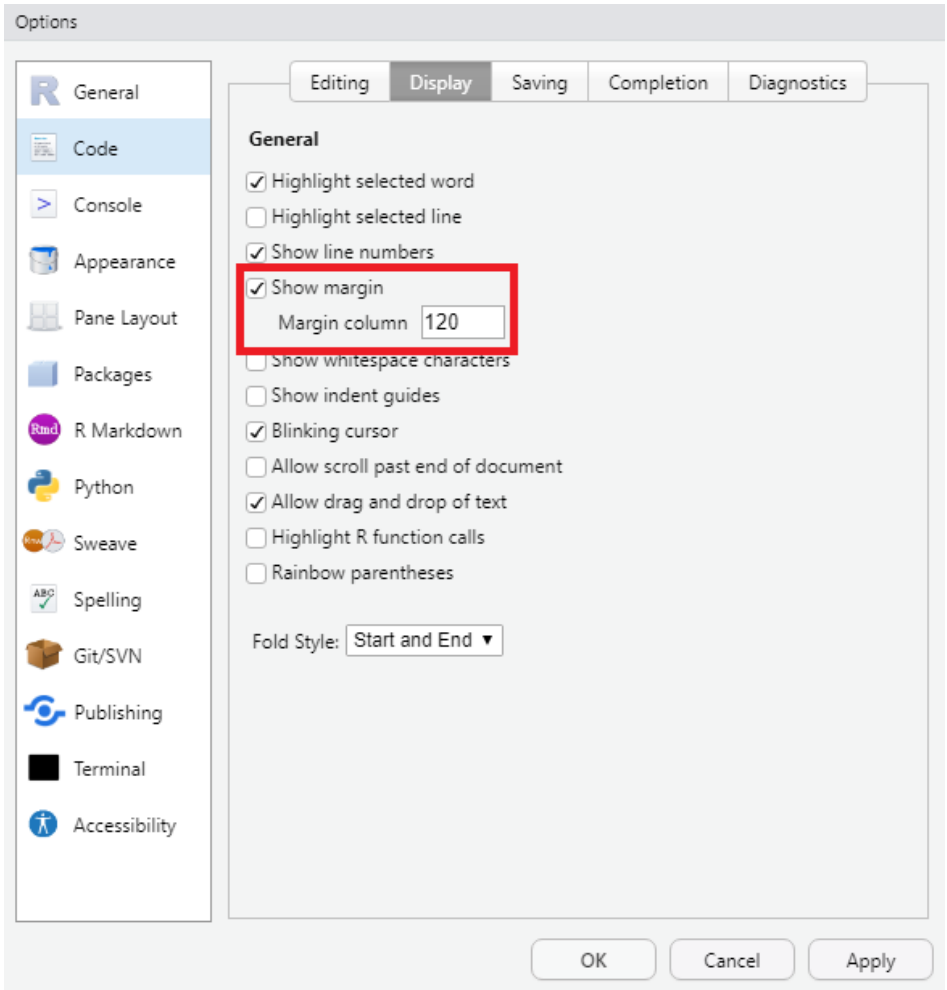
Carga de dados

Daniel C. Mota e Kleber G. Abitante

Organizar o seu código

- Como criar uma `margin column`: no RStudio, acesse Tools > Global Options > Code, aba Display.



Dividir o seu código em mais de um *script*

- Use a função `source()` para executar o código contido em outro *script*.

Obter e definir o diretório de trabalho

- Use a função `getwd()` para identificar a pasta de trabalho do R;
- Use a função `setwd()` para definir uma nova pasta de trabalho.
- **Atenção**: quando informar um diretório ou o caminho de um arquivo no R (em qualquer função), trocar as barras normais (\) por barras invertidas (/) ou por barras duplas (\\). Exemplo: `C:\Users` se torna `C:/Users` ou `C:\\Users`.

Carregar um arquivo CSV

- Pelo package `utils`:

```
pib_ipca <- read.csv("pib_ipca.csv", sep=";", dec=",")
```

- Pelo package `readr`:

```
library(readr)
pib_ipca <- read_csv2("pib_ipca.csv")
```

Carregar um arquivo CSV

- Quando o CSV possuir separador de milhar, melhor utilizar o pacote `readr` para carregá-lo, pois é possível informar o ponto como separador de milhar.

```
export <- read_delim("exportacoes.csv",
delim = ";", escape_double = FALSE,
locale = locale(decimal_mark = ",", grouping_mark = "."), trim_ws = TRUE,
show_col_types = FALSE)
```

Visualizar os dados

- Pelo package base:

```
plot(y=pib_ipca$pib_real, x=pib_ipca$ano, type="l", ylab="%", xlab=NA,
main="PIB real")
```

Visualizar os dados

- Pelo package `ggplot2`:
  a. gráfico de linha (uma série):

```
library(ggplot2)
ggplot(pib_ipca, aes(as.factor(ano), pib_real)) +
geom_line(color="red", group=1) +
labs(x = NULL, y = "%", title="PIB real (%)")
```

  - `ggplot`: função básica para gerar o gráfico;
  - `as.factor(ano)`: transformar o ano em `factor` para ele mostrar em um escala discreta no eixo "x";
  - `geom_line`: para gerar o gráfico de linha:
    * `color="red"`: cor da linha;
    * `group=1`: quantidade de grupos de dados para ele saber como conectar os pontos.
  - `labs`: para rótulos dos eixos e título do gráfico.

