência
```

3. Os textos sucedidos pelo símbolo # corresponde aos comentários no código. Eles não são executados pelo R.

Também é possível aplicar funções como:

```
log(a)   # logaritmo
sqrt(a)  # raiz quadrada
round(pi, 2)   # arredondamento, sendo o termo 2
referente às casas decimais
sum(v)   #soma os elementos de um vetor, utilizamos o
vetor “v” que calculamos anteriormente
length(v)   # conta o número de elementos
min(v) ou max(v)   # valor mínimo ou máximo do vetor
```

 **Dica:** O R é uma linguagem extremamente rica e em constante evolução. Existem milhares de funções disponíveis — e sempre há algo novo para aprender. Quando surgir uma dúvida ou curiosidade, explore fóruns e sites especializados. Essas comunidades são ótimos espaços para aprender com exemplos reais, tirar dúvidas e descobrir novas formas de aplicar o R na prática.

3. INDEXAÇÃO: ACESSANDO ELEMENTOS

Em R, a **indexação** é o processo de acessar partes específicas de um objeto, como um vetor, lista ou data frame. Isso permite selecionar, modificar ou analisar apenas os elementos desejados da base de dados.

3.1 ACESSANDO COLUNAS DE UM DATA FRAME

Um dos usos mais comuns da indexação é acessar **colunas de uma base de dados** (um data.frame) por meio do operador $, ou com colchetes [].

Exemplo: modelo simples de demanda agregada

Vamos construir uma pequena base de dados com variáveis macroeconômicas típicas de um modelo de Keynes (mas, na "vida real" seria diferente, buscaríamos dados em bases confiáveis):

```
dados <- data.frame(
  PIB = c(1000, 1050, 1100, 1150),
  Consumo = c(700, 735, 770, 805),
  Investimento = c(150, 160, 170, 180),
  Gasto = c(100, 110, 115, 120),
  Exportacoes_Liquidas = c(50, 45, 45, 45)
)
```

3.2 ACESSANDO VARIÁVEIS (COLUNAS) ESPECÍFICAS:

- Usando o operador $:

```
dados$PIB
```

OBS: Esse comando retorna os valores da coluna **PIB.**

- Usando colchetes:

```
dados[ , "PIB"]
ou
dados[ , 1]   # primeira coluna
```

3.3 ACESSANDO ELEMENTOS ESPECÍFICOS (LINHA E COLUNA):

- Valor de Consumo na terceira linha:

```
dados[3, “Consumo”]
ou
dados[3, 2]
```

Dica: dados[linha, coluna] é a estrutura geral para acessar elementos em tabelas. Se quiser acessar só a linha ou só a coluna, basta deixar um dos lados em branco.

3.4 OPERADORES LÓGICOS E COMPARAÇÕES

Além das operações matemáticas básicas, o R permite realizar **comparações lógicas**, fundamentais para filtrar dados, testar condições e construir estruturas de decisão.

Veja alguns operadores lógicos:

```
x <- 10
x > 5   # Maior que
x < 5   # Menor que
x == 10   # Igual a
x != 7    # Diferente de
x <= 12   # Menor ou igual a
x >= 12   # Maior ou igual a
```

Dica: Para verificar igualdade no R, sempre use == (com dois sinais de igual).

Esses operadores também podem ser aplicados a vetores ou colunas de um data frame. Por exemplo, usando a base dados criada anteriormente:

```
dados$PIB > 1050
```

Essas comparações serão úteis quando trabalharmos com filtros de dados, seleções condicionais e análises estatísticas segmentadas.

3.5 ESTRUTURAS DE CONTROLE: IF, ELSE, FOR

Embora não sejam o foco principal desta apostila, é útil conhecer os comandos básicos de controle, que permitem automatizar verificações e criar pequenos fluxos de decisão.

- **Condicional simples "if":**

```
x <- 100

if (x > 90) {
      print("Crescimento alto")
      } else {
       print("Crescimento moderado")
}
```

- **Laço de repetição "for":**

```
      valores <- c(1, 2, 3)

      for (v in valores) {
            print(v * 2)
      }
```

Essas estruturas serão especialmente úteis em análises mais avançadas, simulações e automatizações.

Vamos exemplificar com uma equação linear bem conhecida na economia, a função consumo keynesiana, do tipo:

$$C_i = a + bY_i \qquad (1)$$

Onde C_i representa o consumo, *a* é o consumo autônomo, *b* é a propensão marginal a consumir, e Y_i é a renda.

```
# Parâmetros da função consumo
a <- 100       # consumo autônomo
b <- 0.8       # propensão marginal a
consumir

# Vetor com diferentes níveis de renda
renda <- c(500, 600, 700, 800, 900)

# Laço “for” para calcular e imprimir o
consumo correspondente a cada renda
for (Y in renda) {
    C <- a + b * Y
    print(paste(“Para Y =”, Y, “→Consumo C
=”, C))
}
```

OBS: A função paste() concatena (ou seja, junta) diferentes elementos — números, textos ou variáveis — em uma única string de texto. O resultado é uma frase formatada, que facilita a interpretação dos resultados no console.

3.6 APLICANDO FUNÇÕES A VETORES: APPLY() E SAPPLY()

Para aplicar uma mesma função a todos os elementos de uma estrutura (vetor, lista, matriz ou data frame), o R oferece funções como apply() e sapply():

- **sapply():** aplica uma função a cada elemento de uma lista ou vetor e retorna um vetor simplificado.

```
valores <- c(100, 200, 300)
sapply(valores, log)
```

- **apply():** usado em **matrizes** ou **data frames**, permite aplicar funções por linha ou coluna.

```
matriz <- matrix(1:9, nrow = 3)
apply(matriz, 1, sum)  # Soma por linha
```

OBS: O **primeiro argumento** é o nome do objeto (matriz)

O **segundo argumento** indica a dimensão:

- 1 = aplicar por **linha**
- 2 = aplicar por **coluna**

O **terceiro argumento** é a função a ser aplicada: sum (soma)

Exemplos de aplicações mais usadas sum(), mean(), sd().

4. PACOTES: INSTALANDO E CARREGANDO RECURSOS ADICIONAIS

Uma das maiores forças do R está em seus **pacotes**, que expandem a linguagem com funções para tarefas específicas — como gráficos, regressões, séries temporais, mapas, entre outros.

- **Instalar um pacote** (apenas uma vez):

```
install.packages("ggplot2")
```

 **Dica:** Sempre escreva o nome do pacote entre "aspas" ao usar a função.

Esquecer das aspas é um dos erros mais comuns entre iniciantes e até mesmo para os mais experientes!

- **Carregar um pacote** (em cada nova sessão):

```
library(ggplot2)
```

 **Dica:** Deixando de carregar o pacote, os comandos não serão executados, gerando sucessivas mensagens de erro!

Alguns pacotes importantes para quem trabalha com dados econômicos são:

- tidyverse: para manipulação e visualização de dados
- readxl: para leitura de planilhas Excel
- sidrar: para acesso direto a dados do IBGE via API
- ggplot2: para gráficos profissionais
- dplyr: para tratamento de bases de dados de forma intuitiva

5. IMPORTAÇÃO E TRATAMENTO DE DADOS

Para realizar qualquer análise no R, o primeiro passo é **importar os dados** para o ambiente de trabalho. Em seguida, é necessário **organizar e tratar** essas informações para deixá-las no formato adequado para cálculo, visualização ou modelagem. Este capítulo apresenta os principais comandos para leitura e manipulação de dados em formatos comuns.

5.1 LENDO ARQUIVOS .CSV

O formato .csv (valores separados por vírgula) é um dos mais utilizados para armazenar bases de dados.

