Package: brfinance (via r-universe)

November 8, 2026

Title Access to Brazilian Macroeconomic and Financial Time Series

Version 0.9.0

Description Provides simplified access to selected Brazilian macroeconomic and financial time series from official sources, primarily the Central Bank of Brazil through the SGS (Sistema Gerenciador de Séries Temporais) API. The package enables users to quickly retrieve and visualize indicators such as the unemployment rate and the Selic interest rate using a standardized data structure. It is designed for data access and visualization purposes, without performing forecasts or statistical modeling. For more information, see the official API: <https://dadosabertos.bcb.gov.br/dataset/>.

License MIT + file LICENSE

URL https://github.com/efram2/brfinance

BugReports https://github.com/efram2/brfinance/issues

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports dplyr, ggplot2, scales, httr2, lubridate, labelled, yfR

Suggests testthat (>= 3.0.0), rmarkdown, knitr, tidyr

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author João Paulo dos Santos Pereira Barbosa [aut, cre]

Maintainer João Paulo dos Santos Pereira Barbosa <joao.31582129@gmail.com>

Config/pak/sysreqs make libicu-dev libxml2-dev libssl-dev libx11-dev zlib1g-dev

Repository https://cranhaven.r-universe.dev

Date/Publication 2026-11-08 03:02:00 UTC

RemoteUrl https://github.com/cranhaven/cranhaven.r-universe.dev

RemoteRef package/brfinance

RemoteSha 1b21b88db7e76c559aea37f33c917cfd99c36406

RemoteSubdir brfinance

Contents

br_available_series . . . 3
browse_series . . . 4
calc_amortization_schedule . . . 5
calc_compound_interest . . . 5
calc_continuous_compounding . . . 6
calc_effective_rate . . . 7
calc_future_value . . . 7
calc_future_value_ext . . . 8
calc_fv_annuity . . . 9
calc_irr . . . 9
calc_nominal_rate . . . 10
calc_nper . . . 10
calc_npv . . . 11
calc_pmt . . . 12
calc_present_value . . . 12
calc_pv_annuity . . . 13
calc_pv_continuous . . . 14
calc_rate . . . 14
calc_simple_interest . . . 15
get_cdi_rate . . . 16
get_exchange_rate . . . 17
get_gdp_growth . . . 19
get_ibovespa . . . 20
get_inflation_rate . . . 22
get_ipca_from_target . . . 24
get_selic_rate . . . 25
get_series_info . . . 27
get_unemployment . . . 27
plot_cdi_rate . . . 29
plot_exchange_rate . . . 29
plot_ibovespa . . . 30
plot_inflation_rate . . . 30
plot_selic_rate . . . 31
plot_series_comparison . . . 32
plot_unemployment . . . 33
rule_of_114 . . . 34
rule_of_72 . . . 35

Index **36**

`br_available_series` *Brazilian Economic Series Database*

Description

A comprehensive database of Brazilian economic and financial time series available through the Central Bank of Brazil's SGS system.

Usage

```
br_available_series
```

Format

A tibble with multiple rows and the following columns:

series_id SGS code from Central Bank of Brazil

source Data source

source_system Source system

short_name Short name of the series

long_name_pt Full name in Portuguese

long_name_en Full name in English

description_pt Description in Portuguese

description_en Description in English

category_pt Category in Portuguese

category_en Category in English

sub_category_pt Sub-category in Portuguese

sub_category_en Sub-category in English

frequency Data frequency

unit Measurement unit

seasonally_adjusted Logical

start_date Series start date

end_date Series end date

default_transform Suggested transformation

suggested_scale Suggested visualization scale

notes_pt Notes in Portuguese

notes_en Notes in English

Source

Central Bank of Brazil — SGS

Examples

```
br_available_series

br_available_series[br_available_series$series_id == "433", ]

br_available_series[br_available_series$series_id == "433", ]
```

`browse_series` *Browse Available Economic Series*

Description

Displays a searchable table of available Brazilian economic series.

Usage

```
browse_series(
  category = NULL,
  sub_category = NULL,
  source = NULL,
  frequency = NULL,
  search_text = NULL,
  language = c("pt", "en")
)
```

Arguments

`category`	Filter by category (optional)
`sub_category`	Filter by sub-category (optional)
`source`	Filter by source (optional)
`frequency`	Filter by frequency (optional)
`search_text`	Text search across all columns (optional)
`language`	Language for returned labels. Either `"pt"` or `"en"`.

Value

A tibble with available series matching the criteria

Examples

```
browse_series()
browse_series(category = "Macroeconomic")
browse_series(search_text = "inflação")
```

calc_amortization_schedule

Amortization Schedule

Description

Generates an amortization schedule for a loan.

Usage

```
calc_amortization_schedule(pv, rate, n, type = 0)
```

Arguments

pv	Loan amount.
rate	Interest rate per period (decimal).
n	Number of periods.
type	Payment timing: 0 for end of period, 1 for beginning of period.

Value

Data frame with amortization schedule.

Examples

```
calc_amortization_schedule(pv = 10000, rate = 0.01, n = 12)
```

calc_compound_interest

Compound Interest

Description

Calculates the final value under compound interest. This is an alias for `calc_future_value()` — both compute `pv * (1 + rate)^n` — kept as a separate exported name since "compound interest" and "future value" are the terms people search for depending on context (previously this was a fully duplicated implementation).