Visualizar os dados

b. gráfico de linha (duas séries):

```
ggplot(pib_ipca) +
geom_line(aes(as.factor(ano), pib_real, color="PIB real"), group=1) +
geom_line(aes(as.factor(ano), ipca, color="IPCA"), group=1) +
labs(x = NULL, y = "%", title="PIB real e IPCA") +
scale_color_manual(values=c("PIB real" = "blue","IPCA" = "red"))+
theme(
legend.title = element_blank(), legend.key=element_blank() )
```

- `color="PIB real"` e `color="IPCA"`: ambos foram colocados dentro da função aes e representam o nome das variáveis que será mostrado na legenda;
- `scale_color_manual(vvalues=c("PIB real" = "blue","IPCA" = "red"))`: cria a legenda, atribuindo a cor azul ao "PIB real" e a cor vermelha ao "IPCA";
- `legend.title = element_blank()`: remover o título da legenda;
- `legend.key=element_blank()`: remover o sombreado da legenda.

Exercícios 1 (4 min.)

a. Abra o arquivo "import_export.csv" no Excel e verifique se ele possui separador de milhar;

b. Carregue os dados do arquivo "import_export.csv" para dentro do R usando a função apropriada. Os dados se referem às importações (coluna `import`) e exportações (coluna `export`) de mercadorias em geral pelo Brasil, em US$ milhões;

c. Gere um gráfico de linha das duas séries, inserindo título no eixo "y" e um título para o gráfico. A linha das importações deve ser da cor preta (`black`) e a linha das exportações deve ser da cor verde (`green`).

Visualizar os dados

c. gráfico de barras:

```
ggplot(pib_ipca, aes(as.factor(ano), pib_real)) +
geom_bar(stat="identity", fill="lightblue") +
labs(x = NULL, y = "%", title="PIB real") +
scale_y_continuous(n.breaks=8, limits=c(-4,10)) + geom_text(aes(label =
format(round(pib_real,1), decimal.mark = ",", scientific = FALSE), vjust =
ifelse(pib_real>0, -1, 1.5))) +
theme( panel.background = element_rect(fill='transparent'),
panel.grid.major.y = element_line(color = "gray88", linewidth = 0.5,
linetype = 1), panel.grid.major.x = element_blank(),
plot.title = element_text(face = "bold"))
```

Visualizar os dados

- `geom_bar`: gráfico de barra;
- `stat="identity"`: para mostrar que a altura da coluna será o valor do `pib_real`;
- `fill="lightblue"`: preencher as barras de azul claro[1];
- `scale_y_continuous`: alterar a escala do eixo "y", que é contínuo;
- `n.breaks=8`: número de valores a ser mostrado no eixo "y";
- `limits=c(-4,10)`: valores mínimos e máximos, respectivamente, para o eixo "y";
- `geom_text(aes(label = format(round(pib_real,1), decimal.mark = ",", scientific = FALSE)), vjust = -1)`:
  * `geom_text(aes(label =`: informar que serão incluídos rótulos nas colunas;
  * `round(pib_real,1)`: arredondando os valores de `pib_real` para uma casa decimal;
  * `format(round(pib_real,1), decimal.mark = ",", scientific = FALSE)`: alterando o separador de decimal para vírgula e não permitir número científico;
  * `vjust = ifelse(pib_real>0, -1, 1.5)`: indica um ajuste vertical dos rótulos, para que fiquem acima ou abaixo das barras;

[1] SAPE. ggplot2 Quick Reference: colour (and fill).2023. Disponível em: http://sape.inf.usi.ch/quick-reference/ggplot2/colour. Acesso em: 01 mar. 2023

Visualizar os dados

- `theme`: customizar elementos do gráfico (excetos os dados):
  * `panel.background = element_rect(fill='transparent')`: eliminar a cor do fundo do gráfico;
  * `panel.grid.major.y = element_line(color = "gray88", linewidth = 0.5, linetype = 1)`: alterar as linhas principais que partem do eixo "y" para a cor "gray88", espessura = 0.5 e tipo de linha igual a 1;
  * `panel.grid.major.x = element_blank()`: eliminar as linhas principais que parte do eixo "x"; e
  * `plot.title = element_text(face = "bold")`: colocar o título do gráfico em negrito.

Carregar um arquivo .XLSX

- Usando o *package* `readxl`:

```
library(readxl)
res_prim <- read_excel("res_prim.xlsx")
```

Exercícios 2 (3 min.)

a. Carregue o arquivo "res_prim.xlsx" para a memória do R. O arquivo contém o resultado primário do Governo Central (coluna `res_prim_gov_central`) e resultado primário dos governos regionais (`res_prim_gov_reg`), ambos em % do PIB;

b. Gere um gráfico de barra apenas da coluna `res_prim_gov_central`. Coloque títulos nos eixos, título no gráfico (em negrito) e coloque as barras na cor bege claro (`wheat`). Por fim, coloque os rótulos nas barras, com arredondamento para 2 casas decimais.

Visualizar os dados

c. gráfico de barras e linhas com eixo secundário:

```
import_pib <- read_delim("importacoes_pib.csv", delim = ";",
escape_double = FALSE,locale = locale(decimal_mark = ",",
grouping_mark = "."), trim_ws = TRUE, show_col_types = FALSE)
import_pib$import <- import_pib$import/1000
ggplot(import_pib) +
geom_bar(aes(x=as.factor(ano), y=pib_real, fill="PIB real"),
stat="identity") +
geom_line(aes(x=as.factor(ano), y=import/50, group = 1, color =
"Importações")) +
labs(x = NULL, y = "PIB real (%)", title="PIB real e Importações") +
scale_y_continuous(n.breaks=7, limits=c(-5,10),
labels = function(x) format(x, decimal.mark = ",",
scientific = FALSE),
sec.axis = sec_axis(~.*50, name="Importações (US$ bilhões)")) +
geom_text(aes(x = as.factor(ano), y = pib_real,
label = format(round(pib_real,1), decimal.mark = ",",
scientific = FALSE)), vjust = -1, color = "lightblue4", size=3) +
```