```
dados <- read.csv("caminho/do/arquivo.csv", sep = ",",
header = TRUE)
```

- sep = "," indica que os dados estão separados por vírgula.
- header = TRUE informa que a primeira linha contém os nomes das colunas.

Se os dados estiverem separados por ponto e vírgula (como ocorre em arquivos .csv salvos pelo Excel em português), use:

```
dados <- read.csv2("caminho/do/arquivo.csv")
```

5.2 LENDO ARQUIVOS EXCEL .XLSX

Para ler planilhas do Excel, é necessário instalar o pacote readxl:

```
install.packages("readxl")
library(readxl)
dados <- read_excel("caminho/do/arquivo.xlsx",
sheet = 1)
```

- argumento sheet = 1 define que queremos importar

a **primeira aba** da planilha.

5.3 IMPORTANDO BASES PÚBLICAS: EXEMPLO COM O IBGE (SIDRA)

Com o pacote sidrar, é possível acessar diretamente dados do IBGE pelo código da tabela:

```
install.packages("sidrar")
library(sidrar)
dados <- get_sidra(api = "/t/1419/n1/all/v/63/p/
all/c315/7169")
```

Explicando cada parte da API:

PARTE DO CÓDIGO	SIGNIFICADO
/t/1419	Tabela 1419: IPCA - Índice Nacional de Preços ao Consumidor Amplo
/n1/all	Nível territorial 1: Brasil (n1) — all significa "todos"
/v/63	Variável 63: Variação mensal do índice (%)
/p/all	Período: todos os períodos disponíveis
/c315/7169	Filtro da coluna 315: 7169 = Índice Geral (código da "Subitem")

Dica: Se quiser trazer a série por **regiões ou capitais**, pode trocar /n1/all por:

```
/n2/all → Unidades da federação
/n3/all → Regiões metropolitanas (ex: Belo Horizonte,
São Paulo)
```

Dica: Consulte os códigos das tabelas diretamente no site do SIDRA, é o modo mais descomplicado e sem erros (https://sidra.ibge.gov.br/). O pacote retorna um data. frame pronto para ser tratado.

Passo a Passo

1. Acesse o site do sidra;

2. No campo de busca, digite o número da tabela

Por exemplo:

1419 → IPCA (variação mensal)

6612 → PIB trimestral

1737 → IPCA com estrutura por subitem

Clique no título da **tabela desejada.**

3. Faça suas seleções normalmente

Período: selecione "todos" ou os mais recentes

Variáveis: selecione o que deseja (ex: "Variação mensal do índice")

Unidade geográfica: Brasil, regiões, estados ou RM (como RMBH)

Outros filtros: Subitem, sexo, idade (depende da tabela)

Não clique em "Ok" ainda.

4. Clique no botão "API" no canto superior direito da tela

Esse botão aparece ao lado de "Exportar", "Salvar", etc.

Se você não ver o botão "API", revise suas seleções. Nem toda tabela permite todos os cruzamentos com API.

Copie o link gerado

O link completo será algo como:

`https://apisidra.ibge.gov.br/values/t/1419/n1/all/v/63/p/all/c315/7169`

OBS: se colocar o link no navegador de internet, conseguirá observar os filtros e os valores dos códigos.

Use apenas a parte do link a partir de /t/... no R

No R, você usa:

```
get_sidra(api = "/t/1419/n1/all/v/63/p/all/c315/7169")
```

OBS: igual ao quadro anterior.

5.4 EXPLORANDO OS DADOS IMPORTADOS

Depois de importar os dados, é importante entender a estrutura da base:

```
head(dados)        # mostra as primeiras linhas
str(dados)         # estrutura da base: tipos de variáveis
summary(dados)     # resumo estatístico das colunas numéricas
names(dados)       # nomes das colunas
dim(dados)         # dimensões: número de linhas e colunas
```

5.5 TRATAMENTO DE DADOS COM DPLYR

O pacote **dplyr,** parte do **tidyverse**, permite manipular bases de forma eficiente e com comandos intuitivos. Para usá-lo:

```
install.packages("dplyr")
library(dplyr)
```

Comandos principais:

```
PIBbase <- data.frame(
  Ano = c(2020, 2021, 2022, 2023),
  PIB = c(980, 1050, 1120, 1200),
  Consumo = c(700, 740, 790, 830)
)
########OBS: criamos esse novo data.frame com os 3 vetores
acima.
PIBbase %>%
  select(PIB, Consumo)     # seleciona colunas

PIBbase %>%
  filter(PIB > 1000)       # filtra linhas com condição

PIBbase %>%
  mutate(Novo_PIB = PIB * 1.05)  # cria nova variável

PIBbase %>%
  summarise(media_pib = mean(PIB))  # resumo estatístico
```

💡 **Dica:** O operador %>% (pipe) é usado para encadear operações de forma sequencial. Ele pode ser lido como "então faça".

5.6 CONVERTENDO TIPOS DE VARIÁVEIS

No R, as variáveis podem ter diferentes tipos (ou "classes"), como:

- character (texto)
- numeric (número com decimais)
- integer (número inteiro)

- factor (categoria)
- logical (TRUE ou FALSE)

A seguir, veja exemplos simples para criar vetores, verificar o tipo com str(), e fazer as conversões.

Texto (character) → Número (numeric)	**Número → Texto**

x <- c(“100”, “200”, “300”)
str(x) # character
x <- as.numeric(x)
str(x) #numeric
	y <- c(10, 20, 30)
	str(y) # numeric
	y <- as.character(y)
	str(y) # character

Número → Fator	**Texto → Fator**

z <- c(1, 2, 1, 3)
str(z) # numeric
z <- as.factor(z)
str(z) # factor
	setor <- c(“Indústria”, “Serviços”, “Agro”)
	str(setor) # character
	setor <- as.factor(setor)
	str(setor) # factor

Fator → Texto	**Dica:** Use sempre str() para verificar o tipo da variável. Saber o tipo certo evita erros em funções como mean(), ggplot() e outras.

str(setor) # factor
setor <- as.character(setor)
str(setor) # character	

6. JUNTANDO BASES DE DADOS

Na prática econômica, é comum trabalhar com **mais de uma base de dados,** como PIB, investimento, consumo ou população. Para integrá-las, usamos **junções** com base em uma coluna comum — geralmente o Ano, o município, ou o código do setor.

Vamos criar duas bases simples com a variável Ano como chave de junção:

```
# Base 1: PIB por ano
base_pib <- data.frame(
  Ano = c(2000, 2001, 2002),
  PIB = c(1000, 1100, 1200)
)