Usage

```
calc_compound_interest(pv, rate, n)
```

Arguments

pv	Present value.
rate	Interest rate per period (decimal).
n	Number of periods.

Value

Final value with compound interest.

Examples

```
calc_compound_interest(pv = 1000, rate = 0.10, n = 2)
```

---

calc_continuous_compounding

Continuous Compounding

Description

Calculates the future value with continuous compounding.

Usage

```
calc_continuous_compounding(pv, rate, t)
```

Arguments

pv	Present value.
rate	Nominal annual interest rate (decimal).
t	Time in years.

Value

Future value with continuous compounding.

Examples

```
calc_continuous_compounding(pv = 1000, rate = 0.05, t = 2)
```

calc_effective_rate *Effective Interest Rate*

Description

Converts a nominal interest rate into an effective rate.

Usage

```
calc_effective_rate(nominal_rate, m)
```

Arguments

nominal_rate	Nominal interest rate (decimal).
m	Number of compounding periods per year.

Value

Effective interest rate.

Examples

```
calc_effective_rate(nominal_rate = 0.12, m = 12) # 12% nominal monthly
```

calc_future_value *Future Value (FV) - Basic*

Description

Calculates the future value of a present amount given an interest rate and number of periods.

Usage

```
calc_future_value(pv, rate, n)
```

Arguments

pv	Present value.
rate	Interest rate per period (decimal).
n	Number of periods.

Value

Future value.

Examples

```
calc_future_value(pv = 1000, rate = 0.10, n = 1)
calc_future_value(pv = 500, rate = 0.05, n = 10)
```

`calc_future_value_ext` *Future Value Extended (with periodic payments)*

Description

Calculates the future value with optional periodic payments.

Usage

```
calc_future_value_ext(pv, rate, n, pmt = 0, type = 0)
```

Arguments

`pv`	Present value.
`rate`	Interest rate per period (decimal).
`n`	Number of periods.
`pmt`	Periodic payment (default = 0).
`type`	Payment timing: 0 for end of period (ordinary annuity), 1 for beginning of period (annuity due).

Value

Future value.

Examples

```
calc_future_value_ext(pv = 1000, rate = 0.05, n = 10, pmt = 100)
calc_future_value_ext(pv = 0, rate = 0.01, n = 12, pmt = 500, type = 1)
```

calc_fv_annuity *Future Value of an Annuity*

Description

Calculates the future value of a series of equal payments.

Usage

```
calc_fv_annuity(pmt, rate, n, type = 0)
```

Arguments

pmt	Payment amount per period.
rate	Interest rate per period (decimal).
n	Number of periods.
type	Payment timing: 0 for end of period (ordinary annuity), 1 for beginning of period (annuity due).

Value

Future value of the annuity.

Examples

```
calc_fv_annuity(pmt = 100, rate = 0.05, n = 10, type = 0)
calc_fv_annuity(pmt = 100, rate = 0.05, n = 10, type = 1)
```

calc_irr *Internal Rate of Return (IRR)*

Description

Calculates the internal rate of return of a series of cash flows.

Usage

```
calc_irr(cashflows, guess = 0.1)
```

Arguments

cashflows	Vector of cash flows (first is typically negative investment).
guess	Initial guess for IRR (default = 0.1).

Value

Internal rate of return.

Examples

```
calc_irr(cashflows = c(-1000, 300, 400, 500))
```

---

calc_nominal_rate *Nominal Interest Rate*

Description

Converts an effective interest rate into a nominal rate.

Usage

```
calc_nominal_rate(effective_rate, m)
```

Arguments

effective_rate	Effective interest rate (decimal).
m	Number of compounding periods per year.

Value

Nominal interest rate.

Examples

```
calc_nominal_rate(effective_rate = 0.1268, m = 12)
```

---

calc_nper *Number of Periods (NPER)*

Description

Calculates the number of periods for an investment.

Usage

```
calc_nper(pv, fv = 0, rate, pmt = 0, type = 0)
```

Arguments

pv	Present value.
fv	Future value.
rate	Interest rate per period (decimal).
pmt	Payment per period (optional, default = 0).
type	Payment timing: 0 for end of period, 1 for beginning of period.

Value

Number of periods.

Examples

```
calc_nper(pv = -1000, fv = 2000, rate = 0.05)
calc_nper(pv = 0, fv = 100000, rate = 0.01, pmt = -500)
```

---

calc_npv *Net Present Value (NPV)*

Description

Calculates the net present value of a series of cash flows.

Usage

```
calc_npv(rate, cashflows)
```

Arguments

rate	Discount rate per period (decimal).
cashflows	Vector of cash flows (first is typically negative investment).

Value

Net present value.

Examples

```
calc_npv(rate = 0.1, cashflows = c(-1000, 300, 400, 500))
```

calc_pmt *Loan Payment (PMT)*

Description

Calculates the periodic payment for a loan.

Usage

```
calc_pmt(pv, rate, n, type = 0)
```

Arguments

pv	Present value (loan amount).
rate	Interest rate per period (decimal).
n	Number of periods.
type	Payment timing: 0 for end of period, 1 for beginning of period.

Value

Periodic payment amount.

Examples

```
calc_pmt(pv = 10000, rate = 0.01, n = 12) # Loan de R$10k, 1% ao mês, 12 meses
```

calc_present_value *Present Value (PV)*

Description

Calculates the present value of a future amount given an interest rate and number of periods.

Usage

```
calc_present_value(fv, rate, n)
```

Arguments

fv	Future value.
rate	Interest rate per period (decimal, e.g. 0.05 for 5%).
n	Number of periods.

Value

Present value.

Examples

```
calc_present_value(fv = 1100, rate = 0.10, n = 1)
calc_present_value(fv = 1000, rate = 0.05, n = 5)
```

---

calc_pv_annuity *Present Value of an Annuity*

Description

Calculates the present value of a series of equal payments.

Usage

```
calc_pv_annuity(pmt, rate, n, type = 0)
```

Arguments

pmt	Payment amount per period.
rate	Interest rate per period (decimal).
n	Number of periods.
type	Payment timing: 0 for end of period, 1 for beginning of period.

Value

Present value of the annuity.

Examples

```
calc_pv_annuity(pmt = 100, rate = 0.05, n = 10)
calc_pv_annuity(pmt = 100, rate = 0.05, n = 10, type = 1)
```

calc_pv_continuous *Present Value with Continuous Compounding*

Description

Calculates the present value with continuous compounding.

Usage

```
calc_pv_continuous(fv, rate, t)
```

Arguments

fv	Future value.
rate	Nominal annual interest rate (decimal).
t	Time in years.

Value

Present value with continuous compounding.

Examples

```
calc_pv_continuous(fv = 1105.17, rate = 0.05, t = 2)
```

calc_rate *Interest Rate (RATE)*

Description

Calculates the interest rate per period.

Usage

```
calc_rate(
  n,
  pv,
  fv = 0,
  pmt = 0,
  type = 0,
  guess = 0.1,
  max_iter = 100,
  tol = 1e-08
)
```

Arguments

n	Number of periods.
pv	Present value.
fv	Future value.
pmt	Payment per period (optional, default = 0).
type	Payment timing: 0 for end of period, 1 for beginning of period.
guess	Initial guess for the rate (default = 0.1).
max_iter	Maximum number of iterations (default = 100).
tol	Tolerance for convergence (default = 1e-8).

Value

Interest rate per period.

Examples

```
calc_rate(n = 12, pv = -1000, fv = 2000)
calc_rate(n = 60, pv = -20000, fv = 0, pmt = 386.66)
```

---

calc_simple_interest *Simple Interest*

Description

Calculates the final value under simple interest.

Usage

```
calc_simple_interest(pv, rate, n)
```

Arguments

pv	Present value.
rate	Interest rate per period (decimal).
n	Number of periods.

Value

Final value with simple interest.

Examples

```
calc_simple_interest(pv = 1000, rate = 0.10, n = 2)
```

`get_cdi_rate` *Get CDI Rate (Interbank Deposit Certificate)*

Description

Downloads daily CDI (Certificado de Depósito Interbancário) rate from BCB/SGS. This function retrieves the daily CDI rate (SGS series 12), which is the benchmark interest rate for interbank transactions in Brazil.

Usage

```
get_cdi_rate(
  start_date = NULL,
  end_date = NULL,
  language = "eng",
  labels = TRUE
)
```

Arguments

`start_date`	Start date for the data period. Accepts multiple formats: • `"YYYY"` for year only (e.g., `"2020"` becomes `"2020-01-01"`) • `"YYYY-MM"` for year and month (e.g., `"2020-06"` becomes `"2020-06-01"`) • `"YYYY-MM-DD"` for a specific date (e.g., `"2020-06-15"`) • `NULL` defaults to the last 30 days, using today as the end date
`end_date`	End date for the data period. Accepts the same formats as `start_date`: • `"YYYY"` (e.g., `"2023"` becomes `"2023-12-31"`) • `"YYYY-MM"` (e.g., `"2023-12"` becomes the last day of December 2023) • `"YYYY-MM-DD"` for a specific date • `NULL` defaults to the current date (today)
`language`	Language for the `labelled` variable descriptions attached to the returned data.frame ("eng" or "pt"). The column names themselves are always `date`, `value` and `value_annualized`, so output plugs directly into `plot_cdi_rate()` and the rest of the package pipeline regardless of `language`.
`labels`	Logical indicating whether to add variable labels using the `labelled` package. Labels provide descriptive text for each column when available.

Value

A data.frame with columns:

date Reference date

value Daily CDI rate (% per day)

value_annualized Annualized CDI rate (% per year, 252 business days)

Note

Series information: This function uses SGS series 12, which represents the daily CDI rate (taxa de juros - CDI). The CDI is a key benchmark for fixed income investments and interbank lending in Brazil. Data is available from 1986 onward with daily frequency (business days only).

Examples

```
# Default: last 30 days of CDI rate
df <- get_cdi_rate()

# Specific period
df2 <- get_cdi_rate("2023-01-01", "2023-03-31")

# Using year-month format for a specific month
df3 <- get_cdi_rate("2023-06", "2023-06")

# Portuguese column names and labels
df4 <- get_cdi_rate(language = "pt")

# Complete example with all parameters
df5 <- get_cdi_rate("2023-01-01", "2023-12-31", language = "pt", labels = TRUE)

# Historical analysis
df6 <- get_cdi_rate("2020-03-01", "2020-04-30")  # COVID period
```

---

get_exchange_rate *Get US Dollar Exchange Rate (Commercial)*

Description

Downloads daily US dollar exchange rate (commercial, selling) from BCB/SGS. This function retrieves the daily exchange rate (SGS series 1) in Brazilian Real (R$).

Usage

```
get_exchange_rate(
  start_date = NULL,
  end_date = NULL,
  language = "eng",
  labels = TRUE
)
```

Arguments

`start_date` Start date for the data period. Accepts multiple formats:

- `"YYYY"` for year only (e.g., `"2020"` becomes `"2020-01-01"`)

- "YYYY-MM" for year and month (e.g., "2020-06" becomes "2020-06-01")
- "YYYY-MM-DD" for a specific date (e.g., "2020-06-15")
- NULL defaults to the last 30 days, using today as the end date

end_date — End date for the data period. Accepts the same formats as start_date:

- "YYYY" (e.g., "2023" becomes "2023-12-31")
- "YYYY-MM" (e.g., "2023-12" becomes the last day of December 2023)
- "YYYY-MM-DD" for a specific date
- NULL defaults to the current date (today)

language — Language for column names in the returned data.frame:

- "eng" (default): Returns columns date and exchange_rate
- "pt": Returns columns data_referencia and taxa_cambio

labels — Logical indicating whether to add variable labels using the labelled package. Labels provide descriptive text for each column when available.

Value

A data.frame with US dollar exchange rate. Columns depend on the language parameter:

- English (language = "eng"): date (Date), exchange_rate (numeric, R$/US$)
- Portuguese (language = "pt"): data_referencia (Date), taxa_cambio (numeric, R$/US$)

Note

Series information: This function uses SGS series 1, which represents the commercial US dollar selling rate (taxa de câmbio livre - dólar americano - venda). Data is available from 1984 onward with daily frequency.

Examples

```
# Default: last 30 days of exchange rate
df <- get_exchange_rate()

# Specific period
df2 <- get_exchange_rate("2023-01-01", "2023-03-31")

# Using year-month format for a specific month
df3 <- get_exchange_rate("2023-06", "2023-06")

# Portuguese column names and labels
df4 <- get_exchange_rate(language = "pt")

# Complete example with all parameters
df5 <- get_exchange_rate("2023-01-01", "2023-12-31", language = "pt", labels = TRUE)
```

`get_gdp_growth` *Get GDP Growth Rate*

Description

Downloads monthly nominal GDP growth data (% change) from BCB/SGS (Brazilian Central Bank). This function retrieves nominal GDP values (SGS series 2010) and calculates the month-over-month growth rate.

Usage

```
get_gdp_growth(
  start_date = NULL,
  end_date = NULL,
  language = "eng",
  labels = TRUE
)
```

Arguments

`start_date`	Start date for the data period. Accepts multiple formats: • `"YYYY"` for year only (e.g., `"2020"` becomes `"2020-01-01"`) • `"YYYY-MM"` for year and month (e.g., `"2020-06"` becomes `"2020-06-01"`) • `"YYYY-MM-DD"` for a specific date (e.g., `"2020-06-15"`) • `NULL` defaults to `"2000-01-01"`
`end_date`	End date for the data period. Accepts the same formats as `start_date`: • `"YYYY"` (e.g., `"2023"` becomes `"2023-12-31"`) • `"YYYY-MM"` (e.g., `"2023-12"` becomes the last day of December 2023) • `"YYYY-MM-DD"` for a specific date • `NULL` defaults to the current date (today)
`language`	Language for column names in the returned data.frame: • `"eng"` (default): Returns columns `date` and `gdp_growth` • `"pt"`: Returns columns `data_referencia` and `crescimento_pib`
`labels`	Logical indicating whether to add variable labels using the `labelled` package. Labels provide descriptive text for each column when available.

Value

A data.frame with GDP growth rate. Columns depend on the `language` parameter:

- English (`language = "eng"`): `date` (Date), `gdp_growth` (numeric, % month-over-month)
- Portuguese (`language = "pt"`): `data_referencia` (Date), `crescimento_pib` (numeric, % mes a mes)

Note

Frequency: SGS series 2010 (nominal GDP) has **monthly** frequency, so the growth rate this function computes is month-over-month, not quarter-over-quarter – this note previously (incorrectly) described it as quarterly.

Coverage limitation: The series is only available up to 2014. Requests for periods after 2014 will return empty results for that portion of the window (no error – just no rows past the series' last available date).

Examples

```
# Default: data from 2000 to current date (but limited to 2014)
df <- get_gdp_growth()

# Specific period (within available range)
df2 <- get_gdp_growth("2010", "2014")

# Using year-month format
df3 <- get_gdp_growth("2012-06", "2013-12")

# End date only (from earliest available to 2020-12-31)
df4 <- get_gdp_growth(end_date = "2020-12-01")

# Portuguese column names and labels
df5 <- get_gdp_growth(language = "pt")

# Complete example with all parameters
df6 <- get_gdp_growth("2011-01-01", "2014-12-31", language = "pt", labels = TRUE)
```

---

get_ibovespa *Get Ibovespa Index (B3)*

Description

Downloads the daily closing level of the Ibovespa (Índice Bovespa), Brazil's main stock market benchmark, via Yahoo Finance (ticker `^BVSP`). Uses the `yfR` package, the same data source already used in the `{brstocks}` package, so `{brfinance}` now covers the equity side alongside its macroeconomic series.

Usage

```
get_ibovespa(
  start_date = NULL,
  end_date = NULL,
  language = "eng",
  labels = TRUE
)
```

Arguments

start_date	Start date for the data period. Accepts multiple formats: • "YYYY" for year only (e.g., "2020" becomes "2020-01-01") • "YYYY-MM" for year and month (e.g., "2020-06" becomes "2020-06-01") • "YYYY-MM-DD" for a specific date (e.g., "2020-06-15") • NULL defaults to the last 30 days, using today as the end date
end_date	End date for the data period. Accepts the same formats as start_date: • "YYYY" (e.g., "2023" becomes "2023-12-31") • "YYYY-MM" (e.g., "2023-12" becomes the last day of December 2023) • "YYYY-MM-DD" for a specific date • NULL defaults to the current date (today)
language	Language for the labelled variable descriptions attached to the returned data.frame ("eng" or "pt"). Column names are always date and value.
labels	Logical indicating whether to add variable labels using the labelled package.

Value

A data.frame with columns:

date Trading date

value Ibovespa closing level (index points)

Note

Requires the yfR package (install.packages("yfR")) and an internet connection to Yahoo Finance. Unlike get_*() functions backed by BCB/SGS, this one is not subject to the 10-year window limit — yfR handles the full request in one call.

Examples

```
# Default: last 12 months
df <- get_ibovespa()

# Specific period
df2 <- get_ibovespa("2020-01-01", "2023-12-31")

# Portuguese labels
df3 <- get_ibovespa(language = "pt")
```

`get_inflation_rate` *Get IPCA Inflation Data*

Description

Downloads monthly IPCA (Broad National Consumer Price Index) inflation data from the Brazilian Central Bank's SGS API and calculates accumulated inflation rates. IPCA is Brazil's official inflation index, calculated monthly by IBGE.

Usage

```
get_inflation_rate(
  start_date = NULL,
  end_date = NULL,
  language = "eng",
  labels = TRUE
)
```

Arguments

`start_date`	Start date for the data period. Accepts multiple formats: • `"YYYY"` for year only (e.g., `"2020"` becomes `"2020-01-01"`) • `"YYYY-MM"` for year and month (e.g., `"2020-06"` becomes `"2020-06-01"`) • `"YYYY-MM-DD"` for a specific date (e.g., `"2020-06-15"`) • `NULL` defaults to `"2020-01-01"`
`end_date`	End date for the data period. Accepts the same formats as `start_date`: • `"YYYY"` (e.g., `"2023"` becomes `"2023-12-31"`) • `"YYYY-MM"` (e.g., `"2023-12"` becomes the last day of December 2023) • `"YYYY-MM-DD"` for a specific date • `NULL` defaults to the current date (today)
`language`	Language for column names in the returned data.frame: • `"eng"` (default): Returns columns `date`, `monthly_inflation`, `ytd_inflation`, `twelve_month_inflation` • `"pt"`: Returns columns `data_referencia`, `inflacao_mensal`, `inflacao_acumulada_ano`, `inflacao_12_meses`
`labels`	Logical indicating whether to add variable labels using the `labelled` package. Labels provide descriptive text for each column when available.

Value

A data.frame with inflation metrics. Columns depend on the `language` parameter:

- English (`language = "eng"`):
  - `date` (Date): Reference month

  - `monthly_inflation` (numeric): Monthly IPCA variation (%)
  - `ytd_inflation` (numeric): Year-to-date accumulated inflation (%)
  - `twelve_month_inflation` (numeric): 12-month accumulated inflation (%)
- Portuguese (`language = "pt"`):
  - `data_referencia` (Date): Mes de referencia
  - `inflacao_mensal` (numeric): Variacao mensal do IPCA (%)
  - `inflacao_acumulada_ano` (numeric): Inflacao acumulada no ano (%)
  - `inflacao_12_meses` (numeric): Inflacao acumulada nos ultimos 12 meses (%)

Note

Default Period: When `start_date = NULL`, defaults to `"2020-01-01"`, providing data from the start of 2020. This period covers significant economic events including the COVID-19 pandemic and recent inflationary pressures in Brazil.

Data Processing: This function automatically downloads an extra 12 months of historical data to calculate 12-month accumulated inflation correctly. The final output is filtered to show only the requested period.

Calculation Details:

- Year-to-date inflation: Cumulative product of monthly rates within each calendar year
- 12-month inflation: Rolling 12-month cumulative product of monthly rates

Examples

```
# Default: from 2020 to current date (aligned with SELIC function)
df <- get_inflation_rate()

# Specific period with year-only format
df2 <- get_inflation_rate("2021", "2023")

# Using year-month format for precise month selection
df3 <- get_inflation_rate("2022-03", "2023-06")

# Portuguese column names and labels
df4 <- get_inflation_rate(language = "pt")

# Without variable labels
df5 <- get_inflation_rate("2020-01-01", "2022-12-31", labels = FALSE)

# Current year analysis
current_year <- format(Sys.Date(), "%Y")
df6 <- get_inflation_rate(start_date = current_year)

# Compare with SELIC rate (same default period)
selic_data <- get_selic_rate()  # Also starts at 2020-01-01
inflation_data <- get_inflation_rate()  # Same start date
```

`get_ipca_from_target` *Get IPCA vs. Inflation Target Gap*

Description

Compares 12-month accumulated IPCA inflation against the official inflation target set annually by the CMN (Conselho Monetário Nacional), and returns both series plus the gap between them (positive means inflation is running above target).