Visualizar os dados

```
geom_text(aes(x = as.factor(ano), y = import/50,
label = format(round(import,1), decimal.mark = ",",
scientific = FALSE)), vjust = -1, color = "red3", size=3,
position = position_nudge(y = c(0,-1.5,0,0,0,0,0,0,0,0,0,-1.5,0))) +
scale_colour_manual(values=c("Importações" = "red3")) +
scale_fill_manual(values=c("PIB real" = "lightblue"))+
theme(panel.background = element_rect(fill='transparent'),
panel.grid.major.y = element_line(color = "gray88", linewidth = 0.5,
linetype = 1), panel.grid.major.x = element_blank(),
plot.title = element_text(face = "bold"),
legend.key=element_blank(),
axis.title.y = element_text(size = 8),
legend.title=element_blank(),
legend.position = "bottom")
```

Visualizar os dados

- `fill="PIB real"`: definir o nome para a legenda das barras;
- `color = "Importações"`: definir o nome para a legenda da linha;
- `scale_colour_manual(values=c("Importações" = "red3"))`: criar a legenda da linha, na cor "red3", com o nome "Importações";
- `scale_fill_manual(values="lightblue")`: criar a legenda das barras, na cor "lightblue";
- `axis.title.y = element_text(size = 8)`: tamanho da fonte do título do eixo "y";
- `legend.position = "bottom"`: posição da legenda.

O ponto principal para gerar um gráfico com eixo secundário é transformar os dados para a mesma escala (`import/50`, no exemplo) e depois criar o eixo secundário aplicando o mesmo valor usado para transformar a escala (`sec.axis = sec_axis(~.*50, name="Importações (US$ bilhões)")`, no exemplo, o termo `~.*50` está multiplicando o eixo y principal por 50).

Visualizar os dados

- Alguns pacotes possuem um arquivo PDF contendo um resumo de suas funcionalidades, chamado *cheat sheet* (folha de dicas);
- O *cheat sheet* do `ggplot2`, bem como de outros pacotes, pode ser encontrado no menu "Help" do RStudio > "Cheat sheets".
- No *cheat sheet* do `ggplot2`, os exemplos podem ser executados no R, pois as bases de dados que constam no *cheat sheet* são carregadas junto com o pacote.

Função download.file

- A função download.file() pode ser usada para baixar um arquivo da internet.

```
# url do arquivo do IBC-Br
url_ibcbr <- "https://www.bcb.gov.br/content/indeco/
indicadoresselecionados/ie-01.xlsx"
# arquivo de destino
pasta_dest <- "ibcbr.xlsx"
# baixar o arquivo
download.file(url_ibcbr, pasta_dest, mode="wb")
# carregar o arquivo para dentro do R
ibcbr <- read_excel(pasta_dest, range="A11:D27", col_names = F)
# alterar o nome das colunas
colnames(ibcbr) <- c("ano","mes","obs","dessaz")
```

Função download.file

```
# carregar o package zoo
library(zoo)
# preencher os NA's com o ultimo valor observado do ano
ibcbr$ano <- na.locf(ibcbr$ano)
# eliminar a linha que nao possui mes
ibcbr <- ibcbr[complete.cases(ibcbr),]
# criar coluna apenas com os dois ultimos numeros de ano
ibcbr$ano2 <- substr(ibcbr$ano, 3, 4)
# criar nova coluna de mes
ibcbr$mes <- paste(ibcbr$mes, ibcbr$ano2, sep="/")
# excluir a coluna ano2

ibcbr$ano2 <- NULL
```

Função download.file

- na.locf() do pacote zoo preenche os NAs com o último valor disponível.
- complete.cases() mantém apenas as linhas que não possuem nenhum NA.
- substr() extrair os caracteres em um intervalo informado pelo usuário.
- paste() serve para colar dois textos definindo um caracter de separação.
- <- NULL é usado para deletar linhas ou colunas de uma tabela.

Função GET

```
library(httr)
GET(url_ibcbr, write_disk(pastaDest, overwrite = T),
mode="wb")
```

- a função `write_disk()` gera um arquivo no caminho indicado.

Exercícios 3 (3 min.)

a. Baixar a planilha mais recente de "Estatísticas monetárias e de crédito" usando a função `read_excel` (acessar o site https://www.bcb.gov.br/estatisticas/estatisticasmonetariascredito, clicar com o botão direito em cima do botão "Tabelas (xlsx)" e clicar em "Copiar link". Esse é o link do arquivo a ser baixado);

b. Carregar o saldo das colunas "Saldos - PJ (R$ bilhões)" e "Saldos - PF (R$ bilhões)" da aba "Tab 3" do arquivo anterior, mas apenas o intervalo de janeiro de 2022 em diante (sem incluir as linhas de variação que constam no final). Use o argumento `sheet` para definir a aba, `range` para definir o intervalo desejado e `col_names` para definir se a primeira linha da `range` contém ou não os nomes das colunas).

Obter dados via API

- API é a sigla de Application Programming Interface (Interface de Programação de Aplicação). Estes são mecanismos que permitem que dois componentes de *software* se comuniquem usando um conjunto de definições e protocolos[2].
- Por exemplo, o sistema de software do instituto meteorológico contém dados meteorológicos diários. A aplicação para a previsão do tempo em seu celular "conversa" com esse sistema usando APIs e mostra atualizações meteorológicas diárias no telefone.