# Base 2: Investimento por ano
base_invest <- data.frame(
  Ano = c(2001, 2002, 2003),
  Investimento = c(300, 320, 350)
)
```

Instalar pacotes necessário e carregar

```
install.packages("tidyverse")
library(tidyverse)
```

O pacote **dplyr** oferece quatro funções principais para juntar bases:

- **left_join():** mantém todos os dados da base da esquerda (base_pib):

```
left_join(base_pib, base_invest, by = "Ano")
```

- **right_join():** mantém todos os dados da base da direita (base_invest):

```
right_join(base_pib, base_invest, by = “Ano”)
```

- **inner_join():** mantém apenas os anos que existem nas duas bases:

```
inner_join(base_pib, base_invest, by = “Ano”)
```

- **full_join():** mantém todos os anos de ambas as bases:

```
full_join(base_pib, base_invest, by = “Ano”)
```

Dica: As variáveis usadas na junção devem ter o mesmo nome e tipo em ambas as bases. Se os nomes forem diferentes, use: by = c(“ano_base1” = “ano_base2”)

6.1 ORDENANDO DADOS

Para organizar os dados em ordem crescente ou decrescente:

```
# Ordem crescente de PIB
PIBbase %>% arrange(PIB)

# Ordem decrescente
PIBbase %>% arrange(desc(PIB))
```

6.2 REMOVENDO E RENOMEANDO COLUNAS

```
# Remover coluna
PIBbase <- select(PIBbase, -Consumo) # (-coluna)
indesejada ou a que deseja retirar

# Renomear coluna
PIBbase <- rename(PIBbase, Produto_Interno_Bruto
= PIB) # renomeamos para Produto Interno Bruto
```

6.3 LIDANDO COM VALORES FALTANTES (NA)

O R usa NA para representar **valores ausentes**, ou seja, informações que estão faltando em alguma célula da base. Tratar corretamente esses valores é fundamental para evitar erros em análises, cálculos e modelos.

Vamos utilizar a **base airquality**, que já vem com o R e contém valores faltantes.

```
# Base de exemplo
dados <- airquality

# Ver quais linhas têm NA
is.na(dados)

#Quantidade de NA na base
sum(is.na(dados))

#Quantidade de NA por Coluna
colSums(is.na(dados))

# Remover linhas com qualquer NA
dados <- na.omit(dados)

rm(dados)
dados <- airquality

View(dados)

# Substituir NA por zero ou outro valor
dados$Solar.R[is.na(dados$Solar.R)] <-
0

View(dados)
```

OBS: Deletamos a base com rm(nome_da_base) e adicionamos novamente para conseguir executar a substituição do NA por “0”. View(dados) é utilizado para visualizar a base.

OBS: rm(list = ls()) é utilizada para remove **todos** os objetos e bases carregados no ambiente atual.

7. ESTATÍSTICA DESCRITIVA COM R

A **estatística descritiva** é o primeiro passo para analisar qualquer base de dados. Ela permite **resumir, explorar e interpretar informações** por meio de medidas numéricas e tabelas. No R, temos ferramentas simples e poderosas para calcular médias, desvios, frequências e outras medidas estatísticas.

7.1 RELEMBRANDO ALGUMAS ESTATÍSTICAS DESCRITIVAS

MEDIDA	FÓRMULA OU DEFINIÇÃO MATEMÁTICA
Média $\bar{x}$	$\bar{x} = \sum_{i=1}^{n} \frac{x_i}{n}$
Mediana	n ímpar: $x_{\frac{n+1}{2}}$ n par: $\frac{x_{\frac{n}{2}} + x_{\left(\frac{n}{2}\right)+1}}{2}$
Moda	Valor com maior frequência absoluta na amostra
Desvio padrão (s)	$s = \sqrt{(1/(n-1)) \times \Sigma(x_i - \bar{x})^2}$
Variância (s^2)	$s^2 = (1/(n-1)) \times \Sigma(x_i - \bar{x})^2$
Mínimo e Máximo	Menor e maior valores observados, respectivamente
Amplitude	Amplitude = Máximo – Mínimo
Resumo estatístico	Min, Q1, Mediana, Média, Q3, Max
Quartis	Q_1 = 25º percentil, Q_3 = 75º percentil
Percentil (ex: 90%)	P_{90} = valor tal que 90% dos dados estão abaixo dele

Criando a base de exemplo

```
PIBbase3 <- data.frame(
  Ano = c(2000, 2001, 2002, 2003, 2004),
  PIB = c(1000, 1100, 1200, 1150, 1250),
  Investimento = c(300, 320, 330, 310, 340),
  Regiao = c("Sudeste", "Sul", "Sudeste",
"Norte", "Nordeste")
)
```

7.2 MEDIDAS DE TENDÊNCIA CENTRAL

- **Média (mean)**

```
mean(PIBbase3$PIB)
```

- **Mediana (median)**

```
median(PIBbase3$PIB)
```

- **Moda (mais frequente)**

O R não tem uma função nativa para moda, mas podemos usar:

```
# Criando um vetor simples com repetições
valores <- c(10, 20, 10, 30, 20, 10)

# Função para calcular a moda
moda <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

# Aplicando a função
moda(valores)
```

OBS: Criou-se um vetor para apresentar a aplicação da

função moda.

Explicação rápida da função:

- unique(x): pega os valores únicos do vetor
- match(x, ux): mapeia a posição de cada elemento original no vetor de únicos
- tabulate(...): conta quantas vezes cada valor aparece
- which.max(...): retorna a posição do valor mais frequente

7.3 MEDIDAS DE DISPERSÃO

- **Desvio padrão (sd)**

```
sd(PIBbase3$PIB)
```

- **Variância (var)**

```
var(PIBbase3$PIB)
```

- **Amplitude (máximo - mínimo)**

```
range(PIBbase3$PIB)        # mostra min e max
diff(range(PIBbase3$PIB))  # calcula a amplitude
```

7.4 RESUMO ESTATÍSTICO AUTOMÁTICO

```
summary(PIBbase3)
```

Esse comando exibe: i) Mínimo; ii) 1º quartil (Q1); iii) Mediana; iv) Média; v) 3º quartil (Q3); e vi) Máximo.

7.5 MEDIDAS DE POSIÇÃO (QUANTIS)

- **Quartis:**

```
quantile(PIBbase3$PIB)
```

- **Percentis (exemplo: 90º percentil):**

```
quantile(PIBbase3$PIB, probs = 0.9)
```

7.6 TABELAS DE FREQUÊNCIA (QUALITATIVAS)

Para variáveis categóricas como Regiao, use table():

```
table(PIBbase3$Regiao)
```

Com proporções:

```
prop.table(table(PIBbase3$Regiao))
```

7.7 ESTATÍSTICA DESCRITIVA COM O PACOTE PSYCH

O pacote psych fornece uma visão mais detalhada com várias medidas numéricas de uma vez:

```
install.packages("psych")
library(psych)

describe(PIBbase3[, c("PIB", "Investimento")])
```

TERMO[4]	SIGNIFICADO
vars	Número identificador da coluna (ordem das variáveis analisadas)
n	Número de observações válidas (sem NA)
mean	Média aritmética (soma dos valores ÷ número de observações)
sd	Desvio padrão (medida de dispersão)
median	Mediana (valor central dos dados ordenados)
trimmed	Média aparada (exclui 10% dos valores extremos antes de calcular a média)
mad	Desvio absoluto mediano (medida robusta de dispersão)
min	Valor mínimo da variável
max	Valor máximo da variável
range	Amplitude (máximo - mínimo)
skew	Assimetria (simetria da distribuição; 0 = simétrica)
kurtosis	Curtose (achatamento da distribuição; 3 = normal)
se	Erro padrão da média (sd / √n)

4. Interpretações rápidas:
- **skew:** positivo = cauda para a direita; negativo = cauda para a esquerda
- **kurtosis:** > 3 indica distribuição com cauda pesada (leptocúrtica), < 3 = cauda leve (platicúrtica)
- **trimmed:** útil quando há muitos outliers — ela suaviza o impacto desses valores extremos
- **mad:** mais resistente a outliers que o desvio padrão

7.8 ESTATÍSTICAS POR GRUPO

```
PIBbase4 <- data.frame(
  Regiao = c(“Sudeste”, “Sudeste”, “Sul”,
“Sul”, “Nordeste”, “Nordeste”),
  PIB = c(1200, 1300, 900, 950, 700, 750)
)

library(dplyr)

PIBbase4 %>%
  group_by(Regiao) %>%
  summarise(
    media_PIB = mean(PIB),
    mediana_PIB = median(PIB),
    sd_PIB = sd(PIB)
  )
```

8. VISUALIZAÇÃO DE DADOS COM R

A **visualização de dados** é essencial para explorar padrões, comunicar resultados e gerar insights em análises econômicas. O R oferece desde gráficos simples com funções base até visualizações sofisticadas com o pacote ggplot2, um dos mais poderosos e flexíveis da linguagem.

Nesta seção, trabalharemos com o pacote **ggplot2**, parte do **tidyverse**, por sua clareza, estética e padronização.

8.1 INSTALANDO E CARREGANDO O GGPLOT2

```
install.packages("ggplot2")
library(ggplot2)
```

Base de dados usada

Vamos criar a base PIBbase5, contendo regiões, valores de PIB e de investimento:

```
PIBbase5 <- data.frame(
  Regiao = c("Sudeste", "Sudeste", "Sul", "Sul",
"Nordeste", "Nordeste"),
  PIB = c(1200, 1300, 900, 950, 700, 750),
  Investimento = c(400, 420, 280, 300, 200, 210)
)
```

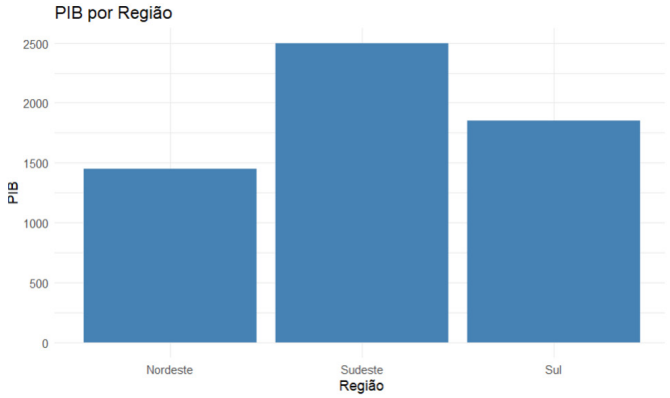
8.2 GRÁFICOS COM GGPLOT2

8.2.1 GRÁFICO DE BARRAS

Ideal para comparar valores entre grupos:

```
ggplot(PIBbase5, aes(x = Regiao, y = PIB)) +
  geom_col(fill = "steelblue") +
  labs(title = "PIB por Região", x = "Região", y =
"PIB") +
  theme_minimal()
```

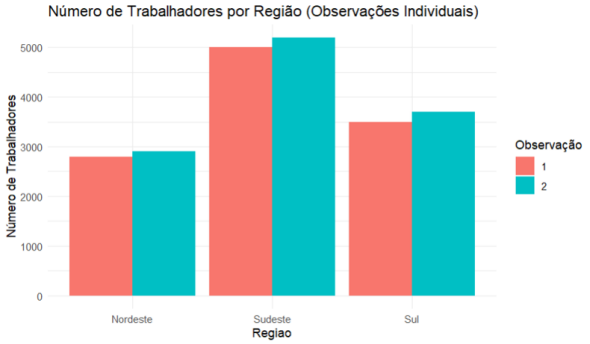
Gráfico produzido:



8.2.2 GRÁFICO DE COLUNAS EMPILHADAS

```
trab <- data.frame(
  Regiao = c("Sudeste", "Sudeste", "Sul", "Sul",
"Nordeste", "Nordeste"),
  Trabalhadores = c(5000, 5200, 3500, 3700, 2800,
2900),
  Investimento = c(400, 420, 280, 300, 200, 210)
)

# Criando a coluna de observação
trab$Observacao <- rep(1:2, 3)
library(ggplot2)

ggplot(trab, aes(x = Regiao, y = Trabalhadores, fill =
factor(Observacao))) +
  geom_col(position = "dodge") +
  labs(title = "Número de Trabalhadores por Região
(Observações Individuais)",
       y = "Número de Trabalhadores", fill =
"Observação") +
  theme_minimal()
```

Gráfico criado:



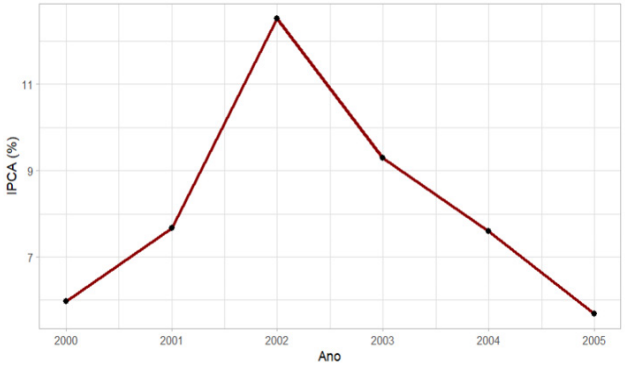
Dica: No lugar de observações pode ser utilizado o período.

8.2.3 GRÁFICO DE LINHAS

Usando uma base simulada com anos:

```
# Criando a base de dados
IPCA_tempo <- data.frame(
  Ano = 2000:2005,
  IPCA = c(5.97, 7.67, 12.53, 9.30, 7.60, 5.69)
)

# Carregando o pacote
library(ggplot2)

# Gráfico com legenda automática
ggplot(IPCA_tempo, aes(x = Ano, y = IPCA, color = “IPCA”))
+
  geom_line(size = 1.2) +
  geom_point(color = “black”, size = 2) +
  scale_color_manual(values = c(“IPCA” = “darkred”)) +
  labs(title = “Evolução do IPCA no Brasil (2000-2005)”,
       x = “Ano”, y = “IPCA (%)”,
       color = “Indicador”) +   # Título da legenda
  theme_light()
```

Este foi o resultado, após execução do código acima:



Exemplo com 2 linhas séries de tempo