Usage

```
get_ipca_from_target(
  start_date = NULL,
  end_date = NULL,
  language = "eng",
  labels = TRUE
)
```

Arguments

`start_date`	Start date for the data period. Accepts multiple formats: • `"YYYY"` for year only (e.g., `"2020"` becomes `"2020-01-01"`) • `"YYYY-MM"` for year and month (e.g., `"2020-06"` becomes `"2020-06-01"`) • `"YYYY-MM-DD"` for a specific date (e.g., `"2020-06-15"`) • `NULL` defaults to `"2020-01-01"`
`end_date`	End date for the data period. Accepts the same formats as `start_date`: • `"YYYY"` (e.g., `"2023"` becomes `"2023-12-31"`) • `"YYYY-MM"` (e.g., `"2023-12"` becomes the last day of December 2023) • `"YYYY-MM-DD"` for a specific date • `NULL` defaults to the current date (today)
`language`	Language for the `labelled` variable descriptions attached to the returned data.frame ("eng" or "pt").
`labels`	Logical indicating whether to add variable labels using the `labelled` package.

Value

A data.frame with columns:

date Reference date (monthly)

ipca_12m IPCA inflation accumulated over the trailing 12 months (%)

target Official CMN inflation target for that calendar year (%)

gap `ipca_12m - target`, in percentage points

Note

Series used: SGS 13522 ("IPCA - Variação acumulada em 12 meses"), monthly, and SGS 13521 ("Meta para a inflação", set by the CMN), which has annual frequency — one value per calendar year. The target is broadcast to every month of its year via a left join on the year, so it lines up with the monthly `ipca_12m` series. If the CMN has not yet set a target for the most recent year in the requested window, `target`/`gap` will be `NA` for those months rather than silently reusing an old value.

Examples

```
df <- get_ipca_from_target()
df2 <- get_ipca_from_target("2015", "2024", language = "pt")
```

`get_selic_rate` *Get Daily Brazilian SELIC Rate (Daily and Annualized)*

Description

Downloads the daily SELIC rate series from the Central Bank of Brazil's SGS API. The SELIC rate (Special System for Settlement and Custody) is Brazil's benchmark overnight interest rate, used as the primary monetary policy instrument. Returns both the raw daily rate and its annualized (base-252) equivalent, mirroring the pattern already used by `get_cdi_rate()`.

Usage

```
get_selic_rate(
  start_date = NULL,
  end_date = NULL,
  language = "eng",
  labels = TRUE
)
```

Arguments

`start_date`	Start date for the data period. Accepts multiple formats: • `"YYYY"` for year only (e.g., `"2020"` becomes `"2020-01-01"`) • `"YYYY-MM"` for year and month (e.g., `"2020-06"` becomes `"2020-06-01"`) • `"YYYY-MM-DD"` for a specific date (e.g., `"2020-06-15"`) • `NULL` defaults to the last 30 days, using today as the end date
`end_date`	End date for the data period. Accepts the same formats as `start_date`: • `"YYYY"` (e.g., `"2023"` becomes `"2023-12-31"`) • `"YYYY-MM"` (e.g., `"2023-12"` becomes the last day of December 2023) • `"YYYY-MM-DD"` for a specific date • `NULL` defaults to the current date (today)

language	Language for the labelled variable descriptions attached to the returned data.frame ("eng" or "pt"). The column names themselves are always date, value and value_annualized, so output plugs directly into plot_selic_rate() and the rest of the package pipeline regardless of language.
labels	Logical indicating whether to add variable labels using the labelled package. Labels provide descriptive text for each column when available.

Value

A data.frame with columns:

date Reference date

value Daily SELIC rate (% per day)

value_annualized Annualized SELIC rate (% per year, 252 business days)

Note

Series information: This function uses SGS series **11** ("Taxa de juros - Selic"), the daily-quoted average rate of overnight repo operations backed by federal government securities held in the Selic system, published in % per day (% a.d.). The annualized column (value_annualized) is computed locally from that daily rate using the standard 252-business-day compounding formula: ((1 + value/100)^252 - 1) * 100 — the same formula get_cdi_rate() already uses for its own value_annualized column. This is offered as an alternative to fetching SGS series 1178 ("Selic anualizada base 252") directly, which is BCB's own officially published annualized series and may differ from the locally-computed value by rounding. Data is available from June 4, 1986 onward. Requests spanning more than 10 years are split and stitched together automatically (see .get_sgs_series()), so the BCB's 10-year window limit no longer needs to be handled manually.

Examples

```
# Default: from 2020 to current date
df <- get_selic_rate()
head(df)  # date, value (daily), value_annualized

# Specific period
df2 <- get_selic_rate("2020-01-01", "2023-12-31")

# Long historical window (>10 years) — chunked automatically
df3 <- get_selic_rate(start_date = "2005")

# Portuguese labels
df4 <- get_selic_rate(language = "pt")

# Complete year analysis
df5 <- get_selic_rate("2018", "2023")
```

get_series_info *Get Series Information*

Description

Returns detailed metadata about a specific economic series.

Usage

```
get_series_info(series_id, field = NULL, language = c("pt", "en"))
```

Arguments

series_id	The series ID (SGS code)
field	Specific field to return (optional)
language	Language for returned labels. Either "pt" or "en".

Value

A list with series metadata or a single value

Examples

```
get_series_info("433")
get_series_info("433", field = "description", language = "en")
```

get_unemployment *Get Brazilian Unemployment Rate (PNAD Continua)*

Description

Downloads monthly unemployment rate data from the Brazilian Central Bank's SGS (Sistema Gerenciador de Series Temporais). The series corresponds to the unemployment rate from IBGE's Continuous PNAD survey (PNAD Continua), replicated and made available by the Central Bank.

Usage

```
get_unemployment(
  start_date = NULL,
  end_date = NULL,
  language = "eng",
  labels = TRUE
)
```

Arguments

start_date	Start date for the data period. Accepts multiple formats: • "YYYY" for year only (e.g., "2020" becomes "2020-01-01") • "YYYY-MM" for year and month (e.g., "2020-06" becomes "2020-06-01") • "YYYY-MM-DD" for a specific date • NULL defaults to "2020-01-01"
end_date	End date for the data period. Accepts the same formats as start_date: • "YYYY" (e.g., "2023" becomes "2023-12-31") • "YYYY-MM" (e.g., "2023-12" becomes the last day of December 2023) • "YYYY-MM-DD" for a specific date • NULL defaults to the current date (today)
language	Language for column names in the returned data.frame: • "eng" (default): Returns columns date, unemployment_rate • "pt": Returns columns data, taxa_desemprego
labels	Logical indicating whether to add variable labels using the labelled package.

Value

A data.frame with:

- date (Date): Reference month
- value (numeric): Unemployment rate (%)

Note

Data Source: Brazilian Central Bank (SGS), series 24369. The data originates from IBGE's Continuous National Household Sample Survey (PNAD Continua) and is published with monthly frequency.

Although published monthly, the unemployment rate follows IBGE's moving-quarter methodology.

Examples

```
# Default: from 2020 to current date (aligned with other functions)
df <- get_unemployment()

# Specific period with year-only format
df2 <- get_unemployment("2018", "2023")

# Portuguese column names and labels
df3 <- get_unemployment(language = "pt")

# Without variable labels
df4 <- get_unemployment("2020-01-01", "2022-12-31", labels = FALSE)
```

plot_cdi_rate *Plot Brazilian CDI rate*

Description

Generates a time series plot of the CDI (Certificado de Depósito Interbancário) rate.

Usage

```
plot_cdi_rate(data, language = "eng")
```

Arguments

data	Tibble returned by `get_cdi_rate()`, with columns `date` and `value`.
language	Language for titles and labels: "pt" (Portuguese) or "eng" (English).

Value

A `ggplot2` object showing the CDI rate over time.

Examples

```
cdi_data <- get_cdi_rate("2020", "2024")
print(plot_cdi_rate(cdi_data))
print(plot_cdi_rate(get_cdi_rate(language = "pt"), language = "pt"))
```

plot_exchange_rate *Plot Brazilian exchange rate (USD/BRL)*

Description

Generates a time series plot of the USD/BRL exchange rate using data from `get_exchange_rate()`.

Usage

```
plot_exchange_rate(data, language = "eng")
```

Arguments

data	Tibble returned by `get_exchange_rate()`.
language	Language for titles and labels: "pt" (Portuguese) or "eng" (English).

Value

A `ggplot2` object showing the exchange rate over time.

Examples

```
exchange_data <- get_exchange_rate("2023-01-01", "2023-12-31")
print(plot_exchange_rate(exchange_data))
```

`plot_ibovespa` *Plot Ibovespa Index*

Description

Generates a time series plot of the Ibovespa index using data from `get_ibovespa()`.

Usage

```
plot_ibovespa(data, language = "eng")
```

Arguments

`data`	Tibble returned by `get_ibovespa()`, with columns `date` and `value`.
`language`	Language for titles and labels: "pt" (Portuguese) or "eng" (English).

Value

A `ggplot2` object showing the Ibovespa level over time.

Examples

```
ibov_data <- get_ibovespa("2023-01-01", "2023-12-31")
print(plot_ibovespa(ibov_data))
```

`plot_inflation_rate` *Plot Brazilian Inflation Rate (IPCA)*

Description

Generates a time series plot of Brazil's monthly inflation rate measured by the IPCA (Índice Nacional de Preços ao Consumidor Amplo).

Usage

```
plot_inflation_rate(data, language = "eng")
```

Arguments

data	Tibble returned by get_inflation_rate(), with columns date and value.
language	Language for titles and labels: "pt" (Portuguese) or "eng" (English).

Value

A ggplot2 object showing the monthly inflation rate over time.

Examples

```
inflation_data <- get_inflation_rate("2020", "2024")
print(plot_inflation_rate(inflation_data))
```

---

plot_selic_rate *Plot Brazilian SELIC rate (annualized, base 252)*

Description

Generates a time series plot of the SELIC interest rate using data from get_selic_rate(). Plots the value_annualized column (% p.a.), not the raw daily value column.

Usage

```
plot_selic_rate(data, language = "eng")
```

Arguments

data	Tibble returned by get_selic_rate().
language	Language for titles and labels: "pt" (Portuguese) or "eng" (English).

Value

A ggplot2 object showing the SELIC rate over time.

Examples

```
selic_data <- get_selic_rate("2020", "2024")
print(plot_selic_rate(selic_data))
print(plot_selic_rate(get_selic_rate(language = "pt"), language = "pt"))
```

plot_series_comparison *Compare multiple financial/time series indices*

Description

Plots multiple time series on the same chart for comparison.

Usage

```
plot_series_comparison(
  data_list,
  y_vars,
  date_vars,
  language = "eng",
  scale_type = c("none", "index", "percent_change"),
  title = NULL,
  subtitle = NULL,
  y_label = NULL,
  caption = NULL,
  colors = NULL,
  line_types = NULL,
  show_legend = TRUE,
  legend_position = "bottom",
  dual_axis = FALSE
)
```

Arguments

data_list	Named list of data frames, each returned by a get_* function
y_vars	Vector of column names containing the values to plot from each data frame
date_vars	Vector of column names containing dates from each data frame
language	Language for labels: "pt" (Portuguese) or "eng" (English)
scale_type	Scaling applied to the series: **"none"** Plots raw values as provided **"index"** Indexes all series to 100 at the first observation **"percent_change"** Plots percentage change relative to the first observation
title	Plot title
subtitle	Plot subtitle
y_label	Y-axis label
caption	Plot caption
colors	Vector of colors for each series
line_types	Vector of line types for each series

show_legend Whether to show the legend (default: TRUE)

legend_position
Position of legend ("bottom", "top", "left", "right", or "none")

dual_axis Logical. If TRUE, plots exactly 2 series on separate y-axes (a primary axis on the left for the first series, a secondary axis on the right for the second) instead of overlaying them on one scale. Useful for comparing series with very different units, e.g. SELIC (%) vs. the exchange rate (R$/US$). Requires length(data_list) == 2 and is incompatible with scale_type != "none", since the whole point of a dual axis is to preserve each series' native scale.

Value

A ggplot2 object

Examples

```
# Example comparing multiple series
selic <- get_selic_rate(2020, 2024)
ipca <- get_ipca_from_target(2020, 2024)

comparison_plot <- plot_series_comparison(
  data_list = list(SELIC = selic, IPCA = ipca),
  y_vars = c("value", "ipca_12m"),
  date_vars = c("date", "date"),
  scale_type = "index",
  title = "Comparison of Brazilian Economic Indicators",
  y_label = "Index (2020-01 = 100)",
  language = "eng"
)
print(comparison_plot)

# Same two series, but on their native scales via a dual axis
dual_plot <- plot_series_comparison(
  data_list = list(SELIC = selic, `Exchange Rate` = get_exchange_rate(2020, 2024)),
  y_vars = c("value", "value"),
  date_vars = c("date", "date"),
  dual_axis = TRUE
)
print(dual_plot)
```

---

plot_unemployment *Plot Brazil's monthly unemployment rate*

Description

Generates a ggplot2 line chart of Brazil's unemployment rate (PNAD Contínua) using data returned by get_unemployment().

Usage

```
plot_unemployment(data, language = "eng")
```

Arguments

data	Tibble or data.frame returned by `get_unemployment()`. Must contain columns `date` (Date) and `value` (numeric).
language	Language of plot labels: "eng" (default) or "pt".

Value

A ggplot2 object.

Examples

```
unemployment_data <- get_unemployment("2020", "2024")
print(plot_unemployment(unemployment_data))
```

---

rule_of_114 *Rule of 114*

Description

Estimates the number of years required to triple an investment.

Usage

```
rule_of_114(rate)
```

Arguments

rate	Annual interest rate (decimal).

Value

Approximate years to triple.

Examples

```
rule_of_114(rate = 0.08)
```

`rule_of_72` *Rule of 72*

Description

Estimates the number of years required to double an investment.

Usage

```
rule_of_72(rate)
```

Arguments

`rate`	Annual interest rate (decimal).

Value

Approximate years to double.

Examples

```
rule_of_72(rate = 0.08)
```

Index

∗ **datasets**
br_available_series, 3

br_available_series, 3
browse_series, 4

calc_amortization_schedule, 5
calc_compound_interest, 5
calc_continuous_compounding, 6
calc_effective_rate, 7
calc_future_value, 7
calc_future_value_ext, 8
calc_fv_annuity, 9
calc_irr, 9
calc_nominal_rate, 10
calc_nper, 10
calc_npv, 11
calc_pmt, 12
calc_present_value, 12
calc_pv_annuity, 13
calc_pv_continuous, 14
calc_rate, 14
calc_simple_interest, 15

get_cdi_rate, 16
get_exchange_rate, 17
get_gdp_growth, 19
get_ibovespa, 20
get_inflation_rate, 22
get_ipca_from_target, 24
get_selic_rate, 25
get_series_info, 27
get_unemployment, 27

plot_cdi_rate, 29
plot_exchange_rate, 29
plot_ibovespa, 30
plot_inflation_rate, 30
plot_selic_rate, 31
plot_series_comparison, 32
plot_unemployment, 33

rule_of_114, 34
rule_of_72, 35