[2] AWS. O que é uma API?. Disponível em: https://aws.amazon.com/pt/what-is/api/. Acesso em: 05 mar. 2023

Obter dados via API do SGS em JSON

a. Como carregar os dados do SGS via API:

1. Obtenha o código do SGS da série;
2. Primeira opção de URL (informar datas ou extrair toda a série):
   - Para extrair os valores da série em um intervalo de tempo ou toda a série disponível:
   `https://api.bcb.gov.br/dados/serie/bcdata.sgs.{codigo_serie}/dados?formato=json&dataInicial={dataInicial}&dataFinal={dataFinal}`
   - codigo_Serie (obrigatório): parâmetro numérico que representa o código da série a ser consultada (substituir `{codigo_serie}` pelo código);
   - dataInicial (opcional): parâmetro textual que representa a data de início da consulta, no formato dd/MM/aaaa (substituir `{dataInicial}` pela data inicial);
   - dataFinal (opcional): parâmetro textual que representa a data final da consulta, no formato dd/MM/aaaa (substituir `{dataFinal}` pela data final).

Obter dados via API do SGS em JSON

Exemplos:
- Com datas: `https://api.bcb.gov.br/dados/serie/bcdata.sgs.433/dados?formato=json&dataInicial=01/01/2022&dataFinal=01/01/2023` (remover as aspas das datas).

- Sem datas: `https://api.bcb.gov.br/dados/serie/bcdata.sgs.433/dados?formato=json`

Obter dados via API do SGS em JSON

```
library(jsonlite)
# url da consulta
url_ipca <- "https://api.bcb.gov.br/dados/serie/bcdata.sgs.433/dados?
formato=json&dataInicial=01/01/2022&dataFinal=01/01/2023"
# baixar os dados
sgs_ipca <- fromJSON(url_ipca)
# converter a coluna de valor em numeric

sgs_ipca$valor <- as.numeric(sgs_ipca$valor)
```

Obter dados via API do SGS em JSON

c. Segunda opção de URL (extrair os últimos valores):

- `https://api.bcb.gov.br/dados/serie/bcdata.sgs.{codigo_serie}/dados/ultimos/{N}?formato=json`
  N: extrair os N últimos valores da série.
  Exemplo:
  `https://api.bcb.gov.br/dados/serie/bcdata.sgs.433/dados/ultimos/10?formato=json`

Obter dados via API do SGS em JSON

```
# url da consulta
url_ipca <-
"https://api.bcb.gov.br/dados/serie/bcdata.sgs.433/dados/ultimos/10?formato=json"
# baixar os dados
sgs_ipca <- fromJSON(url_ipca)
# converter a coluna de valor em numeric

sgs_ipca$valor <- as.numeric(sgs_ipca$valor)
```

Obter dados via API do SGS em CSV

```
# url da serie do SGS
url_sgs <- "https://api.bcb.gov.br/dados/serie/bcdata.sgs.433/dados/
ultimos/10?formato=csv"
# criar arquivo temporário de destino
nome_arq <- paste0(tempfile("sgs_arq"), ".csv")
# baixar o arquivo
download.file(url_sgs, nome_arq, mode="wb")
# carregar o arquivo

ipca <- read.csv(nome_arq, dec = ",", sep=";")
```

Obter dados via API do SGS

- `paste0()` serve para colar dois textos, sem definir um caracter delimitador.
- `tempfile()` cria um arquivo temporário.

Exercício 4 (2 min.)

a. Crie a URL para baixar a série do SGS "Índice de Commodities - Brasil - Agropecuária (em US Dólares)" (série nr. 29041) no período de jan/19 a jan/23, no formato JSON;

b. Carregue a série para memória do R usando o *package* `jsonlite`.

Exercício 5 (2 min.)

a. Crie a URL para baixar a série do SGS "Balança comercial - mercadorias em geral - Balanço de Pagamentos - anual - saldo" (série nr. 23470) no período de 2010 a 2022, no formato CSV;

b. Crie um arquivo temporário.

c. Baixe a série citada e salve no arquivo temporário;

d. Carregue a série para a memória do R.

Obter dados via API do Focus

- API do Focus:
  `https://dadosabertos.bcb.gov.br/dataset/expectativas-mercado`

Obter dados via API do Focus

- Tela para criar a URL dos dados desejados:

Obter dados via API do Focus

```
# url da série do Focus
url_focus <- "https://olinda.bcb.gov.br/olinda/servico/Expectativas/
versao/v1/odata/ExpectativaMercadoMensais?$top=24&$filter=Indicador%20eq%20'
C%C3%A2mbio'%20and%20baseCalculo%20eq%200&$orderby=Data%20desc&$format=json
&$select=Indicador,Data,DataReferencia,Mediana"
# carregar o arquivo
cambio <- fromJSON(url_focus)
# extrair o arquivo de dentro da lista

cambio <- cambio$value
```

Obter dados via API do Focus

a. Preenchimento dos campos da URL usados no exemplo:

- Máximo: 24 (a projeção é realizado para o horizonte de 24 meses);
- Filtro: `Indicador eq 'Câmbio' and baseCalculo eq 0` (obter o Focus da série "Câmbio" com base de cálculo igual a zero);
- Ordenação: `Data desc` (trazer por ordem decrescente de data, para obter o Focus mais recente);
- Saída: `json` (para obter os dados no formato JSON); e
- Campos: `Indicador`, `Data`, `Data de Referência` e `Mediana` (campos a serem extraídos).