```
# Criando a base de dados
dados_economia <- data.frame(
  Ano = 2000:2005,
  Juros = c(10.0, 12.0, 14.0, 10.5, 9.0, 8.5),
  Investimento = c(16.8, 17.0, 16.4, 15.3, 16.1, 15.9)
)

# Carregando o pacote
library(ggplot2)

# Transformando os dados para formato longo
library(tidyr)
dados_longos <- pivot_longer(dados_economia, cols = c("Juros",
"Investimento"),
                             names_to = "Indicador", values_to =
"Valor")

# Gerando o gráfico
ggplot(dados_longos, aes(x = Ano, y = Valor, color = Indicador)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  labs(title = "Taxa de Juros Real e Taxa de Investimento no Brasil
(2000-2005)",
       x = "Ano", y = "Percentual (%)", color = "Indicador") +
  theme_minimal()
```

Explicação do Pivot_longer[5] :

Por que isso? O ggplot2 precisa de uma coluna com o **tipo de variável** (nesse caso: “Juros” ou “Investimento”) e outra com os **valores numéricos**.

Essa função converte duas colunas (Juros e Investimento) em:

- Indicador: o nome da variável (ex: “Juros”)
- Valor: o número correspondente a cada ano

8.2.4 GRÁFICO DE DISPERSÃO (SCATTERPLOT)

Para verificar relações entre duas variáveis numéricas:

```
ggplot(PIBbase5, aes(x = Investimento, y = PIB)) +
  geom_point(size = 3, color = “tomato”) +
  geom_smooth(method = “lm”, se = FALSE, color =
“gray40”) +
  labs(title = “PIB vs Investimento”, x =
“Investimento”, y = “PIB”) +
  theme_classic()
```

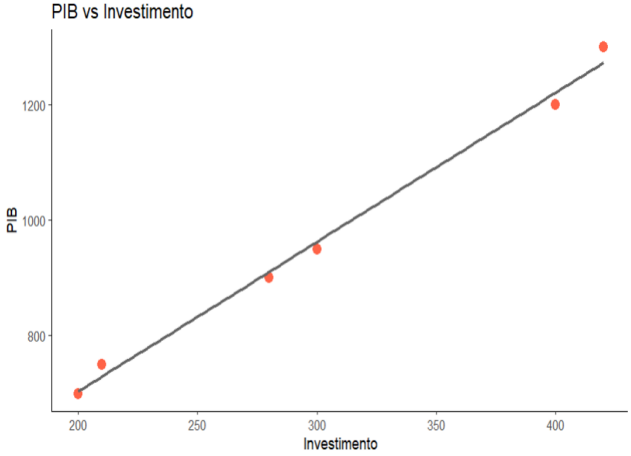
---

5. **cols = c(“Juros”, “Investimento”)**
- Aqui você está **especificando quais colunas** serão “esticadas” na vertical.
- Ou seja, queremos transformar as colunas Juros e Investimento em **linhas**, com uma nova coluna dizendo o nome do indicador.

names_to = “Indicador”
- Essa será a **nova coluna** que vai guardar os **nomes das variáveis originais**: “Juros” e “Investimento”.
- Ela indica **“de onde vêm os nomes das colunas que estavam no formato largo”.**

values_to = “Valor”
- Essa será a **nova coluna** que vai guardar os valores numéricos correspondentes (como 10.0, 16.8 etc).
- Ou seja, os dados que estavam distribuídos em várias colunas serão reunidos nessa **única coluna de valores.**

O resultado foi o seguinte:



OBS: Plotando também a relação linear (lm – linear model) entre as variáveis. geom_smooth() responsável por adicionar **uma linha de tendência** ao gráfico de dispersão, ou seja, um **modelo de regressão linear simples** traçado sobre os pontos.

? Este gráfico representa algo na Econometria?

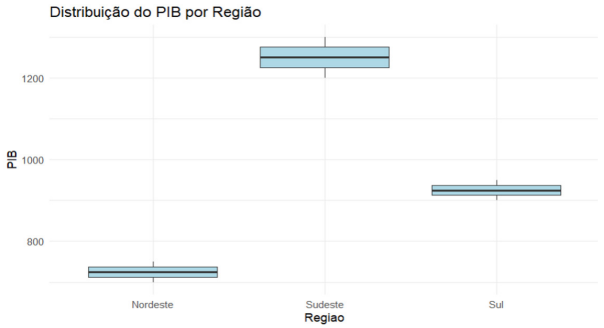
Outros modelos que poderiam ser utilizados são:

MÉTODO	FINALIDADE	QUANDO USAR	EXEMPLO DE FÓRMULA	OBSERVAÇÕES
lm	Regressão linear simples/ múltipla	Relações lineares e fácil interpretação	y ~ x	Mais comum e interpretável
glm	Regressão generalizada	Quando a variável dependente não é contínua (ex: binária)	y ~ x, family = gaussian()	Permite distribuições como binomial, poisson etc.
loess	Suavização local não paramétrica	Bases pequenas com relações não lineares suaves	—	Padrão do geom_smooth() quando method não é definido
gam	Regressão aditiva generalizada	Grandes amostras com relações complexas	y ~ s(x)	Requer o pacote mgcv
rlm	Regressão robusta (menos sensível a outliers)	Presença de outliers que distorcem a regressão	y ~ x	Requer o pacote MASS
nls	Regressão não linear com fórmula explícita	Quando a relação segue uma função conhecida (exponencial, log)	y ~ a * exp(b * x)	Deve ser ajustado fora do ggplot

8.2.5 BOXPLOT

```
ggplot(PIBbase4, aes(x = Regiao, y = PIB)) +
  geom_boxplot(fill = “lightblue”) +
  labs(title = “Distribuição do PIB por Região”, y = “PIB”) +
  theme_minimal()
```

Gráfico correspondente:



8.2.6 HISTOGRAMA (DISTRIBUIÇÃO DE FREQUÊNCIA)

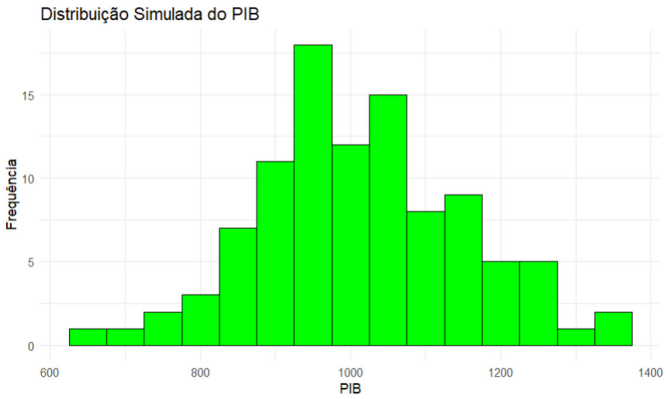
```
set.seed(123)  # Para reprodutibilidade