Exercício 6 (4 min.)

a. Crie a URL para baixar as expectativas **anuais** do Focus "IPCA Administrados" referente à pesquisa mais recente disponível, no formato JSON.

b. Carregue a série para a memória do R e extraia ela de dentro da lista;

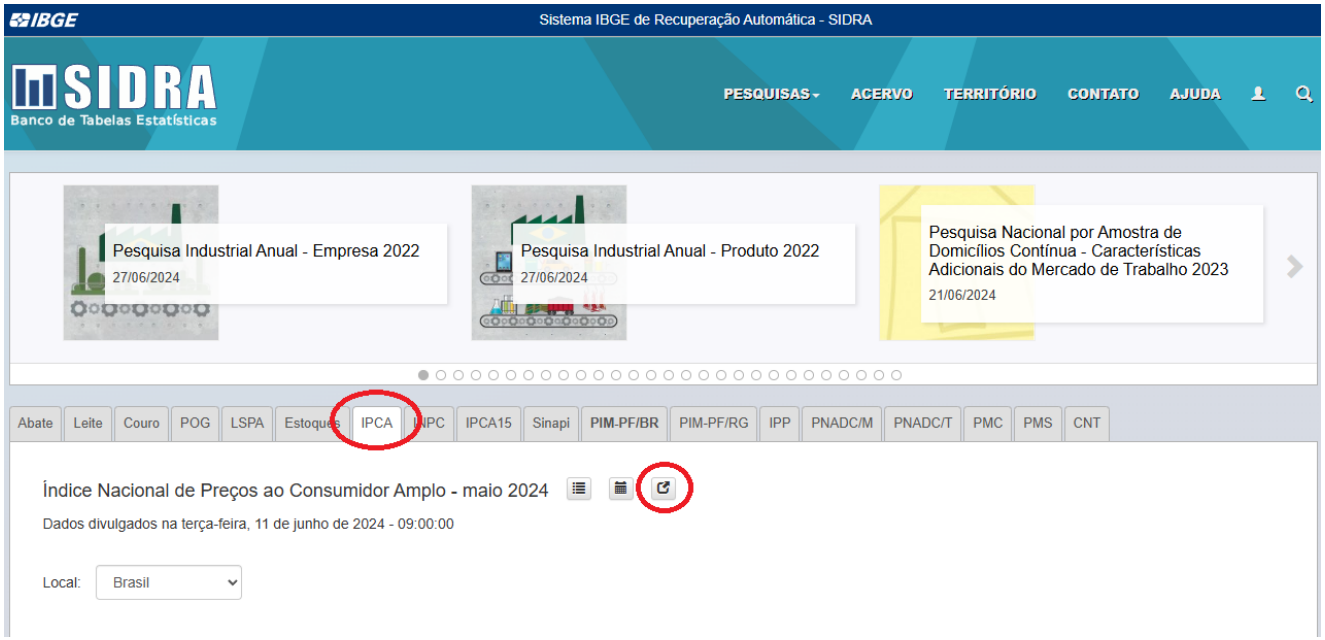
c. Faça um gráfico de linha da série.

Obter dados via API do IBGE

- A API do IBGE é chamada de SIDRA (Sistema IBGE de Recuperação Automática) (https://sidra.ibge.gov.br/home/ipp/brasil);
- A extração de dados pode ser feita usando o pacote `sidrar`.

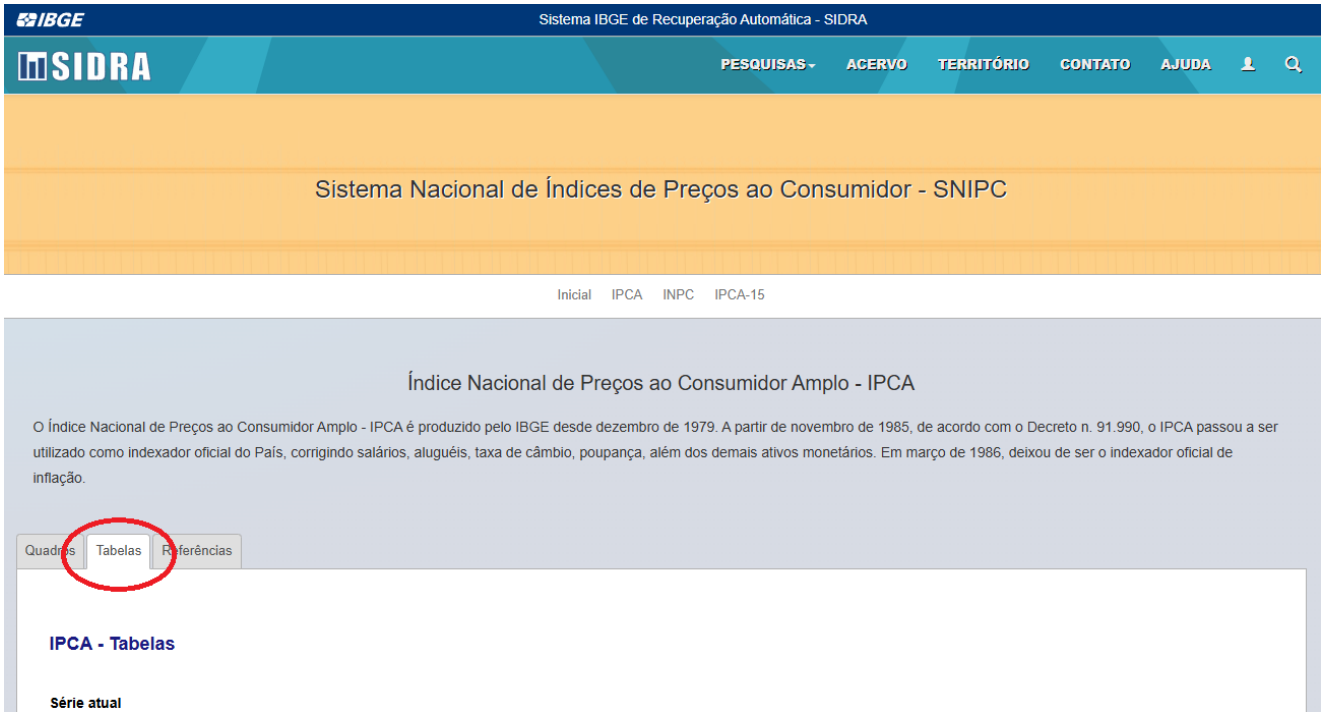
Como criar a URL da API do IBGE

Passo 1. Escolha o indicador (no exemplo abaixo, foi escolhido o IPCA) e clique no botão “Ir para a página de pesquisa”.



Como criar a URL da API do IBGE

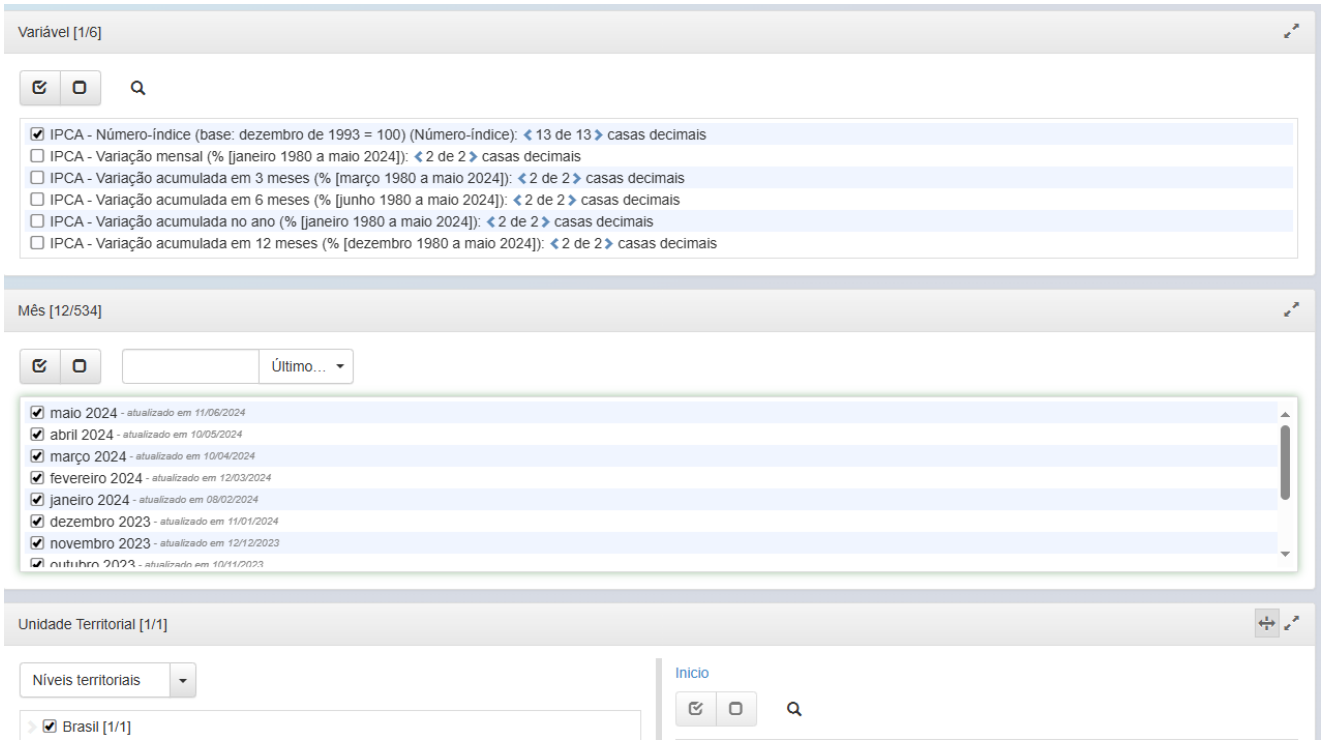
Passo 2. Clicar na aba “Tabelas”.



Como criar a URL da API do IBGE

Passo 3. Escolher a tabela que contém os dados desejados.

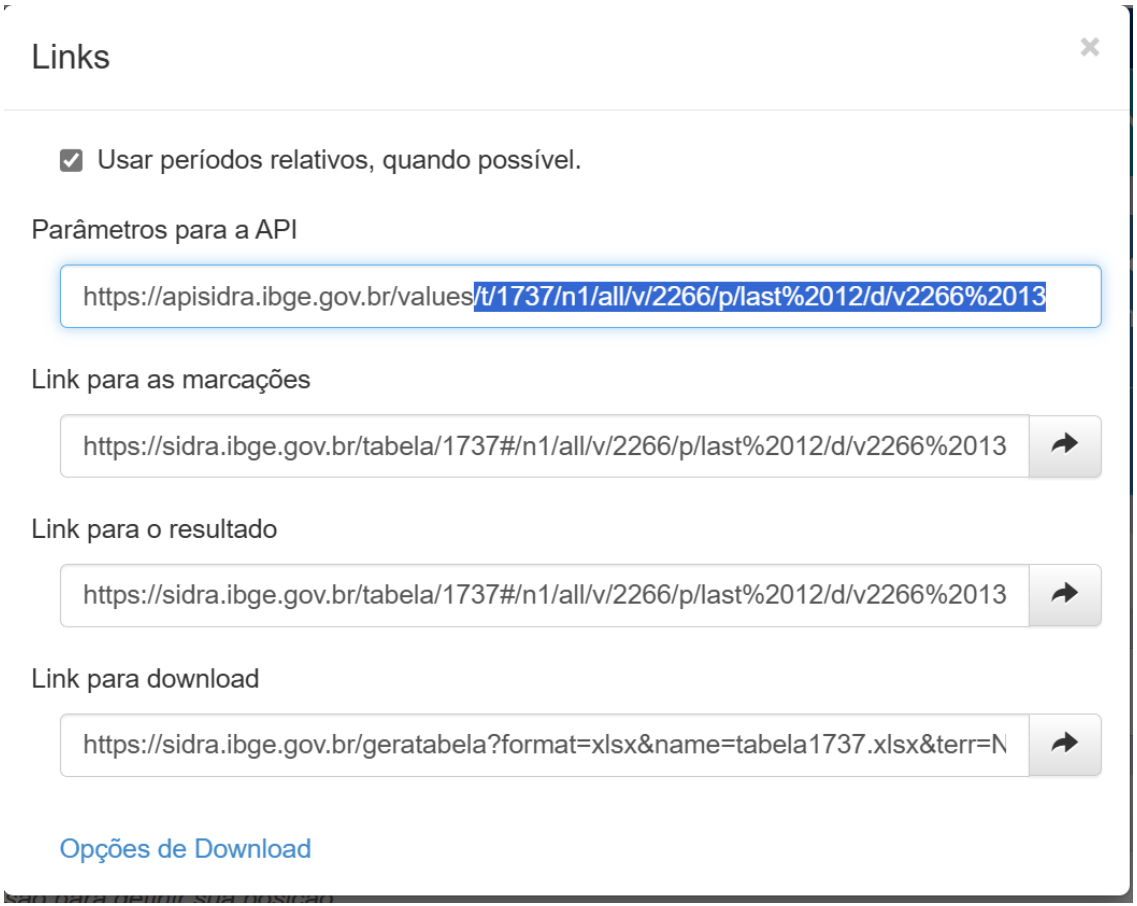
Passo 4. Após escolher a tabela, escolher o(s) indicador(es), período dos dados e nível territorial.