PIBbase_hist <- data.frame(
  PIB = round(rnorm(100, mean = 1000, sd = 150), 1)
)
```

- rnorm(100, mean = 1000, sd = 150): gera 100 valores com média 1000 e desvio padrão 150
- round(..., 1): arredonda para 1 casa decimal

Gerando o histograma com essa base:

```
ggplot(PIBbase_hist, aes(x = PIB)) +
  geom_histogram(binwidth = 50, fill =
“green”, color = “black”) +
  labs(title = “Distribuição Simulada do
PIB”, x = “PIB”, y = “Frequência”) +
  theme_minimal()
```

Como ficou no gráfico:



8.3 PAINEL COMPARATIVO

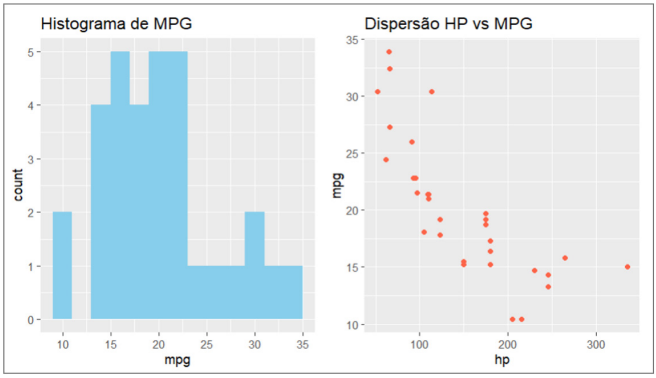
```
install.packages("patchwork")
library(patchwork)
```

Exemplo:

```
# Dois gráficos separados
g1 <- ggplot(mtcars, aes(x = mpg)) +
  geom_histogram(binwidth = 2, fill =
"skyblue") +
  labs(title = "Histograma de MPG")

g2 <- ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(color = "tomato") +
  labs(title = "Dispersão HP vs MPG")

# Juntar os dois
g1 + g2
```

OBS: matcars é uma base que já vem internamente no R. Para saber mas sobre a base: head(mtcars); summary(mtcars); plot(mtcars$hp, mtcars$mpg).

Os gráficos gerados pelo comando na cor laranja acima



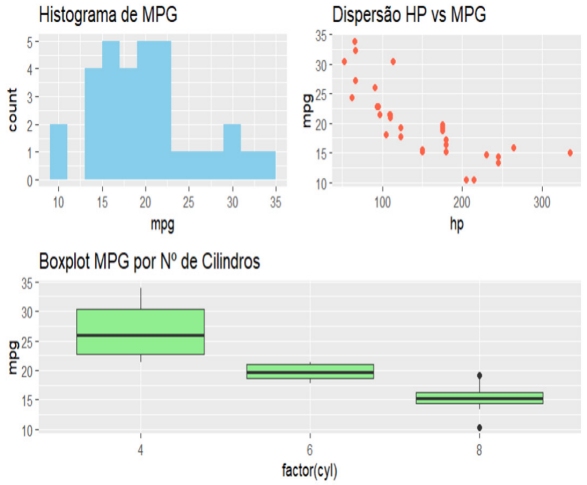
Dica: Colocando em colunas ou linhas, substituir por:

- Um ao lado do outro: g1 + g2
- Um abaixo do outro: g1 / g2
- Grid: (g1 | g2) / g3

Exemplo com grid:

```
g1 <- ggplot(mtcars, aes(x = mpg)) +
  geom_histogram(binwidth = 2, fill = “skyblue”) +
  labs(title = “Histograma de MPG”)

g2 <- ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(color = “tomato”) +
  labs(title = “Dispersão HP vs MPG”)

g3 <- ggplot(mtcars, aes(x = factor(cyl), y = mpg))
+
  geom_boxplot(fill = “lightgreen”) +
  labs(title = “Boxplot MPG por N° de Cilindros”)

(g1 | g2) / g3
```

Como ficou plotado:



O que significa essa sintaxe?

- | → coloca **lado a lado** (colunas)
- / → coloca **um embaixo do outro** (linhas)
- No exemplo: g1 e g2 ficam na **linha de cima**, e g3 fica **abaixo**, ocupando toda a linha de baixo.

facet_wrap() ou facet_grid() (quando os dados estão na mesma base)

Ideal quando você quer criar **múltiplos gráficos do mesmo tipo**, divididos por uma variável (ex: região, ano, setor):

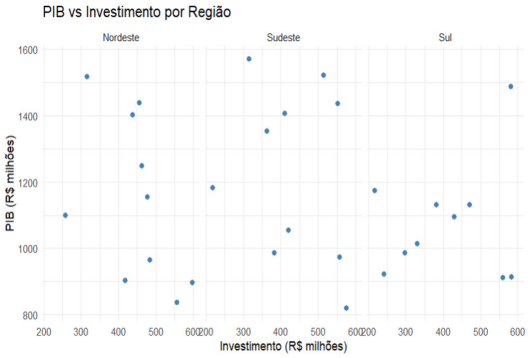
Criando a base dados_facetados

```
set.seed(123)  # para reprodutibilidade

dados_facetados <- data.frame(
  Regiao = rep(c("Sudeste", "Sul", "Nordeste"),
each = 10),
  Investimento = round(runif(30, 200, 600), 1),
# valores entre 200 e 600
  PIB = round(runif(30, 800, 1600), 1)
# valores entre 800 e 1600
)
```

8.4 VISUALIZANDO COM FACET_WRAP()

```
library(ggplot2)

ggplot(dados_grid, aes(x = Investimento, y = PIB))
+
  geom_point(color = "tomato", size = 2) +
  facet_grid(Ano ~ Regiao) +
  labs(title = "PIB vs Investimento por Região e
Ano",
       x = "Investimento (R$ milhões)", y = "PIB
(R$ milhões)") +
  theme_minimal()
```

Gráfico criado:



Criando a base para facet_grid()

Vamos estender a base anterior, adicionando uma variável de Ano:

```
set.seed(123)

dados_grid <- data.frame(
  Regiao = rep(c("Sudeste", "Sul", "Nordeste"),
each = 6),
  Ano = rep(c(2020, 2021), times = 9),  # al-
terna entre os anos
  Investimento = round(runif(18, 200, 600), 1),
  PIB = round(runif(18, 800, 1600), 1)
)
```

8.5 GRÁFICO COM FACET_GRID()

```
library(ggplot2)

ggplot(dados_facetados, aes(x = Investimento, y =
PIB)) +
  geom_point(color = "steelblue", size = 2) +
  facet_wrap(~Regiao) +
  labs(title = "PIB vs Investimento por Região",
       x = "Investimento (R$ milhões)",
       y = "PIB (R$ milhões)") +
  theme_minimal()
```

Dica: Se quiser deixar as escalas independentes por painel (quando os valores variam muito), adicione:

```
facet_grid(Ano ~ Regiao, scales = "free")
```

Quadro com as principais diferenças entre os facet_

grid() e facet_wrap()

CARACTERÍSTICA	FACET_WRAP()	FACET_GRID()
Organização	Disposição automática em linha única ou várias linhas	Grade com linhas e colunas fixas
Usa 1 ou 2 variáveis?	Normalmente 1 variável	Exige 2 variáveis (ou repetição de uma delas)
Controle do layout	Menos controle — automático	Mais controle: estrutura explícita
Sintaxe	facet_wrap(~ variável)	facet_grid(linhas ~ colunas)
Uso típico	Separar gráficos por uma dimensão (ex: região)	Comparar duas dimensões (ex: ano x região)

Quadro com as principais corres utilizadas no pacote (ggplot2)

NOME DA COR	VISUAL	CÓDIGO NO GGPLOT2
Azul		fill = 'blue'
Vermelho		fill = 'red'
Verde		fill = 'green'
Cinza		fill = 'gray'
Amarelo		fill = 'yellow'
Roxo		fill = 'purple'
Laranja		fill = 'orange'
Preto		fill = 'black'
Branco		fill = 'white'
Marrom		fill = 'brown'
Coral		fill = 'coral'
Tomate		fill = 'tomato'
Verde-limão		fill = 'limegreen'
Dourado		fill = 'gold'
Rosa quente		fill = 'hotpink'
Ciano		fill = 'cyan'
Magenta		fill = 'magenta'
Turquesa		fill = 'turquoise'
Laranja escuro		fill = 'darkorange'
Vermelho indiano		fill = 'indianred'
Azul claro		fill = 'lightblue'
Verde claro		fill = 'lightgreen'
Amarelo claro		fill = 'lightyellow'
Cinza claro		fill = 'lightgray'
Cor de pele		fill = 'navajowhite'
Lavanda		fill = 'lavender'
Rosa claro		fill = 'lightpink'
Bege		fill = 'beige'
Turquesa claro		fill = 'paleturquoise'
Verde menta		fill = 'mintcream'

9. TRANSFORMAÇÕES APLICADAS EM SÉRIES ECONÔMICAS COM DADOS REAIS

Neste tópico iremos apresentar, passo a passo, como importar, transformar e visualizar séries temporais reais utilizando o R. A série escolhida será a Inflação (IPCA – IBGE/SIDRA)

9.1 IMPORTAÇÃO DE DADOS REAIS

IPCA Mensal (IBGE/SIDRA – Tabela 1419)

```
library(sidrar)
library(dplyr)

ipca <- get_sidra(
  api = “/t/1419/n1/all/v/63/p/all/c315/7169”,
  format = 2
) %>%
  select(Data = “Mês (Código)”, IPCA = “Valor”) %>%
  mutate(Data = as.Date(paste0(Data, “01”), format
= “%Y%m%d”))
```

9.2 TRANSFORMAÇÕES ECONÔMICAS ÚTEIS

9.2.1 LOGARITMO NATURAL

```
ipca$ln_IPCA <- log(ipca$IPCA)
```

O logaritmo (log()) não pode ser calculado para valores menores ou iguais a zero, então se houver:

- valores zerados (0)
- valores negativos
- ou até mesmo NA

... o R retorna NaN (Not a Number) como forma de aviso.

Filtrar os valores positivos antes de aplicar o log

```
ipca <- ipca %>%
  filter(IPCA > 0) %>%
  mutate(ln_IPCA = log(IPCA))
```

Aplicar o log só nos valores válidos, mantendo os outros como NA:

```
ipca$ln_IPCA <- ifelse(ipca$IPCA > 0, log(ipca$IPCA), NA)
```

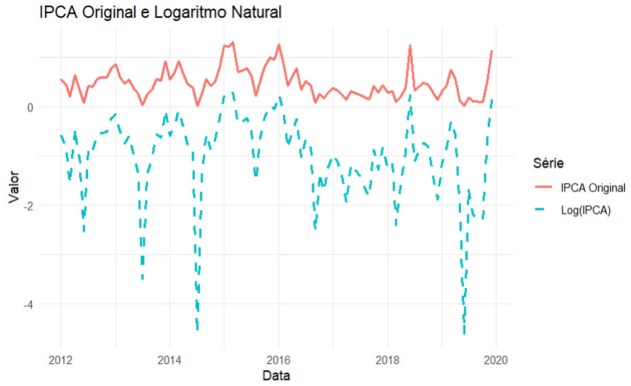
Plotar gráfico:

```
library(ggplot2)

ggplot(ipca, aes(x = Data)) +
  geom_line(aes(y = IPCA, color = “IPCA Original”), size
= 1) +
  geom_line(aes(y = ln_IPCA, color = “Log(IPCA)”),
linetype = “dashed”, size = 1) +
  labs(title = “IPCA Original e Logaritmo Natural”,
       y = “Valor”,
       color = “Série”) +
  theme_minimal()
```

Outros tipos de linha

LINETYPE	ESTILO VISUAL	TRADUZIDO
"solid"	——————— (padrão)	linha contínua
"dashed"	— — — —	linha tracejada
"dotted"	●●●●●●●	linha pontilhada
"dotdash"	• — • — •	linha ponto-traço
"longdash"	——— ——— ———	linha com traço longo
"twodash"	— —— — ——	linha com dois traços curtos

Gráfico plotado



9.2.2 VARIAÇÃO PERCENTUAL

```
ipca$Crescimento <- (ipca$IPCA / lag(ipca$IPCA) - 1) * 100
```

9.2.3 DIFERENÇA DE LOG (TAXA DE CRESCIMENTO RELATIVA)

```
ipca$log_diff <- c(NA, diff(log(ipca$IPCA)))
```

OBS: c(NA, ...) → adiciona manualmente um NA no início da série.

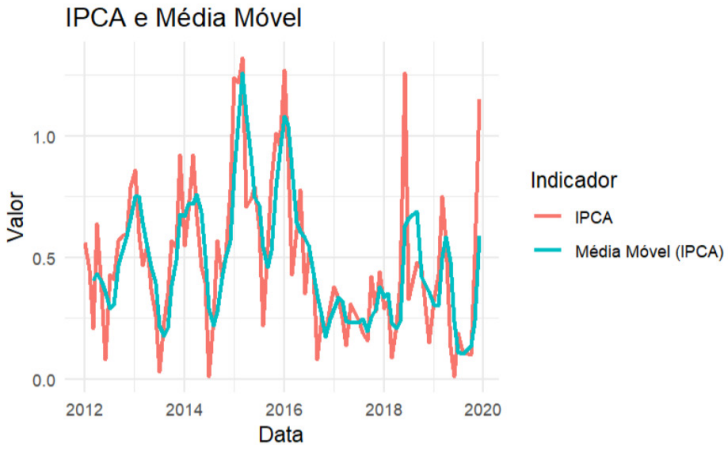
9.2.4 MÉDIA MÓVEL (3 MESES)

```
library(zoo)

ipca$MM3 <- rollmean(ipca$IPCA, 3, fill =
NA, align = “right”)
```

Criar um gráfico da média móvel e do IPCA

```
ggplot(ipca, aes(x = Data)) +
  geom_line(aes(y = IPCA, color = “IPCA”),
size = 1) +
  geom_line(aes(y = MM3, color = “Média
Móvel (IPCA)”), linetype = “solid”, size =
1) +
  labs(
    title = “IPCA e Média Móvel”,
    y = “Valor (%)”,
    color = “Indicador”
  ) +
  theme_minimal()
```

Como ficou:



9.2.5 ÍNDICE BASE 100

```
ipca$Base100 <- (ipca$IPCA / ipca$IPCA[1]) * 100
```

Como definir um ano ou mês específico como base 100

Vamos supor que você quer usar **janeiro de 2015** como referência:

```
base_ref <- ipca$IPCA[ipca$Data ==
as.Date("2015-01-01")]

ipca$Base100_2015 <- (ipca$IPCA / base_ref) * 100
```

9.2.6 NORMALIZAÇÃO (ESCALA 0–1)

```
ipca$Norm <- (ipca$IPCA - min(ipca$IPCA)) /
(max(ipca$IPCA) - min(ipca$IPCA))
```

9.2.7 ACUMULADO EM 12 MESES

```
ipca$Acum12m <- (ipca$IPCA / lag(ipca$IPCA, 12) - 1)
* 100
```

9.2.8 DEFASAGEM (LAG)

```
ipca$Lag1 <- lag(ipca$IPCA, 1)
```

9.2.9 CORRIGINDO VALORES NOMINAIS: CONSTRUINDO UM DEFLATOR COM O IPCA

Em análises econômicas, é fundamental converter variáveis nominais (afetadas pela inflação) em variáveis reais, que representam o poder de compra constante. Para isso, utiliza-se um deflator, que ajusta os valores nominais com base em um índice de preços — como o IPCA.

Passo 1 – Construir o índice base 100

```
ipca <- ipca %>%
  mutate(Deflator_IPCA = (IPCA / IPCA[Data ==
as.Date("2018-01-01")]) * 100)
```

Aqui, estamos definindo **janeiro de 2018 como base 100**. Você pode escolher outro mês alterando a data.

Passo 2 – Criar uma variável nominal fictícia (exemplo)

```
# Variável nominal fictícia para demonstrar a
correção
ipca$Receita_nominal <- seq(1000, by = 15, length.
out = nrow(ipca))
```

PARTE DO CÓDIGO	O QUE FAZ
seq(...)	Cria uma sequência de números
1000	Valor inicial da sequência: começa em 1000
by = 15	A cada linha, o valor aumenta 15 unidades
length.out = nrow(ipca)	Gera exatamente a mesma quantidade de valores da base IPCA
ipca$Receita_nominal <- ...	Atribui essa sequência como uma nova coluna no data frame

Passo 3 – Corrigir a série para valores reais

```
ipca <- ipca %>%
  mutate(Receita_real = Receita_nominal /
(Deflator_IPCA / 100))
```

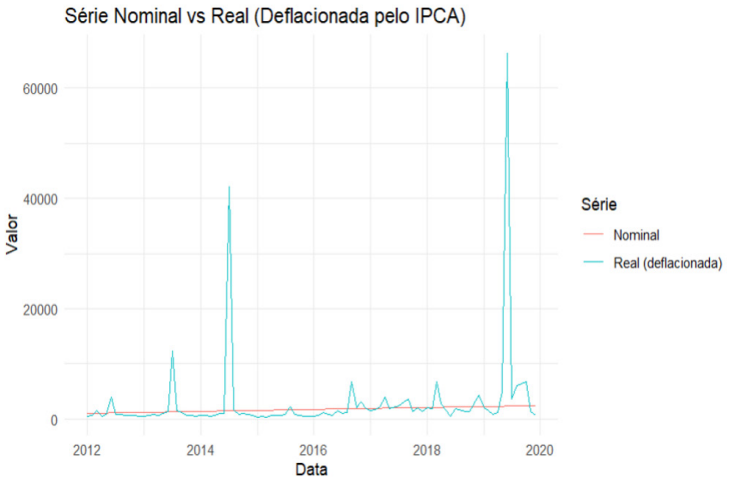
A fórmula aplica o deflator **dividido por 100**, já que ele foi criado com base 100.

O resultado é uma série **em preços constantes de jan/2018.**

Gráfico comparando a série nominal e a real

```
library(ggplot2)

ggplot(ipca, aes(x = Data)) +
  geom_line(aes(y = Receita_nominal, color
= “Nominal”)) +
  geom_line(aes(y = Receita_real, color =
“Real (deflacionada)”)) +
  labs(title = “Série Nominal vs Real
(Deflacionada pelo IPCA)”,
       y = “Valor”, color = “Série”) +
  theme_minimal()
```

Como ficou:



SOBRE OS AUTORES

STEFAN WILSON D'AMATO

Doutor em Economia Aplicada pelo Centro de Desenvolvimento e Planejamento Regional da Universidade Federal de Minas Gerais - CEDEPLAR / UFMG, com passagem pela Comissão Econômica para América Latina e Caribe - CEPAL/ONU. Tem experiência áreas de Teoria Macroeconômica Pós-Keynesiana e Macroeconomia do Desenvolvimento. Atualmente, exerço as funções de Consultor econômico, Conselheiro de política econômica e Pesquisador de Pós-doutorado pelo PPGE/UFPA. Graduado em Ciências Econômicas pela Universidade Federal de Ouro Preto - UFOP e Mestre em Economia pelo Departamento de Economia da Universidade Federal de Viçosa - UFV. Pesquisador no Grupo de Pesquisa Macroeconomia Estruturalista do Desenvolvimento (MED) e no Laboratório de Política Econômica e Desenvolvimento Produtivo - LAPED.

WALLACE MARCELINO PEREIRA

Doutor em Economia Aplicada pelo Centro de Desenvolvimento e Planejamento Regional da Universidade Federal de Minas Gerais - CEDEPLAR / UFMG, com passagem pela Comissão Econômica para América Latina e Caribe - CEPAL/ONU e pelo Centre of Latina American Studies - CLAS da University of Cambridge - UK. Graduado em Ciências Econômicas pela UFMG. Mestre em Economia da Indústria e Desenvolvimento Econômico pela Universidade Federal de Santa Catarina - UFSC. Tem experiência na área de Economia com ênfase em Macroeconomia, Desenvolvimento, Crescimento Econômico e Economia Regional. Atua principalmente nos seguintes temas: Mudança Estrutural (Industria e Serviços), Política Econômica, Planejamento Regional, Economia Brasileira e Latino Americana. Professor da UFPA e coordenador do Laboratório de Pesquisa em Política Econômica e Desenvolvimento Produtivo – LAPED.

LAPED

Laboratório de Pesquisa em
Política Econômica
e Desenvolvimento Produtivo

www.olaped.com.br

Com uma linguagem clara e abordagem prática, ***Introdução ao R e Analytics: Aplicações em Economia*** apresenta, de forma acessível, o uso da linguagem R como ferramenta essencial para a análise de dados e pesquisa empírica nas Ciências Econômicas. Voltado a estudantes e profissionais da área, o livro conduz o leitor passo a passo – da compreensão dos fundamentos da programação à aplicação de métodos analíticos em séries econômicas –, reforçando a importância do pensamento quantitativo e da autonomia na construção de evidências. Trata-se de uma obra indispensável para quem deseja dominar as competências analíticas que hoje definem a formação contemporânea em Economia.