Como criar a URL da API do IBGE

Passo 5. Clicar no botão "Links de Compartilhar" (parte inferior direita da página).

Passo 6. Na caixa "Parâmetros para a API", copiar a parte da URL que começam em `/t` (incluindo estes). Esta é a URL para extração dos dados.



Obter dados via API do IBGE

Colar a URL no argumento `api` da função `get_sidra` do pacote `sidrar`.

Exemplo de extração de dados do IPCA:

```
library(sidrar)
# retornar os dados do SIDRA

ipca <- get_sidra(api = "/t/1737/n1/all/v/2266/p/last%2012/d/v2266%2013")
```

Exercício 7 (2 min.)

a. Utilizar o *package* `sidrar` para carregar no R a população, por grupo de idade (marcar todos os grupos), em mil pessoas, no nível territorial do Brasil, da data mais recente disponível da Pesquisa Nacional por Amostra de Domicílios Contínua Trimestral - PNADC/T.

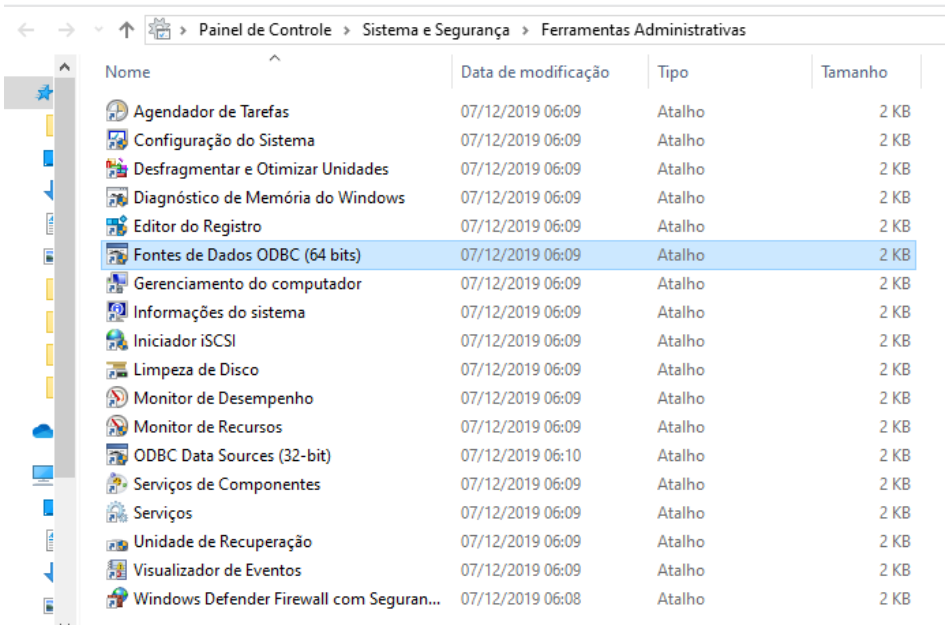
Package rbcb

- O *package* rbcb permite extrair as séries do SGS e do Focus (https://cran.r-project.org/web/packages/rbcb/index.html).

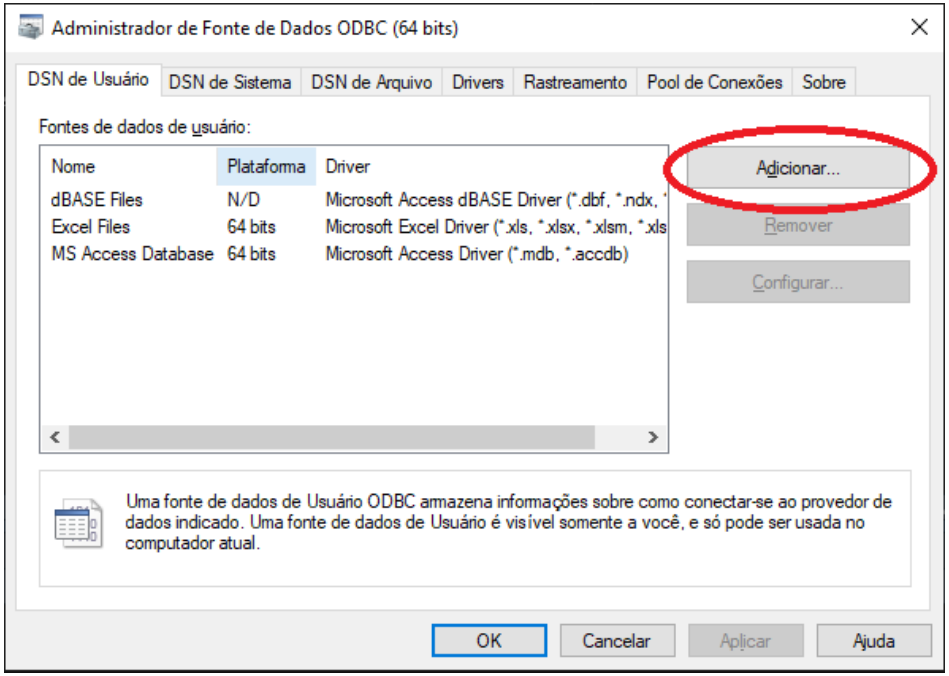
```
library(rbcb)
# obter todas as moedas em uma data
get_all_currencies("2023-03-09")
# obter as expectativas anuais do IGP-M
get_annual_market_expectations("IGP-M", start_date = "2023-03-01")
# obter os dados do IPCA mensal
get_series(433, start_date = "2022-11-01")
```

Obter dados do servidor SQL

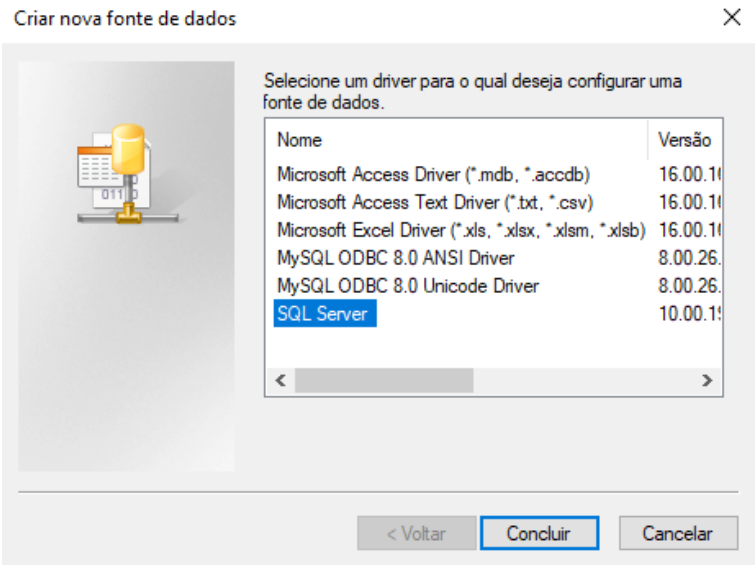
- O R consegue extrair dados de servidores SQL.
- O primeiro passo é criar uma conexão ODBC com o servidor SQL usando a ferramenta do Windows “Fontes de Dados ODBC (64 bits)”:

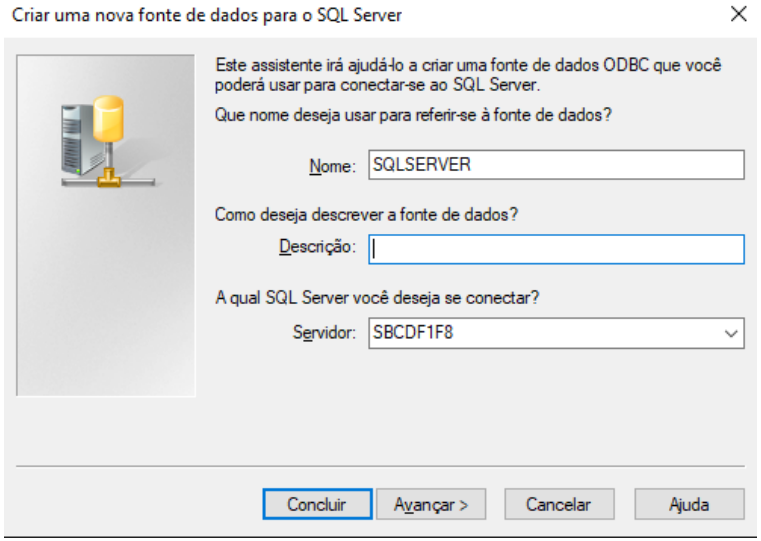


Obter dados do servidor SQL



Obter dados do servidor SQL





Obter dados do servidor SQL

a. Instalar os pacotes `DBI` e `odbc`;

b. Executar consulta dentro do servidor:

```
library(DBI)
# estabelecer a conexao com o servidor SQL
cdbcon <- DBI::dbConnect(odbc::odbc(),
driver = "SQL Server",
server = "SBCDF1F8",
trusted_connection = "yes")
# executar uma query no servidor
tabela <- dbSendQuery(cdbcon, "select top 10 * from
DEPEC_PROJECOES_DP.db_coace.Series")
# gerar o data.frame dos dados
tabela <- dbFetch(tabela)
```

Obter dados do servidor SQL

c. Carregar uma tabela inteira e filtrar ela usando o pacote dplyr:

```
library(dbplyr)
library(dplyr)
# carregar a tabela do SQL
tab_sql <- tbl(cdbcon,
in_catalog("DEPEC_PROJECOES_DP","db_coace","Series"))
# filtrar com o pacote dplyr
tab_sql_2 <- tab_sql %>% filter(Pacote == "PIM",AjusteSazonal == "OBS")
# converter a tabela em data.frame usando a função as.data.frame
tab_sql_2 <- as.data.frame(tab_sql_2)
```

Obter dados de arquivo Access

- Podem ser utilizadas as mesmas funções usadas para conectar em um servidor SQL para obter dados de arquivo Access.
- A diferença é que deve ser criada uma conexão ODBC com o arquivo Access: