Análise Macro Treinamento e Consultoria em Dados LTDA

EBOOK 01 - CLUBE AM

Como se tornar um Economista Quantitativo?

Vítor Wilher

Fundador e Cientista-Chefe da Análise Macro

analisemacro.com.br

setembro de 2021

Sobre o Ebook

A Análise Macro é um sonho acalentado durante muito tempo. Acreditamos que a teoria não pode estar dissociada da prática. Por isso, em março de 2011, nasceu o nosso blog, cujo objetivo era prover análises das mais variadas áreas e indicadores da conjuntura econômica para estudantes e professores interessados em aprimorar seus conhecimentos práticos. No início de 2015, passamos a publicar exercícios macroeconométricos regulares usando o R como linguagem de programação. Recebemos muitos comentários, elogios e pedidos de scripts.

A boa aceitação desses exercícios nos fez perceber que nossos leitores tinham grande interesse por ferramentas estatísticas e econométricas. Eles adoravam ver como a teoria pode andar em conjunto com os dados! Não tivemos dúvidas: era hora de ofertar um curso de introdução à análise de dados que fosse diferente, que fizesse a ponte entre a teoria e a prática, que fosse aplicado e intuitivo para quem nunca teve contato, nem jamais pensou em aprender esse tipo de coisa. Nasceu o nosso carro-chefe: o Curso de Introdução ao R. Com mais de 2000 alunos formados, temos enorme prazer em perceber o impacto que o R, a linguagem criada para data analysis, teve em suas vidas.

Seis anos se passaram desde que abrimos a primeira turma. Atualmente, temos mais de vinte Cursos Aplicados de R (e Python) em nossa plataforma on-line e já participamos de inúmeros projetos de consultoria em análise de dados. Somos uma das empresas pioneiras na oferta de Cursos de R no Brasil.

Para comemorar essa trajetória e o nosso sexto aniversário em setembro de 2021, estamos lançando o primeiro `Ebook` gratuito do Clube AM, a Comunidade da Análise Macro que tem por objetivo compartilhar com os membros o passo a passo em vídeos, pdfs e scripts dos exercícios de análise de dados que desenvolvemos por aqui.

O `Ebook` mostra alguns exemplos do que é compartilhado no Clube AM toda semana. Lá, divulgamos as publicações da Análise Macro, nossos códigos envolvendo análise de dados séria e de qualidade.

Ao longo do `Ebook`, o leitor poderá ver através de exemplos práticos como utilizar o **R** para construir análises quantitativas que fazem parte do dia a dia dos milhares de profissionais que temos o prazer de ter ajudado a formar.

Sobre o Autor

Vítor Wilher é Bacharel e Mestre em Economia, pela Universidade Federal Fluminense e Especialista em Data Science. Sua dissertação de mestrado foi na área de política monetária, titulada *Clareza da Comunicação do Banco Central e Expectativas de Inflação: evidências para o Brasil*, defendida perante banca composta pelos professores Gustavo H. B. Franco (PUC-RJ), Gabriel Montes Caldas (UFF), Carlos Enrique Guanziroli (UFF) e Luciano Vereda Oliveira (UFF). Já trabalhou em grandes empresas, nas áreas de telecomunicações, energia elétrica, consultoria financeira e consultoria macroeconômica. Atualmente, ocupa o cargo de Cientista-Chefe da Análise Macro, responsável pela área de Cursos e Consultoria, além de ser Sócio-Fundador da empresa. Seu portfólio está disponível em vitorwilher.github.io. Você pode contactá-lo pelo e-mail vitorwilher@analisemacro.com.br.

Advertência

Tentei deixar o código o mais `reprodutível` possível, porque tenho o conceito como filosofia de trabalho. Mas, por óbvio, é sempre importante ressaltar que é possível que no momento que você rodar o código, alguem erro inesperado possa ocorrer. Em geral, os possíveis principais problemas para isso estão em um pacote que não está mais disponível no CRAN ou algum dado que não está mais disponível na fonte utilizada no código.

Os Membros do Clube AM, a propósito, podem compartilhar suas dúvidas com o grupo fechado no Whatsapp, bem como com a Equipe da Análise Macro.

Sumário

1 Como se tornar um economista quantitativo?

Sempre me perguntam qual seria a grade ideal de um Curso de Economia. Acho que cada economista, estudante ou professor vai ter a sua. A minha tem o objetivo de formar o que eu chamo de `economista quantitativo`. É uma espécie de antítese ao que alguns colegas e professores que admiro chamam de `economista alternativo`. E vou procurar explicar nesse espaço o porquê, mostrando a ementa que considero a mais ideal para formar esse tal `economista quantitativo`. Vamos lá?

Antes de mais nada, é preciso dizer que *economia*, em qualquer lugar do mundo, *é uma disciplina menor*, que procura entender e resolver *problemas práticos*. Para isso, o economista deve construir, estimar e interpretar modelos. Então, o dia a dia de um economista passa por ler *papers*, para entender a literatura sobre um determinado assunto, construir ou pegar emprestado algum modelo teórico e estimá-lo empiricamente.

Economista não é, nessa abordagem, um filósofo, que fica divagando sobre os problemas do mundo. Não. Economia, por si só, já é um baita problema difícil. Sejam os problemas macro, sejam os problemas microeconômicos. Por isso, é preciso se concentrar em aprender teoria e instrumental para lidar com eles.

Para fazer isso hoje em dia, você vai precisar de *um curso de linguagem de programação voltada para data analysis*, como o R (ou o Python). Esse curso vai te ajudar a lidar com dados. Vai precisar, ainda, de dois cursos de Cálculo, um curso de Equações Diferenciais, um de Álgebra Linear, quatro cursos de teoria microeconômica e outros quatro cursos de teoria macroeconômica. Para fechar, dois cursos de Estatística e três de econometria. Pronto, está formado o que o mundo entende quando a palavra *economista* é pronunciada.

Será sempre possível acrescentar a esses cursos obrigatórios alguns optativos, em finanças, em econometria, macro, micro, história do pensamento econômico, história econômica, etc. Mas eu - opinião pessoal - acho um equívoco não ter aquela sequência de cursos antes de qualquer coisa. Saber lidar com dados é algo essencial hoje em dia e um economista que não sabe fazer isso não pode ter o título de economista. Ele vira um economista alternativo.

Em resumo, um bom economista, com aceitação no mercado, deve cumprir as seguintes condições:

1. Entender a teoria econômica;
2. Saber estimar modelos econométricos;
3. Saber as limitações dos modelos que estima;
4. Ter bom senso.

Infelizmente, entretanto, a maioria das nossas faculdades de economia não consegue entregar para o mercado esse profissional. Seja a nível de graduação ou mesmo pós-graduação. Em geral, as grades de economia estão cheias de disciplinas teóricas ou de história do pensamento econômico, com pouco ou nenhum espaço para disciplinas aplicadas que ensinam o aluno a trabalhar com dados. Exigência, diga-se, cada vez maior do mercado de trabalho.

Nesse `Ebook`, por suposto, procurei mostrar a relevância desse tipo de conhecimento através de exemplos práticos do dia a dia um `economista quantitativo`. A marca comum desses exercícios é a *automatização do processo de coleta, tratamento, análise e apresentação de dados* através do R, uma das linguagens de programação mais utilizadas hoje em dia.

O objetivo principal disso, por suposto, é que você veja na prática como é possível ir além do ensinado nas graduações e pós-graduações país a fora, tornando-se um profissional disputado no mercado de trabalho.

Alerto desde já que, o caminho para `se tornar um economista quantitativo` não é simples, mas bastante gratificante. Atualmente, o mercado de trabalho para esse tipo de profissional está em franca expansão. O `Economista Quant` tem vaga não apenas no mercado de economia, como também tem sido absorvido cada vez mais no mercado de dados, onde seus *insights* são muito bem vistos pelas empresas. Há milhares de vaga disponíveis em consultorias, grandes empresas, bancos e institutos de pesquisa.

Por onde começar afinal? A gama de conhecimentos necessários para se tornar esse profissional é hoje coberta de forma plena pela grade de Cursos da Análise Macro. A trilha `Começando a lidar com dados` contém cursos iniciais, que podem ser feitos por estudantes de graduação e pós-graduação, professores e profissionais de mercado que ainda não dominam o R.

Uma vez **iniciado**, o aluno pode se especializar em uma das cinco trilhas disponíveis: econometria, macroeconomia aplicada, microdados, finanças quantitativas e central banking. Cada uma delas irá formar um profissional capacitado a lidar com os mais variados desafios do mundo contemporâneo.

O aprendizado dentro da Análise Macro não pára por aí. Nosso compromisso com a prática é inegociável. Dentro do Clube AM, a Comunidade da Análise Macro, o membro tem acesso semanal a novos exercícios, como esses que você tem acesso nesse `Ebook`, de modo que estará sempre em contato com exercícios de análise de dados, vídeos explicativos, pdfs e scripts comentados.

Além disso, poderá ainda tirar dúvidas com a nossa Equipe, bem como fazer networking com os demais membros do Clube através do grupo fechado no Whatsapp.

Nesse `Ebook`, por suposto, damos a você uma pequena amostra de tudo que compartilhamos dentro do Clube AM. Esperamos, assim, que você se sinta impelido com esses exercícios práticos a vislumbrar uma nova carreira ou simplesmente dar uma guinada na sua carreira atual, adicionando uma ferramenta poderosa de análise de dados no currículo.

Seja bem-vindo ao Clube!

2 Introdução ao R

2.1 Por que aprender uma linguagem de programação?

Eu comecei a utilizar o **R** há pouco mais de sete anos, influenciado por amigos. Minha introdução ao mundo dos programas estatísticos foi através do Eviews, ainda nos tempos da graduação em economia, como provavelmente muitos dos que fazem economia. Ainda que seja possível *escrever códigos* no Eviews e em outros pacotes estatísticos fechados (que precisam de licença), as vantagens do **R** são inúmeras, como comentarei mais à frente. Por ora, talvez seja necessário tecer algumas palavras sobre por que afinal é preciso aprender uma linguagem de programação.

Eu poderia falar que o mundo está mudando, que cada vez mais empregos e empresas têm exigido conhecimentos de programação. E isso de fato é verdade, o que por si só gera uma necessidade de saber programação. Mas, aqui entre nós, acho que é meio chato aprender algo por necessidade, não é mesmo?

Minha motivação para aprender a programar foi de fato outra. Eu estava um pouco cansado de *apertar botões* e fazer tarefas repetitivas em pacotes estatísticos como o Eviews, então parecia natural aprender uma forma de *automatizar* as coisas. Essa, afinal, era uma baita motivação para mim e talvez também seja para você. Mas por que o **R**, você pode perguntar.

Essa é de fato uma boa pergunta. Por que não aprender a escrever códigos no próprio Eviews? E a resposta é bastante simples. Você já tentou encontrar algum código em Eviews na internet? E sobre **R**? Pois é. Entre as inúmeras vantagens do **R**, posso destacar:

- A existência de uma comunidade grande e bastante entusiasmada, que compartilha conhecimento todo o tempo;
- o **R** é gratuito, *open source*, de modo que você não precisa comprar licenças de software para instalá-lo;
- Tem inúmeras bibliotecas (pacotes) em estatística, *machine learning*, visualização, importação e tratamento de dados;
- Possui uma linguagem estabelecida para *data analysis*;
- Ferramentas poderosas para comunicação dos resultados da sua pesquisa, seja em forma de um website ou em pdf.

Ao aprender **R**, você conseguirá integrar as etapas de coleta, tratamento, análise e apresentação de dados em um único ambiente. Você vai esquecer ter de abrir o excel, algum pacote estatístico, depois o

power point ou o word, depois um compilador de pdf para gerar seu relatório. Todas essas etapas serão feitas em um único ambiente. E essa talvez seja a grande motivação para você entrar de cabeça nesse mundo.[1]

Certamente não será simples, tenho de lhe dizer. Haverá muitos momentos em que você pensará em desistir. Um erro inesperado em algum *script* poderá lhe tirar horas do seu tempo. No início, mais especificamente, qualquer pequeno problema pode parecer uma barreira intransponível. Nesses momentos, minha dica é *não se demorar muito nessas dificuldades*. Vá para outro problema ou mesmo descanse. Faça outras coisas. Depois volte mais tranquilo, procure nos canais de ajuda que colocaremos aqui e as coisas irão se acertar. Acredite: passado esse tempo inicial, com persistência, você certamente se beneficiará muito em ter aprendido uma linguagem de programação como o **R**.

Nosso objetivo na Análise Macro, com nossos Cursos Aplicados de R é justamente lhe ajudar nesse processo. Todos os nossos cursos apresentam sólida teoria, mas sempre recheada de exercícios e exemplos práticos. Nosso objetivo é trazer para o Brasil algo que já é bastante corriqueiro no resto do mundo: *você deve aprender fazendo*. Isso vai tornar o aprendizado mais interessante, mais divertido até.

Isso dito, vamos começar então? Nessa primeira seção, você verá alguns procedimentos básicos, sobre programas e um *overview* do **R**. A partir daí, você verá diversos exemplos de como usar o **R** para fazer análise de dados!

Seja bem vindo a esse mundo e espero, sinceramente, que você se beneficie bastante dessa nova fonte de conhecimento.

2.2 Instalando os programas

Antes de tudo, é preciso que você tenha os programas que utilizaremos por aqui instalados em seu computador. Serão dois programas: **R** e **RStudio**. Não se preocupe, posto que são todos programas gratuitos e com *download* seguro. Desse modo, para que não tenhamos problemas, siga a sequência abaixo:

1. Baixe o **R** em http://cran-r.c3sl.ufpr.br/;
2. Baixe o **RStudio** em https://www.rstudio.com/products/rstudio/download/;

[1] Para maiores detalhes sobre esse ponto, ver Grolemund and Wickham (2017).

2.3 Trabalhando com o R a partir do RStudio

Ao longo desse `Ebook`, nós não trabalharemos diretamente no **R**. Ao invés disso, usaremos o **RStudio**, uma interface mais amigável, que nos permite emular todos os códigos do **R**, visualizar gráficos, ver o histórico de nossas operações, importar dados, criar scripts, etc. Com o **RStudio**, poderemos otimizar o nosso processo de análise de dados, de maneira que você tenha mais facilidade para interagir com a linguagem.

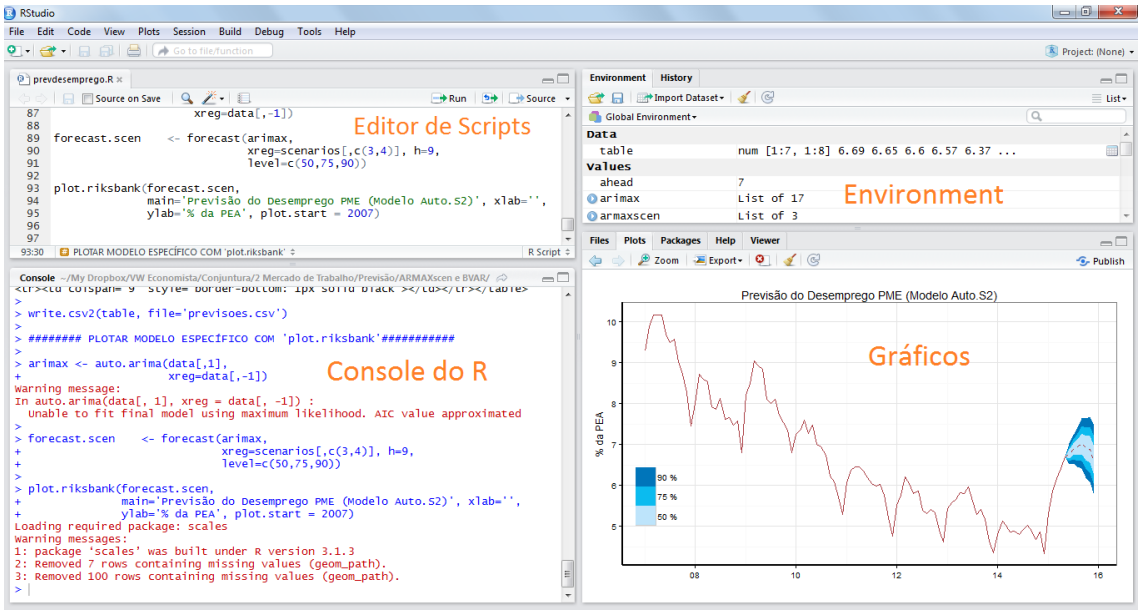


Figure 1: Ambiente do RStudio.

A figura acima resume as quatro principais partes de uma tela do **RStudio**. Na parte superior esquerda é onde ficará o nosso *editor de scripts*. Um *script* é uma sequência de comandos com um determinado objetivo. Por exemplo, você pode estar interessado em *construir um modelo univariado para fins de previsão do índice BOVESPA*. Para isso, terá de primeiro importar os dados do Ibovespa, bem como fazer uma análise descritiva inicial dos dados. Depois, com base nessa análise inicial, você terá de decidir entre alguns modelos univariados distintos. De posse da sua decisão, você enfim construirá um modelo de previsão para o índice BOVESPA. Essa sequência de *linhas de comando* pode ficar armazenada em um *script*, com extensão .R, podendo ser acessada posteriormente por você mesmo ou compartilhada com outros colegas de trabalho. Para abrir um novo *script*, vá em *File*, *New File* e clique em *R Script*.

Na parte inferior esquerda, está o *console do R*, onde você poderá executar comandos rápidos, que não queira registrar no seu script, bem como será mostrados os *outputs* dos comandos que você executou no seu script.

Já na parte superior direita, está o *Environment*, onde ficam mostrados os objetos que você cria ao longo da sua seção no RStudio. Por fim, na parte inferior direita, ficarão os gráficos que você solicitar, bem como pacotes que você instalou, alguma ajuda sobre as funções e os arquivos disponíveis no seu diretório de trabalho.

2.4 Definindo seu diretório de trabalho

Agora que você já instalou os programas e já conhece um pouco do ambiente do **RStudio**, podemos começar a brincar um pouco. Para isso, antes de mais nada é preciso definir o seu *diretório de trabalho* ou a pasta onde ficará salvo o seu *script*. Uma vez definido, você poderá importar arquivos, colocar figuras no seu documento LaTeX, etc. Logo, dois comandos são importantes para isso. O primeiro é o **getwd**, para você ver o seu atual *working directory*. O segundo é o **setwd**, para você *setar* o seu diretório de trabalho[2]

```
getwd()
```

[1] "C:/Users/Caroline/Dropbox/Nova Análise Macro/5. Produtos/3. Clube AM/14. Ebook"

```
setwd('C:/Users/Vítor Wilher/Dropbox/VW Economista')
```

Uma vez *setado* o seu diretório de trabalho, você poderá importar dados contidos naquela pasta. Assim, é um ponto importante realizar isso antes de qualquer coisa. Caso já tenha um *script* de R, por suposto, uma vez abrindo-o, o RStudio setará automaticamente o diretório onde o mesmo esteja.

2.5 Uma linguagem orientada a objetos

O **R** é uma linguagem orientada a objetos, de modo que o que você fará dentro do programa será, basicamente, manipulá-los. Seja lidando com objetos criados por terceiros, seja criando seus próprios objetos. As principais estruturas de dados dentro do **R** envolvem *vetores*, *matrizes*, *listas* e *data frames*. Abaixo colocamos um exemplo da estrutura mais simples do **R**: um vetor que exprime a sequência de 1 a 10.

```
vetor <- c(1:10)
vetor
```

[1] 1 2 3 4 5 6 7 8 9 10

[2]Como você verá ao longo do nosso curso, as funções dentro do **R** são todas em inglês e bastante intuitivas, como *get something* ou *set something*.

2.6 Milhares de pacotes a sua disposição

O **R** é uma linguagem aberta, onde qualquer pessoa em qualquer parte do mundo pode dar a sua contribuição. Em geral, elas fazem isso através de *pacotes*, que são coleções de funções que fazem algum coisa dentro do **R**. Veremos muitos desses pacotes ao longo do nosso curso. A instalação de pacotes é feita primariamente pelo CRAN, através da seguinte função:

```
install.packages('tidyverse')
```

O pacote é instalado corretamente quando aparece a mensagem *package 'fBasics' successfully unpacked and MD5 sums checked* no seu console. Ademais, no processo de instalação de um pacote, pode ser necessário instalar outros pacotes, chamados de *dependentes*, porque uma ou mais funções do pacote que você quer instalar fazem uso de funções de outros pacotes. Assim, **para que a instalação seja efetuada com sucesso, é preciso que todos os pacotes dependentes sejam instalados corretamente.** Por isso, procure verificar as mensagens no console de forma a verificar se o pacote foi instalado corretamente, como mostrado na figura 2.

```
> install.packages('fBasics')
Warning in install.packages :
  cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/src/contrib/PACKAGES.rds'
: HTTP status was '404 Not Found'
Installing package into 'C:/Users/Vítor Wilher/Documents/R/win-library/3.4'
(as 'lib' is unspecified)
Warning in install.packages :
  cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.4/P
ACKAGES.rds': HTTP status was '404 Not Found'
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.4/fBasics_3011.87.zi
p'
Content type 'application/zip' length 1559813 bytes (1.5 MB)
downloaded 1.5 MB

package 'fBasics' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Vítor Wilher\AppData\Local\Temp\Rtmp6rMrmM\downloaded_packages
```

Figure 2: Instalando pacotes.

Uma vez instalado, os seus pacotes ficam armazenados na pasta *library* da versão correspondente do seu R.[3] Você pode ver a lista de pacotes naturamente, diretamente no RStudio, através da aba *Packages* no canto inferior direito.[4]

Uma outra forma muito comum de instalar pacotes é através do **GitHub**, uma plataforma bem bacana utilizada por desenvolvedores para compartilhar códigos. Ali ficam armazenados pacotes *em*

[3]Toda vez que você instalar uma nova versão do R, uma dica é pegar os pacotes da sua pasta library e copiar os mesmos para a pasta library da nova versão, de modo a não ter que instalar tudo novamente.

[4]Uma lista dos pacotes disponíveis pode ser encontrada aqui.

desenvolvimento, que ainda não estão disponíveis no CRAN. Para instalar um pacote via GitHub, você deve ter instalado primeiro o pacote *devtools*. O código abaixo exemplifica com a instalação do pacote brasileiro BETS.[5]

```
install.packages('devtools')
require(devtools)
install_github("pedrocostaferreira/BETS")


if(!require(devtools)) install.packages("devtools")
```

No código acima, nos instalamos o pacote *devtools*, depois **carregamos** o mesmo com a função `require` - também pode ser utilizado a função `library` - e então utilizamos a função `install_github` para instalar um pacote armazenado no GitHub.

Às vezes pode ser necessário instalar uma versão antiga de um pacote, seja porque a versão atual tem algum *bug* - acontece! - seja porque ela é incompatível com o pacote que queremos usar. Para instalar versões antigas, você tem duas opções. Uma é subir o arquivo fonte zipado do pacote diretamente no RStudio, na opção *Tools* e *Install Packages*. Outra é utilizar a função `install_version` do pacote *devtools*, como abaixo.

```
install_version("DBI", version = "0.5", repos = "http://cran.us.r-project.org")
```

É muito comum os alunos receberem o erro de *caminho inválido* para a biblioteca de pacotes. Nesse caso, você pode especificar o caminho da biblioteca com o código abaixo.

```
library(withr) # É instalado e carregado junto com o devtools
withr::with_libpaths(new = "C:/Program Files/R/R-3.4.1/library", install_github("StatsWithR/sta
```

Por fim, vale ressaltar que todo pacote disponível no CRAN tem uma página com suas informações. Veja o exemplo do *devtools* aqui. Lá você pode ver um resumo do pacote, um pdf com as principais funções, versões antigas, etc.

2.7 Obtendo ajuda

Uma parte importante de aprender uma nova linguagem é saber conseguir resolver os pepinos que irão surgir ao longo do caminho. E, acredite: eles serão muitos! Mas não se desespere. Como há uma comunidade incrível trabalhando com **R**, existem inúmeros sites, blogs, tutoriais, apostilas, etc, que

[5]Para saber mais sobre o BETS, vá em https://github.com/pedrocostaferreira/BETS.

podem lhe ajudar. No próprio ambiente do **RStudio**, você pode invocar ajuda com os comandos abaixo.[6]

```
help.start() # Você terá acesso à página de ajuda do R.
help("read.csv") # Uma ajuda sobre a função 'read.csv'
?read.csv # A mesma coisa do comando acima.
```

Caso continue com dúvidas, entretanto, nada melhor do que o bom e velho Google. Recomendo que você jogue sempre sua dúvida lá, antes de qualquer coisa. Uma dica: jogue ela em inglês e verá logo na primeira página um monte de gente com o mesmo problema que você! Em particular, um fórum bastante conhecido é o Stackoverflow, onde pessoas mais experientes procuram ajudar os mais novos. Caso sua dúvida não seja resolvível via google, considere jogá-la nesse fórum. A versão em português do fórum está disponível em https://pt.stackoverflow.com/questions/tagged/r.

Outra coisa que você deve fazer a partir de agora é acompanhar blogs que falam de **R**. Modéstia ao largo, não se esqueça de acompanhar o meu próprio site www.analisemacro.com.br/blog. Lá faço alguns exercícios interessantes de como usar o **R** para resolver nossos problemas diários de análise de dados. No exterior, há uma infinidade de blogs reunidos no famoso R-bloggers.

[6]Observe que utilizamos o 'jogo da velha' # para colocar um comentário após a função. Isso é extremamente útil para que você lembre seus passos em um script no **R**, bem como no momento que você compartilhar seu script com outra pessoa.

3 Pacotes utilizados nesse `Ebook`

A seguir, carregamos os `pacotes` que utilizaremos ao longo do nosso `Ebook`.

```
# Carregar Pacotes
library(tidyverse)
library(zoo)
library(gridExtra)
library(scales)
library(lubridate)
library(forecast)
library(knitr)
library(tidyverse)
library(tsibble)
library(fable)
library(feasts)
library(tsibbledata)
library(fpp3)
library(readxl)
library(zoo)
library(quantmod)
library(timetk)
library(tidyquant)
library(rbcb)
library(corrplot)
library(stargazer)
library(grid)
library(png)
library(GetTDData)
library(ggcorrplot)
library(vars)
library(aod)
library(magrittr)
library(ggrepel)
library(xts)
library(ecoseries)
library(ipeadatar)
library(RcppRoll)
library(BETS)
```

```
library(forecast)
library(tidyverse)
library(timetk)
library(lmtest)
library(scales)
library(mFilter)
library(dynlm)
library(comtradr)
library(splitstackshape)
library(inlmisc)

pacman::p_load(
  "remotes",
  "GetBCBData",
  "GetTDData",
  "dplyr",
  "lubridate",
  "httr",
  "tidyr",
  "ggplot2",
  "scales",
  "magrittr",
  "jsonlite",
  "purrr",
  "stringr",
  "rvest",
  "officer",
  "flextable",
  "BCDating",
  "ggthemes"
)



if(!require("meedr")) remotes::install_github("schoulten/meedr")
```

4 Evolução da pandemia no Brasil

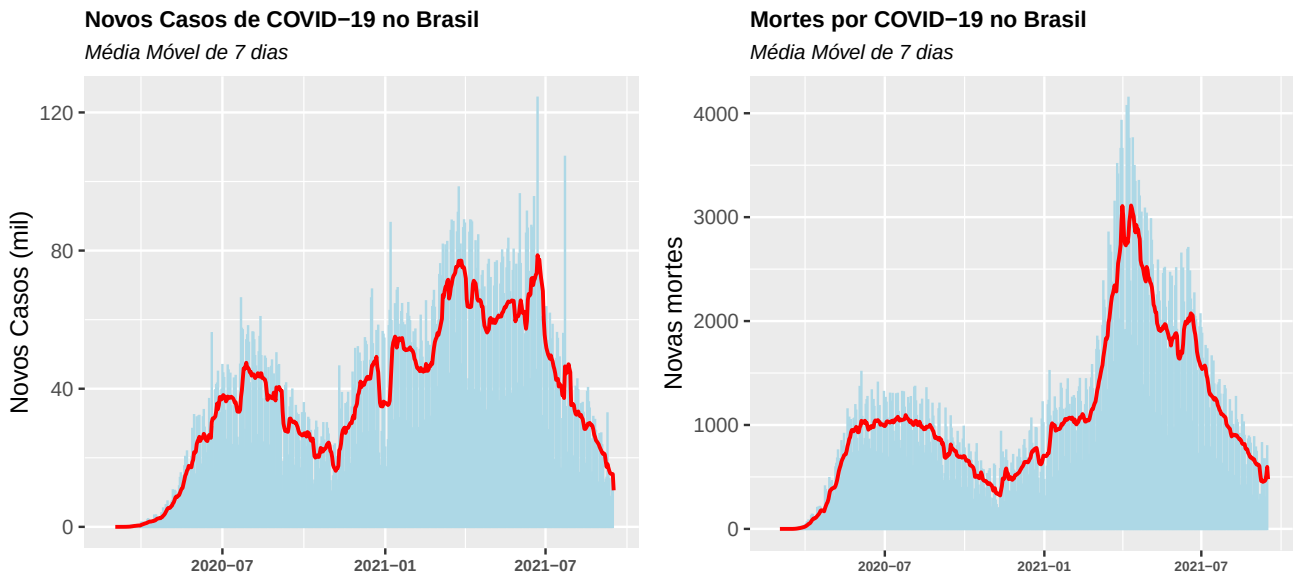
4.1 Coleta de dados

```
## Coletar dados
url = "https://raw.githubusercontent.com/wcota/covid19br/master/cases-brazil-states.csv"
covid = readr::read_csv(url, guess_max = 10000) %>%
  group_by(state) %>%
  mutate(MM_mortes = zoo::rollmean(newDeaths, k = 7, fill = NA, align = "right"),
         MM_casos = zoo::rollmean(newCases, k = 7, fill = NA, align = "right")) %>%
  mutate(d_vaccinated = vaccinated - lag(vaccinated,1)) %>%
  mutate(MM_dose1 = rollmean(d_vaccinated, 7, NA, align='right'))
```

4.2 Mortes e Novos Casos no Brasil

```
g1 = covid %>%
  dplyr::filter(state == "TOTAL") %>%
  ggplot(aes(x=date))+
  geom_bar(aes(y=newDeaths), stat='identity', fill='lightblue',
           colour='lightblue', width = .00001)+
  geom_line(aes(y=MM_mortes), colour='red', size=.8)+
  labs(x="",y="Novas mortes",
       title="Mortes por COVID-19 no Brasil",
       subtitle = 'Média Móvel de 7 dias')+
  theme(plot.title = element_text(size=10, face='bold'),
        axis.text.x = element_text(size=6, face='bold'),
        plot.subtitle = element_text(size=9, face='italic'))

g2 = covid %>%
  dplyr::filter(state == "TOTAL") %>%
  ggplot(aes(x=date))+
  geom_bar(aes(y=newCases/1000), stat='identity', fill='lightblue',
           colour='lightblue', width = .00001)+
  geom_line(aes(y=MM_casos/1000), colour='red', size=.8)+
  labs(x="",y="Novos Casos (mil)",
       title="Novos Casos de COVID-19 no Brasil",
       subtitle = 'Média Móvel de 7 dias')+
  theme(plot.title = element_text(size=10, face='bold'),
```

```
        axis.text.x = element_text(size=7, face='bold'),
        plot.subtitle = element_text(size=9, face='italic'))

grid.arrange(g2, g1, ncol=2,
             bottom='Fonte: analisemacro.com.br')
```

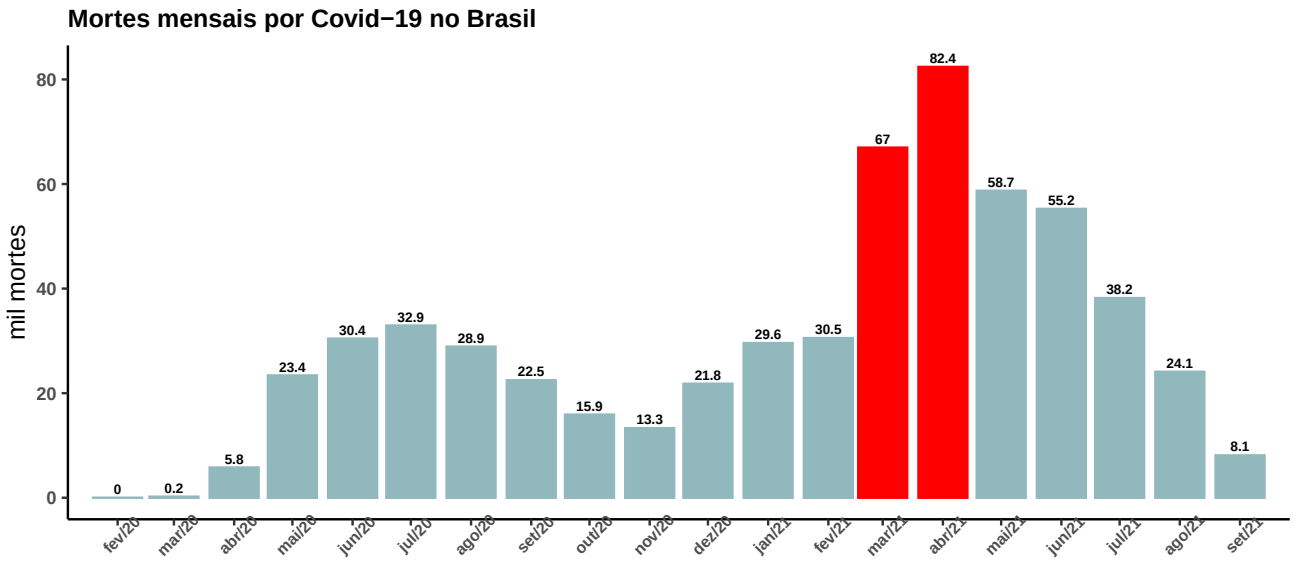


Fonte: analisemacro.com.br

4.3 Mortes mensais no Brasil

```
covid_mensal = covid %>%
  dplyr::select(date, state, newDeaths) %>%
  group_by(state, month = floor_date(date, 'month')) %>%
  summarise(newDeaths = sum(newDeaths))

covid_mensal_brasil = covid_mensal %>%
  dplyr::filter(state == 'TOTAL')

covid_mensal_brasil %>%
  ggplot(aes(x=month, y=newDeaths/1000))+
  geom_bar(stat='identity', fill=ifelse(covid_mensal_brasil$newDeaths>60000,
                                        'red', '#91b8bd'),
           colour=ifelse(covid_mensal_brasil$newDeaths>60000,
                         'red', '#91b8bd'))+
  geom_text(aes(label = round(newDeaths/1000,1)), size=2,
```

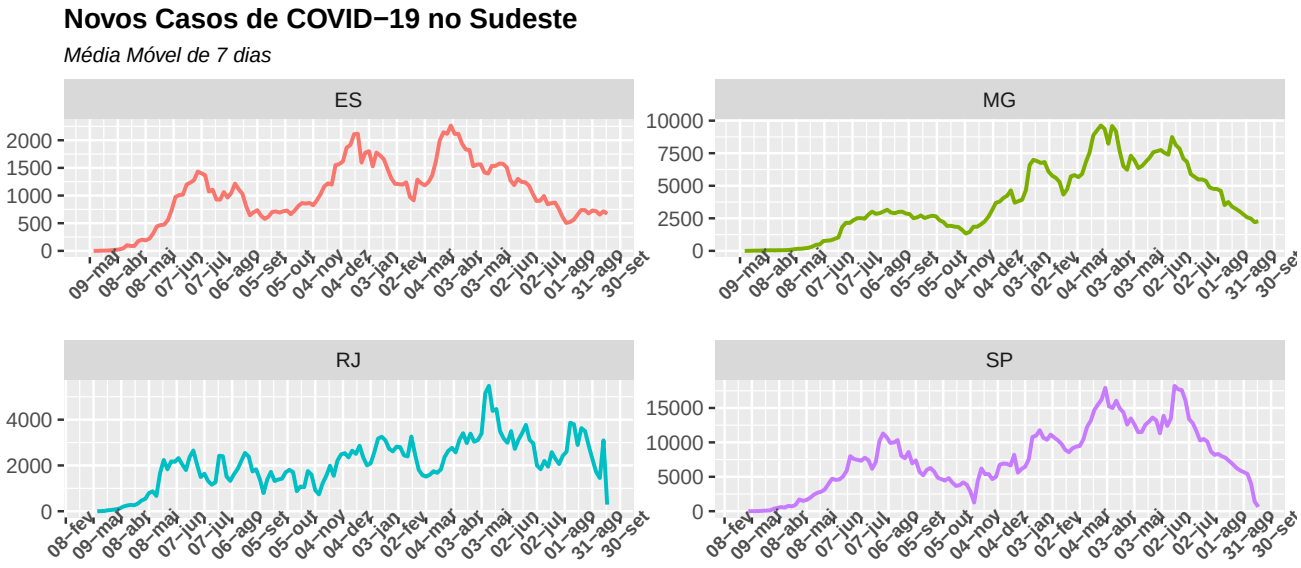
```
            vjust=-0.4, fontface='bold')+
  scale_x_date(breaks = '1 month',
               date_labels = '%b/%y',
               expand = c(0,13))+
  theme(axis.text.x = element_text(size=7, face='bold', angle=45),
        axis.text.y = element_text(size=8, face='bold'),
        plot.title = element_text(size=11, face='bold'),
        plot.caption = element_text(face='bold'),
        panel.background = element_rect(fill='white',
                                        colour='white'),
        axis.line.x.bottom = element_line(colour='black',
                                          linetype = 'solid'),
        axis.line.y.left = element_line(colour='black',
                                        linetype = 'solid'))+
  labs(x='', y='mil mortes',
       title='Mortes mensais por Covid-19 no Brasil',
       caption='Fonte: analisemacro.com.br')
```



4.4 Mortes e Novos Casos no Sudeste

```
covid %>%
  tsibble::as_tsibble(index = date, key = state) %>%
  dplyr::filter(state == c('RJ', 'SP', "MG", "ES")) %>%
```

```
  autoplot(MM_casos, size=.8)+
  facet_wrap(~state, scales = 'free')+
  scale_x_date(breaks='30 days',
               date_labels = '%d-%b')+
  labs(x='', y='',
       title="Novos Casos de COVID-19 no Sudeste",
       subtitle='Média Móvel de 7 dias',
       caption='Fonte: analisemacro.com.br')+
  theme(legend.position = 'none',
        plot.title = element_text(face='bold', size=12),
        plot.subtitle = element_text(face='italic', size=9),
        axis.text.x = element_text(size=8, angle=45, face='bold'))
```

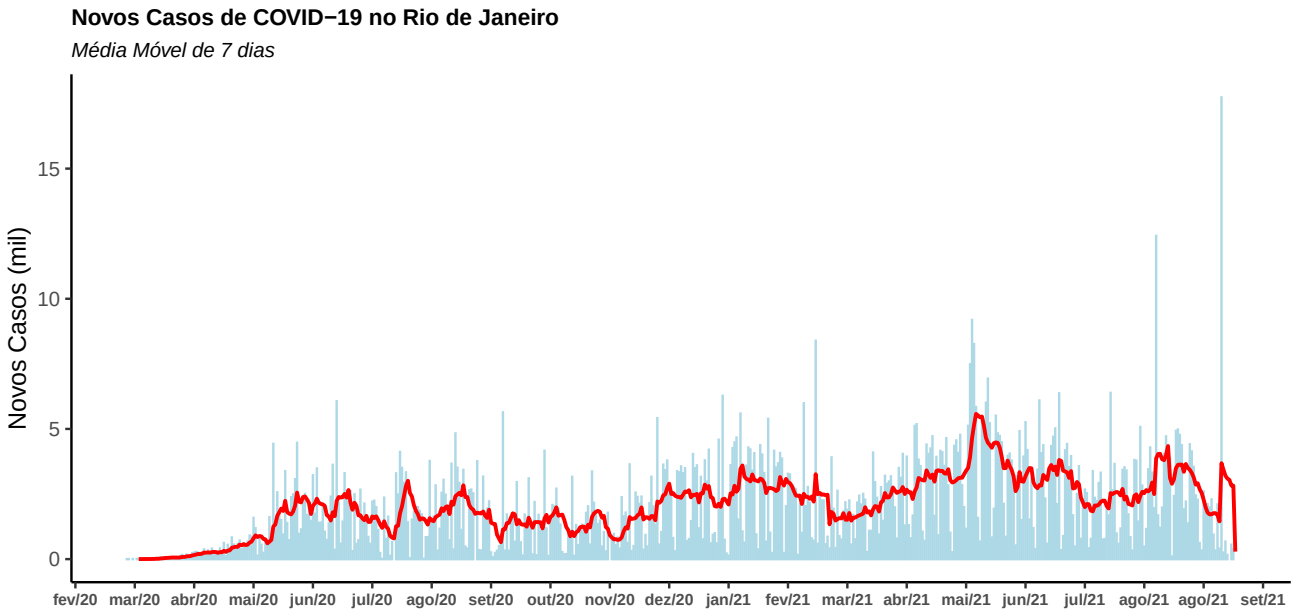


Fonte: analisemacro.com.br

4.5 Novos Casos no Rio de Janeiro

```
covid %>%
  dplyr::filter(state == "RJ") %>%
  ggplot(aes(x=date))+
  geom_bar(aes(y=newCases/1000), stat='identity', fill='lightblue',
           colour='lightblue', width = .00001)+
  geom_line(aes(y=MM_casos/1000), colour='red', size=.8)+
  scale_x_date(breaks='30 days',
               date_labels = '%b/%y')+
```

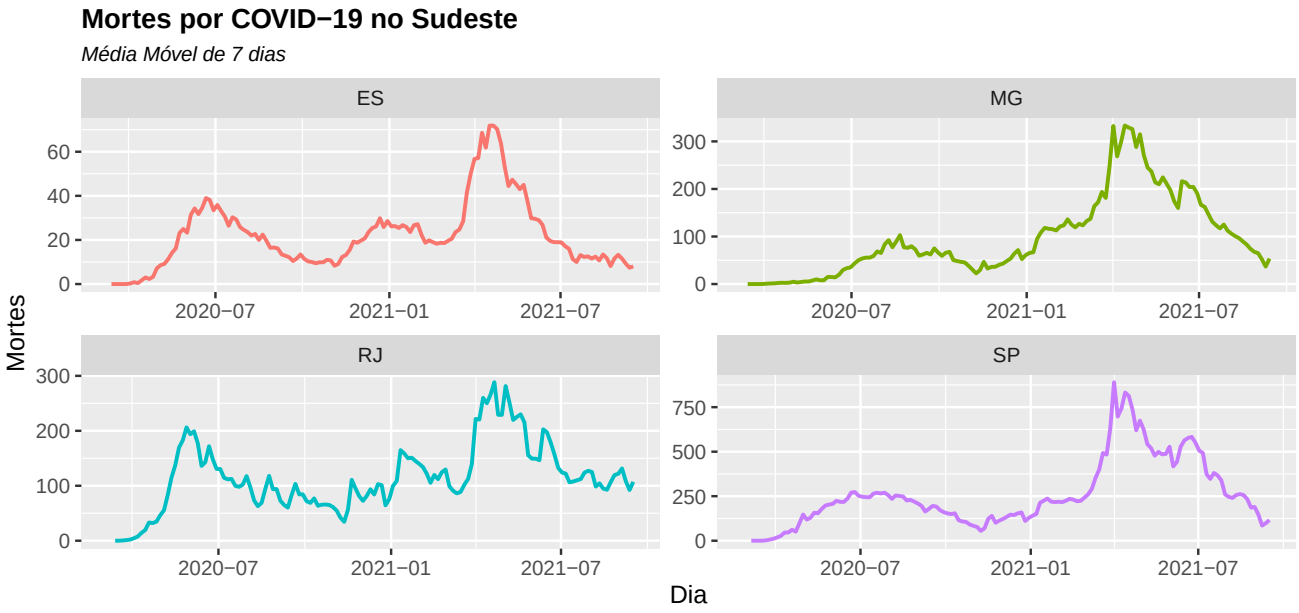
```
  labs(x="",y="Novos Casos (mil)",
       title="Novos Casos de COVID-19 no Rio de Janeiro",
       subtitle = 'Média Móvel de 7 dias')+
  theme(plot.title = element_text(size=10, face='bold'),
        axis.text.x = element_text(size=7, face='bold'),
        plot.subtitle = element_text(size=9, face='italic'),
        panel.background = element_rect(colour='white', fill='white'),
        axis.line.x.bottom = element_line(colour='black'),
        axis.line.y.left = element_line(colour='black'))
```



4.6 Mortes e Novos Casos no Sudeste

```
covid %>%
  tsibble::as_tsibble(index = date, key = state) %>%
  dplyr::filter(state == c('RJ', 'SP', "MG", "ES")) %>%
  autoplot(MM_mortes, size=.8)+
  facet_wrap(~state, scales = 'free')+
  labs(x="Dia",y="Mortes",
       title="Mortes por COVID-19 no Sudeste",
       subtitle='Média Móvel de 7 dias',
       caption='Fonte: analisemacro.com.br')+
  theme(legend.position = 'none',
        plot.title = element_text(face='bold', size=12),
```

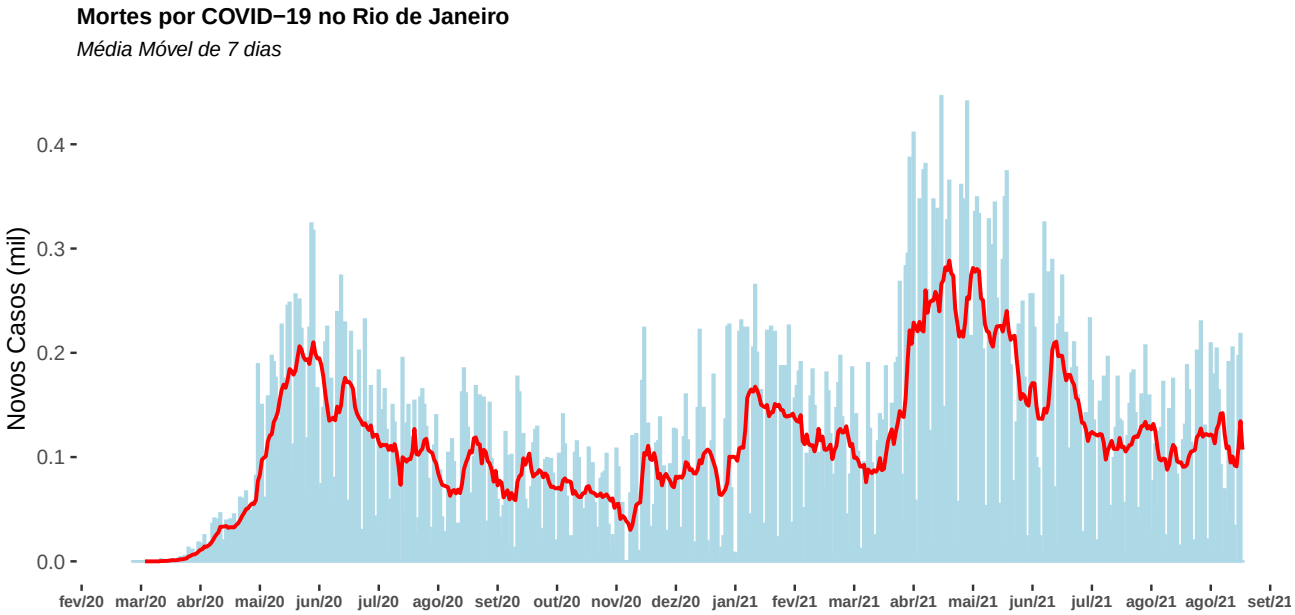
```
        plot.subtitle = element_text(face='italic', size=9))
```



Fonte: analisemacro.com.br

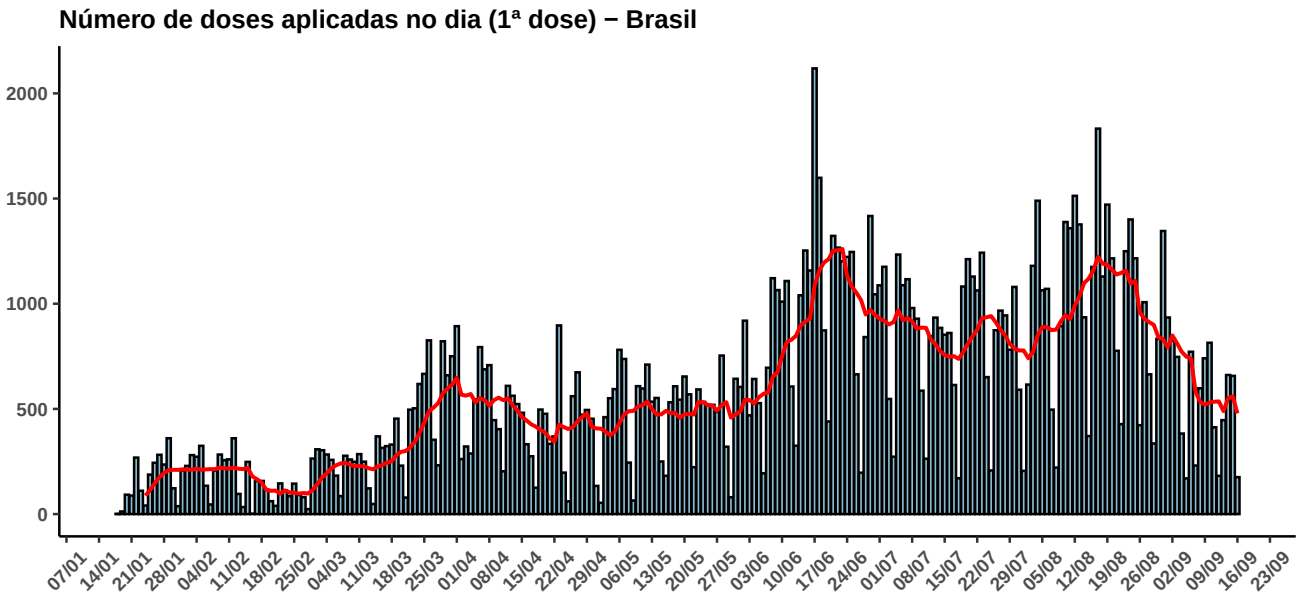
4.7 Mortes no Rio de Janeiro

```
covid %>%
  dplyr::filter(state == "RJ") %>%
  ggplot(aes(x=date))+
  geom_bar(aes(y=newDeaths/1000), stat='identity', fill='lightblue',
           colour='lightblue')+
  geom_line(aes(y=MM_mortes/1000), colour='red', size=.8)+
  scale_x_date(breaks='30 days',
               date_labels = '%b/%y')+
  labs(x="",y="Novos Casos (mil)",
       title="Mortes por COVID-19 no Rio de Janeiro",
       subtitle = 'Média Móvel de 7 dias')+
  theme(plot.title = element_text(size=10, face='bold'),
        axis.text.x = element_text(size=7, face='bold'),
        plot.subtitle = element_text(size=9, face='italic'),
        panel.background = element_rect(colour='white', fill='white'))
```



4.8 Número de doses aplicadas no dia

```
covid %>%
  dplyr::filter(date > '2021-01-07' & state %in% "TOTAL" &
           d_vaccinated > 0) %>%
  ggplot(aes(x=date, y=d_vaccinated))+
  geom_bar(aes(y=d_vaccinated/1000), stat='identity',
           fill='#8abbd0', colour='black')+
  geom_line(aes(y=MM_dose1/1000), colour='red', size=.8)+
  scale_x_date(breaks = '7 days',
               date_labels = "%d/%m")+
  theme(axis.text.x = element_text(size=8, face='bold', angle=45, vjust=.5),
        axis.text.y = element_text(size=8, face='bold'),
        plot.title = element_text(size=11, face='bold'),
        plot.caption = element_text(face='bold'),
        panel.background = element_rect(fill='white',
                                        colour='white'),
        axis.line.x.bottom = element_line(colour='black',
                                          linetype = 'solid'),
        axis.line.y.left = element_line(colour='black',
                                        linetype = 'solid'))+
  labs(x='', y='',
       title='Número de doses aplicadas no dia (1ª dose) - Brasil',
       caption='Fonte dos dados: https://github.com/wcota/covid19br')
```



Fonte dos dados: https://github.com/wcota/covid19br

5 Consumo em Restaurantes e Supermercados

5.1 Introdução

A Fipe (Fundação Instituto de Pesquisas Econômicas), em parceria com a Alelo, criou dois índices de consumo em restaurantes e supermercados que são bastante interessantes para verificar o impacto da pandemia nos hábitos de consumo dos brasileiros. Nesse Comentário de Conjuntura, fazemos uma análise dos índices. O código completo está disponível para os membros do Clube AM.

5.2 Coleta de Dados

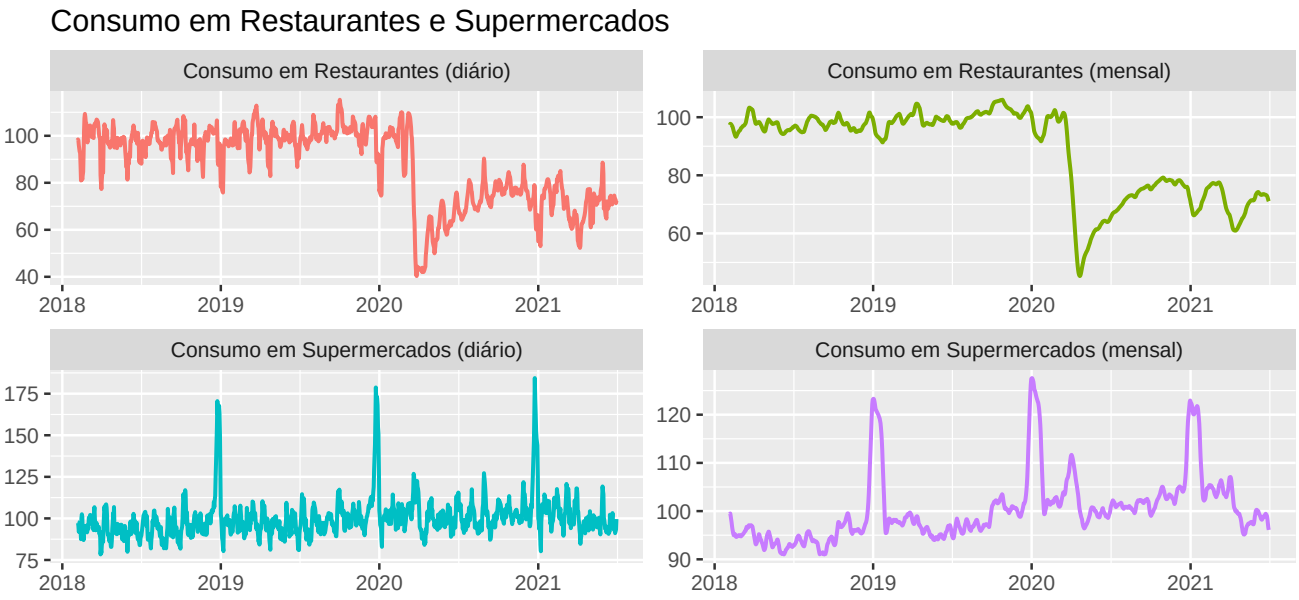
A seguir, nós baixamos a planilha excel disponível no site da FIPE.

```
url = 'https://downloads.fipe.org.br/indices/indicesconsumoalelo-serieshistoricas.xlsx'
download.file(url, destfile = 'seriehistorica.xlsx', mode='wb')
data = read_excel('seriehistorica.xlsx', sheet=1, skip=1) %>%
  dplyr::select(dia, "Índice de Consumo em Supermercados (ICS) - Valor",
         "Índice de Consumo em Restaurantes (ICR) - Valor") %>%
  rename('Consumo em Supermercados (diário)' = "Índice de Consumo em Supermercados (ICS) - Valor",
         'Consumo em Restaurantes (diário)' = "Índice de Consumo em Restaurantes (ICR) - Valor") %>%
  mutate(`Consumo em Supermercados (mensal)` = rollapply(`Consumo em Supermercados (diário)`, 30,
                                  mean, align='right',
                                  fill=NA)) %>%
  mutate(`Consumo em Restaurantes (mensal)` = rollapply(`Consumo em Restaurantes (diário)`, 30,
                                  mean, align='right',
                                  fill=NA)) %>%
  drop_na() %>%
  gather(variavel, valor, -dia)
```

5.3 Visualização de Dados

A seguir, nós podemos gerar o gráfico múltiplo abaixo.

```
ggplot(data, aes(x=dia, y=valor, colour=variavel))+
  geom_line(size=.8)+
  facet_wrap(~variavel, scales='free')+
  theme(legend.position = 'none')+
  labs(x='', y='',
       title='Consumo em Restaurantes e Supermercados',
       caption='Fonte: analisemacro.com.br com dados da FIPE')
```



Fonte: analisemacro.com.br com dados da FIPE

A análise dos gráficos sugere que o consumo em supermercados aumentou no início da pandemia, enquanto houve uma queda brusca no consumo em restaurantes. Enquanto aquele parece ter voltado ao seu nível normal, este ainda não completou a volta ao nível anterior à pandemia.

A esse respeito, é interessante verificar se haverá uma mudança permanente nos hábitos de consumo, isto é, mais pessoas cozinhando em casa, por exemplo, o que reduz o consumo potencial em restaurantes.

6 Coleta de preços de ações com o R

6.1 Coleta de dados de empresas estatais na B3

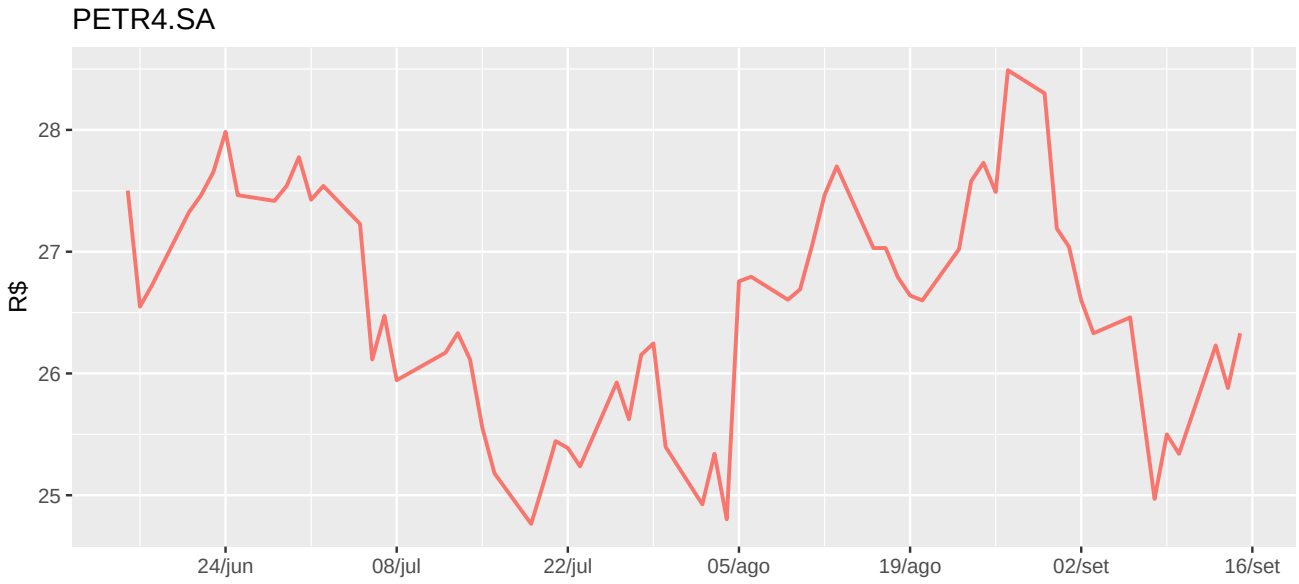
Abaixo, pegamos os dados de ações dessas três estatais a partir da base de dados do yahoo finance.

```
## Coleta de preços de ações
symbols = c('PETR4.SA', 'BBAS3.SA', 'ELET6.SA')
prices_date = Sys.Date() %m-% months(3)
prices = getSymbols(symbols, src='yahoo', from=prices_date) %>%
  map(~Ad(get(.))) %>%
  reduce(merge) %>%
  `colnames<-` (symbols) %>%
  tk_tbl(preserve_index = TRUE, rename_index = 'date') %>%
  drop_na() %>%
  gather(variavel, valor, -date)
```

6.2 Visualização de dados da Petrobras

Com os dados carregados, nós podemos gerar um gráfico da ação da Petrobras.

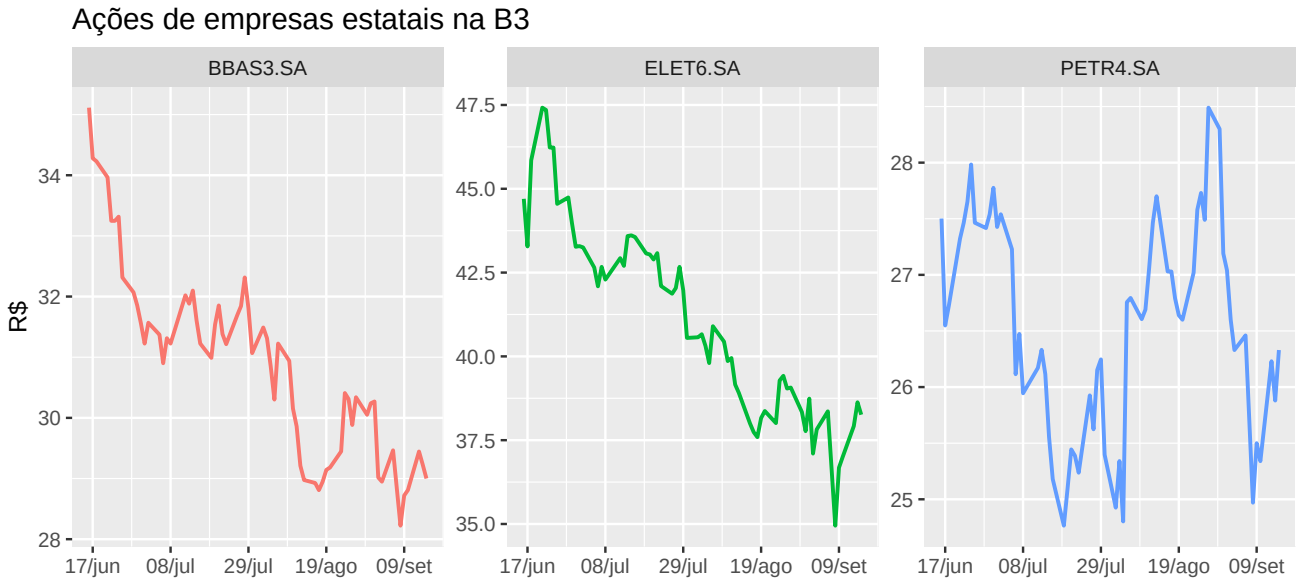
```
filter(prices, variavel == 'PETR4.SA') %>%
  ggplot(aes(x=date, y=valor, colour=variavel))+
  geom_line(size=.8)+
  theme(legend.position = 'none')+
  scale_x_date(breaks = date_breaks("14 days"),
               labels = date_format("%d/%b"))+
  labs(x='', y='R$',
       title='PETR4.SA',
       caption='Fonte: analisemacro.com.br com dados do Yahoo Finance')
```



Fonte: analisemacro.com.br com dados do Yahoo Finance

6.3 Visualização de dados de empresas estatais

```
prices %>%
  ggplot(aes(x=date, y=valor, colour=variavel))+
  geom_line(size=.8)+
  theme(legend.position = 'none')+
  facet_wrap(~variavel, scales='free')+
  scale_x_date(breaks = date_breaks("21 days"),
               labels = date_format("%d/%b"))+
  labs(x='', y='R$',
       title='Ações de empresas estatais na B3',
       caption='Fonte: analisemacro.com.br com dados do Yahoo Finance')
```



Fonte: analisemacro.com.br com dados do Yahoo Finance

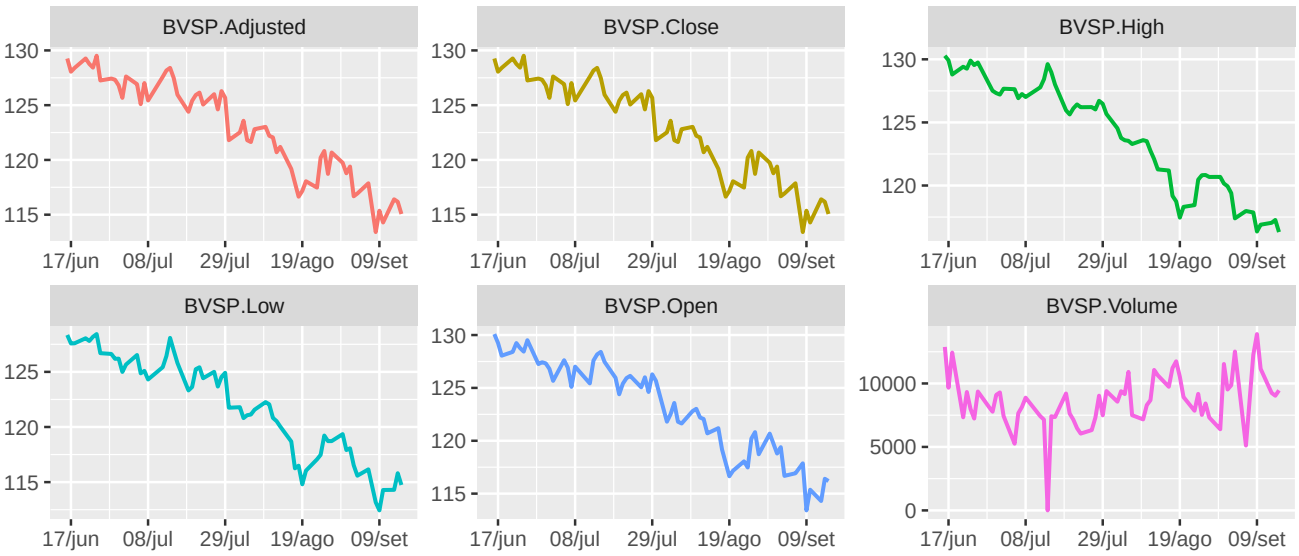
6.4 O índice Bovespa

```
getSymbols('^BVSP', from=prices_date)
```

[1] "^BVSP"

```
ibov = tk_tbl(BVSP, preserve_index = TRUE, rename_index = 'date') %>%
  drop_na() %>%
  gather(variavel, valor, -date)


ibov %>%
  ggplot(aes(x=date, y=valor/1000, colour=variavel))+
  geom_line(size=.8)+
  theme(legend.position = 'none')+
  facet_wrap(~variavel, scales='free')+
  scale_x_date(breaks = date_breaks("21 days"),
               labels = date_format("%d/%b"))+
  labs(x='', y='',
       title='IBOVESPA',
       caption='Fonte: analisemacro.com.br com dados do Yahoo Finance')
```

IBOVESPA



Fonte: analisemacro.com.br com dados do Yahoo Finance

7 Gráfico com eixo y secundário no R

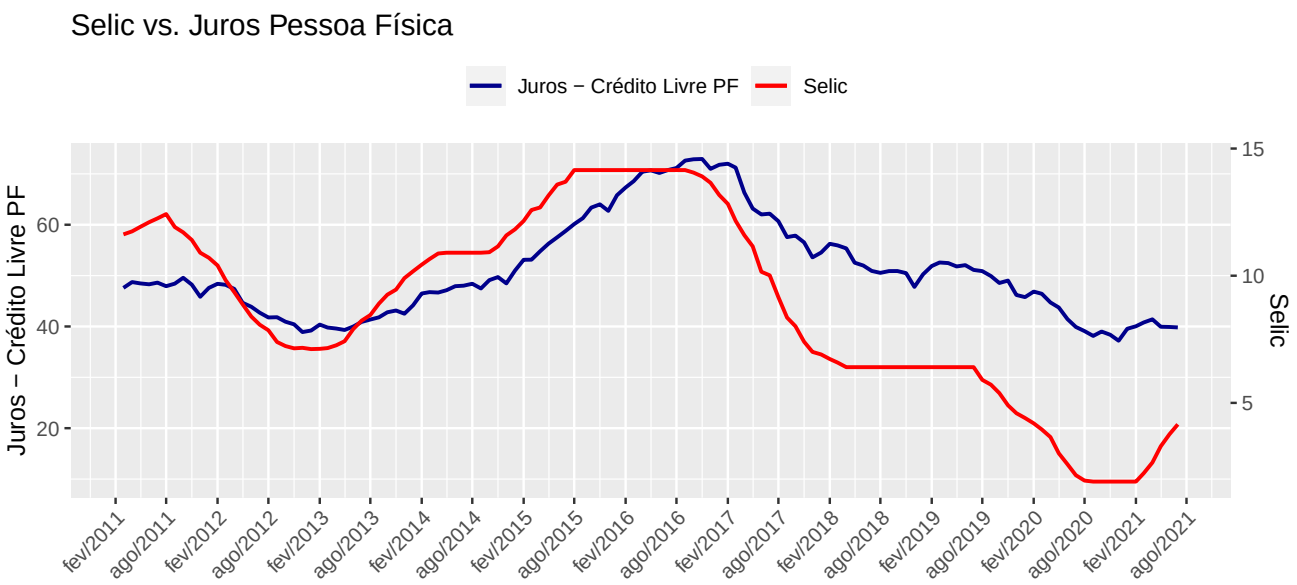
7.1 Coleta de dados

```
series = list('Selic' = 4189,
              'Juros PF' = 20740)


data = get_series(series) %>%
  purrr::reduce(inner_join)
```

7.2 Visualização de dados

O gráfico abaixo ilustra a diferença entre a taxa básica de juros e os juros associados ao crédito livre, pessoa física.

```
ggplot(data, aes(x=date))+
  geom_line(aes(y=`Juros PF`, colour='Juros - Crédito Livre PF'), size=.8)+
  geom_line(aes(y=Selic*5, colour='Selic'), size=.8)+
  scale_y_continuous(sec.axis = sec_axis(~./5, name='Selic'))+
  scale_colour_manual('',
                      values=c('darkblue', 'red'))+
  theme(legend.position = 'top')+
  xlab('')+ylab('Juros - Crédito Livre PF')+
  scale_x_date(breaks = date_breaks("6 month"),
               labels = date_format("%b/%Y"))+
  theme(axis.text.x=element_text(angle=45, hjust=1))+
  labs(title='Selic vs. Juros Pessoa Física',
       caption='Fonte: analisemacro.com.br com dados do BCB.')
```



Fonte: analisemacro.com.br com dados do BCB.

8 Juro real ex-ante vs. juro neutro

```
##### Função para calcular o Juro Neutro e Juro Real
fisher <- function(juros, inflacao) {
  (((1 + (juros / 100)) / (1 + (inflacao / 100))) - 1) * 100
}
```

8.1 Coleta de Dados

```
###################### COLETA DE DADOS ################################

##### Expectativas de mercado anuais dos indicadores Focus/BCB
raw_focus <- meedr::get_annual(
  indicator   = c('IPCA', 'Selic'),
  first_date  = Sys.Date() - 10 * 365,
  use_memoise = FALSE
)

##### Taxa referencial - swaps - DI pré-360 dias - média do período (IPEADATA/B3)
raw_swaps <- httr::GET(
  sprintf("http://ipeadata.gov.br/api/odata4/ValoresSerie(SERCODIGO='%s')",
          "BMF12_SWAPDI36012")
)

##### Expectativa média do IPCA - tx. acumulada para os próximos 12 meses (IPEADATA/B3)
raw_ipca_expect_12m <- httr::GET(
  sprintf("http://ipeadata.gov.br/api/odata4/ValoresSerie(SERCODIGO='%s')",
          "BM12_IPCAEXP1212")
)
```

8.2 Tratamento de Dados

```
################## TRATAMENTO DE DADOS ############################

# Juntar expectativas do IPCA e Selic e calcular proxy do Juro Neutro (RTI 12/2019 BCB)
proxy_neutro <- dplyr::left_join(
  # Expectativas de mercado anuais para o IPCA (Focus/BCB)
  raw_focus %>%
```

```
    dplyr::filter(
      indicator == "IPCA",
      reference_date == lubridate::year(date) + 3,
      basis == 0
    ) %>%
    dplyr::select(
      date,
      "ipca_e"     = median,
      "ipca_e_min" = min,
      "ipca_e_max" = max
    ),
  # Expectativas de mercado anuais para a Selic (Focus/BCB)
  raw_focus %>%
    dplyr::filter(
      indicator == "Selic",
      # detail == "Fim do ano", DEPRECATED
      reference_date == lubridate::year(date) + 3
    ) %>%
    dplyr::select(
      date,
      "selic_e"     = median,
      "selic_e_min" = min,
      "selic_e_max" = max
    ),
  by = "date"
) %>%
  # Calcula proxy do Juro Neutro
  dplyr::mutate(
    neutro     = fisher(selic_e, ipca_e),
    neutro_min = fisher(selic_e_min, ipca_e_min),
    neutro_max = fisher(selic_e_max, ipca_e_max),
  ) %>%
  # Mensaliza os dados (média)
  dplyr::group_by(date = lubridate::floor_date(date, unit = "month")) %>%
  dplyr::summarise(
    dplyr::across(dplyr::everything(), ~mean(.x, na.rm = TRUE))
  )
```

```r
# Taxa referencial - swaps - DI pré-360 dias - média do período (IPEADATA/B3)
swaps <- httr::content(raw_swaps)[[2]] %>%
  dplyr::bind_rows() %>%
  dplyr::select("date" = `VALDATA`, "swap" = `VALVALOR`) %>%
  dplyr::mutate(date = lubridate::as_date(date))

# Expectativa média do IPCA - tx. acumulada para os próximos 12 meses (IPEADATA/B3)
ipca_expect_12m <- httr::content(raw_ipca_expect_12m)[[2]] %>%
  dplyr::bind_rows() %>%
  dplyr::select("date" = `VALDATA`, "ipca_e_12m" = `VALVALOR`) %>%
  dplyr::mutate(date = lubridate::as_date(date))

# Juntar Taxa Swap e Expectativa do IPCA de 12 meses e calcular juro real ex-ante
ex_ante <- dplyr::left_join(
  ipca_expect_12m,
  swaps,
  by = "date"
) %>%
  dplyr::mutate(juro_real = fisher(swap, ipca_e_12m)) %>%
  tidyr::drop_na()


# Juntar dados da proxy do Juro Neutro e do juro real ex-ante
tbl_juros <- dplyr::left_join(
  proxy_neutro,
  ex_ante,
  by = "date"
) %>%
  tidyr::drop_na()
```

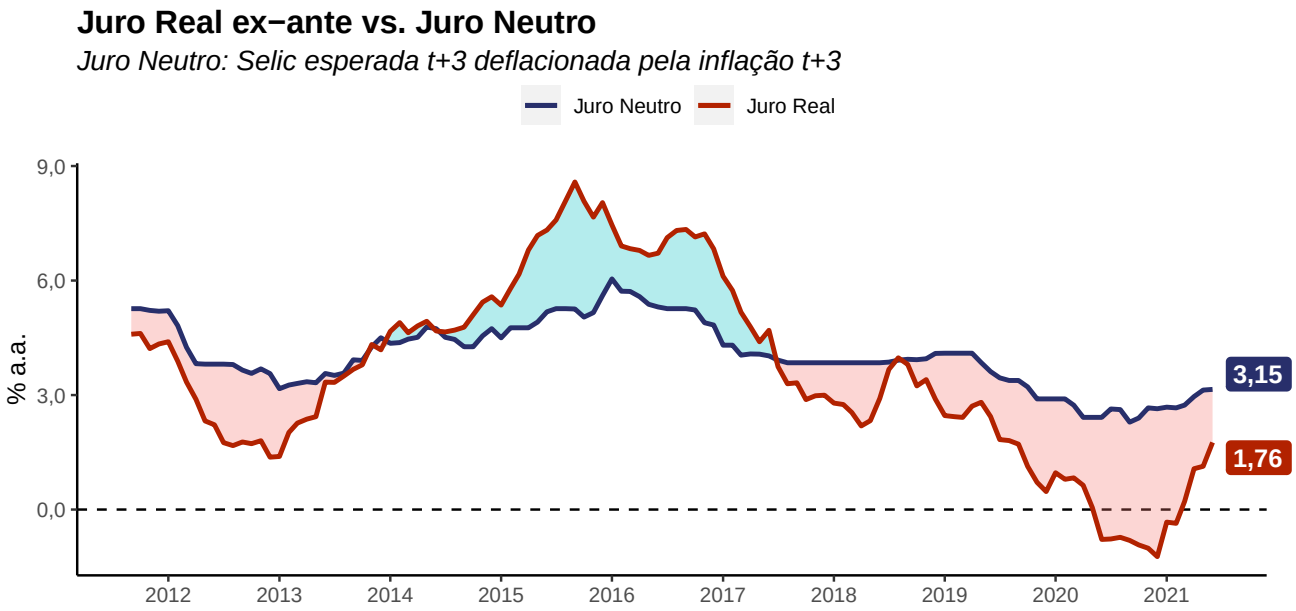
8.3 Visualização de Dados

```r
############################## VISUALIZAÇÃO DE DADOS ############################

# Cores para gráficos e tabelas
colors <- c(
  blue       = "#282f6b",
```

```
  red        = "#b22200",
  yellow     = "#eace3f",
  green      = "#224f20",
  purple     = "#5f487c",
  orange     = "#b35c1e",
  turquoise  = "#419391",
  green_two  = "#839c56",
  light_blue = "#3b89bc",
  gray       = "#666666"
)

# Fonte para gráficos e tabelas
foot_bcb <- "Fonte: analisemacro.com.br com dados do BCB."
foot_b3_bcb <- "Fonte: analisemacro.com.br com dados de B3 e BCB."
foot_anbima <- "Fonte: analisemacro.com.br com dados da Anbima."

# Definir padrão de gráficos
ggplot2::theme_set(
  ggplot2::theme(
    plot.title = ggplot2::element_text(size = 14, face = "bold", hjust = 0, vjust = 2),
    plot.subtitle = ggplot2::element_text(size = 12, face = "italic", hjust = 0),
    plot.caption = ggplot2::element_text(size = 10, hjust = 1),
  )
)

# código para o gráfico
tbl_juros %>%
  ggplot2::ggplot(ggplot2::aes(x = date)) +
  ggplot2::geom_hline(yintercept = 0, linetype = "dashed", colour = "black") +
  ggplot2::geom_ribbon(
    ggplot2::aes(ymin = pmin(neutro, juro_real), ymax = juro_real, fill = unname(colors["red"])),
    alpha = 0.3
  ) +
  ggplot2::geom_ribbon(
    ggplot2::aes(ymin = pmin(neutro, juro_real), ymax = neutro, fill = unname(colors["blue"])),
    alpha = 0.3
  ) +
```

```
ggplot2::geom_line(ggplot2::aes(y = neutro, colour = "Juro Neutro"), size = 1) +
ggplot2::geom_line(ggplot2::aes(y = juro_real, colour = "Juro Real"), size = 1) +
ggplot2::guides(fill = "none") +
ggplot2::scale_colour_manual(
  NULL,
  values = c("Juro Neutro" = unname(colors["blue"]), "Juro Real" = unname(colors["red"]))
) +
ggplot2::labs(
  title    = "Juro Real ex-ante vs. Juro Neutro",
  subtitle = "Juro Neutro: Selic esperada t+3 deflacionada pela inflação t+3",
  caption  = foot_b3_bcb,
  y        = "% a.a.",
  x        = ""
) +
ggplot2::scale_x_date(
  breaks = scales::breaks_width("1 year"),
  labels = scales::date_format("%Y")
) +
ggplot2::scale_y_continuous(
  labels = scales::comma_format(big.mark = " ", decimal.mark = ",", accuracy = 0.1)
) +
ggplot2::coord_cartesian(clip = "off") +
ggplot2::geom_label(
  ggplot2::aes(y = juro_real, label = format(round(juro_real, 2), decimal.mark = ",")),
  nudge_x     = 0.1,
  nudge_y     = -0.4,
  hjust       = -0.2,
  label.size  = 0,
  fontface    = "bold",
  colour      = "white",
  show.legend = FALSE,
  fill        = unname(colors["red"]),
  data        = dplyr::filter(tbl_juros, date == dplyr::last(date))
) +
ggplot2::geom_label(
  ggplot2::aes(y = neutro, label = format(round(neutro, 2), decimal.mark = ",")),
  nudge_x     = 0.1,
```

```
    nudge_y     = 0.4,
    hjust       = -0.2,
    label.size  = 0,
    fontface    = "bold",
    colour      = "white",
    show.legend = FALSE,
    fill        = unname(colors["blue"]),
    data        = dplyr::filter(tbl_juros, date == dplyr::last(date))
  ) +
  ggplot2::theme(
    panel.background   = ggplot2::element_rect(fill = "white", colour = "white"),
    axis.line.x.bottom = ggplot2::element_line(colour = "black"),
    axis.line.y.left   = ggplot2::element_line(colour = "black"),
    legend.position    = "top",
    plot.margin        = ggplot2::margin(5, 25, 5, 5)
  )
```



Fonte: analisemacro.com.br com dados de B3 e BCB.

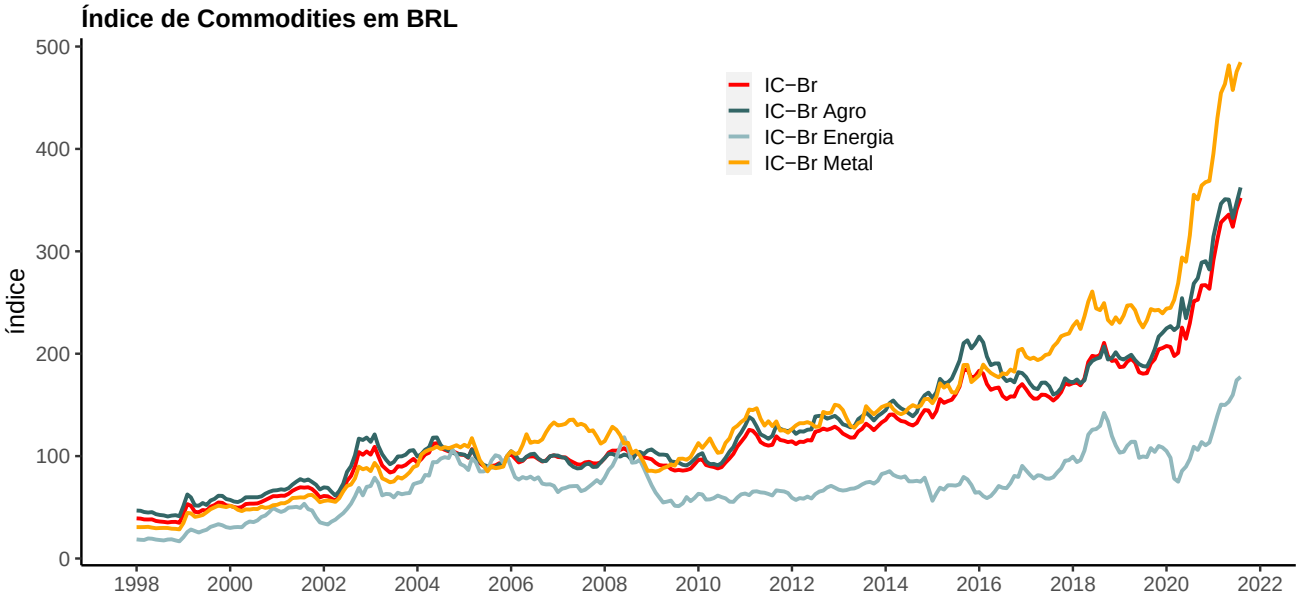
9 Commodities vs. Preços no Atacado

9.1 Coleta de Dados

```
series = list("IC-Br" = 27574, "IC-Br Agro" = 27575, "IC-Br Metal" = 27576,
              "IC-Br Energia" = 27577, "IPA" = 225, "IPA Indústria" = 7459,
              "IPA Agro" = 7460)


data = get_series(series) %>%
  reduce(inner_join) %>%
  mutate(across(c('IPA', 'IPA Indústria', 'IPA Agro'),
                (function(x) 1+x/100))) %>%
  mutate(across(c('IPA', 'IPA Indústria', 'IPA Agro'),
                (function(x) (roll_prod(x, n=12, align='right',
                                        fill = NA)-1)*100 )))
```

9.2 Índice de Commodities em BRL

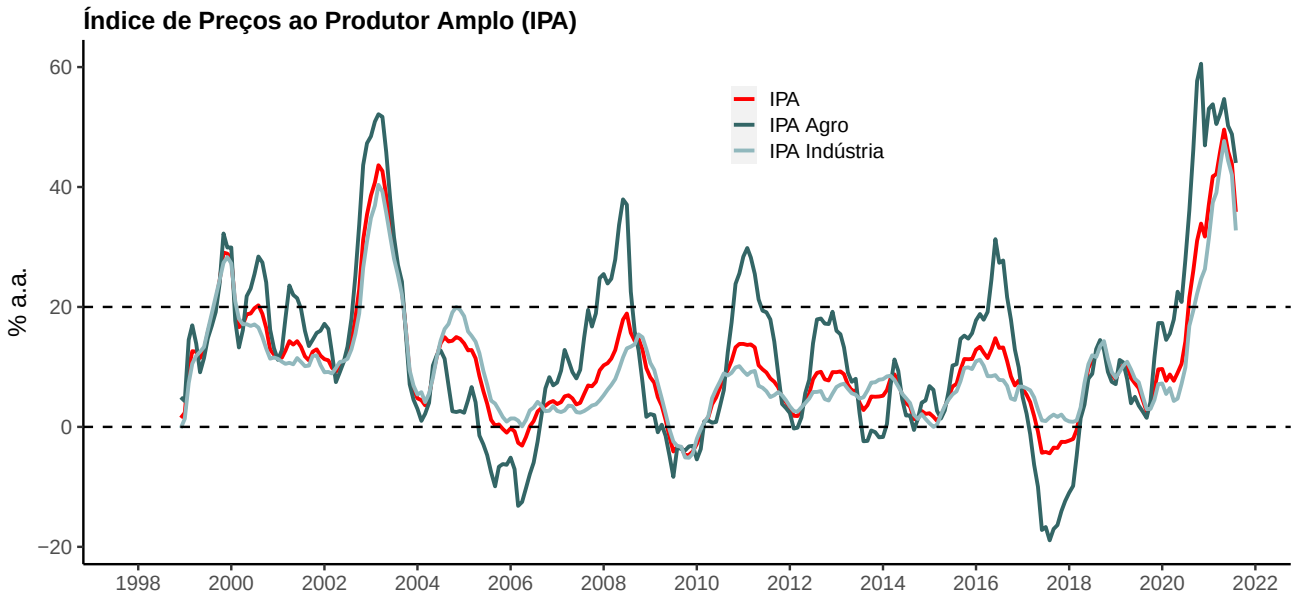
```
data %>%
  dplyr::select(-IPA, -`IPA Indústria`, -`IPA Agro`) %>%
  tidyr::gather(variavel, valor, -date) %>%
  ggplot(aes(x=date, y=valor, colour=variavel))+
  geom_line(size=.8)+
  theme(legend.title = element_blank(),
        legend.position = c(.6,.85),
        legend.key.size = unit(.4, 'cm'),
        panel.background = element_rect(colour='white', fill='white'),
        axis.line.x.bottom = element_line(colour='black'),
        axis.line.y.left = element_line(colour='black'),
        plot.title = element_text(size=11, face='bold'))+
  scale_colour_manual(values=c('red', '#336666', '#91b8bd', "orange"))+
  scale_x_date(date_breaks = '2 year',
               date_labels = '%Y')+
  labs(x='', y='índice',
       title='Índice de Commodities em BRL',
       caption='Fonte: analisemacro.com.br')
```



Fonte: analisemacro.com.br

9.3 Índice de Preços ao Produtor Amplo (IPA)

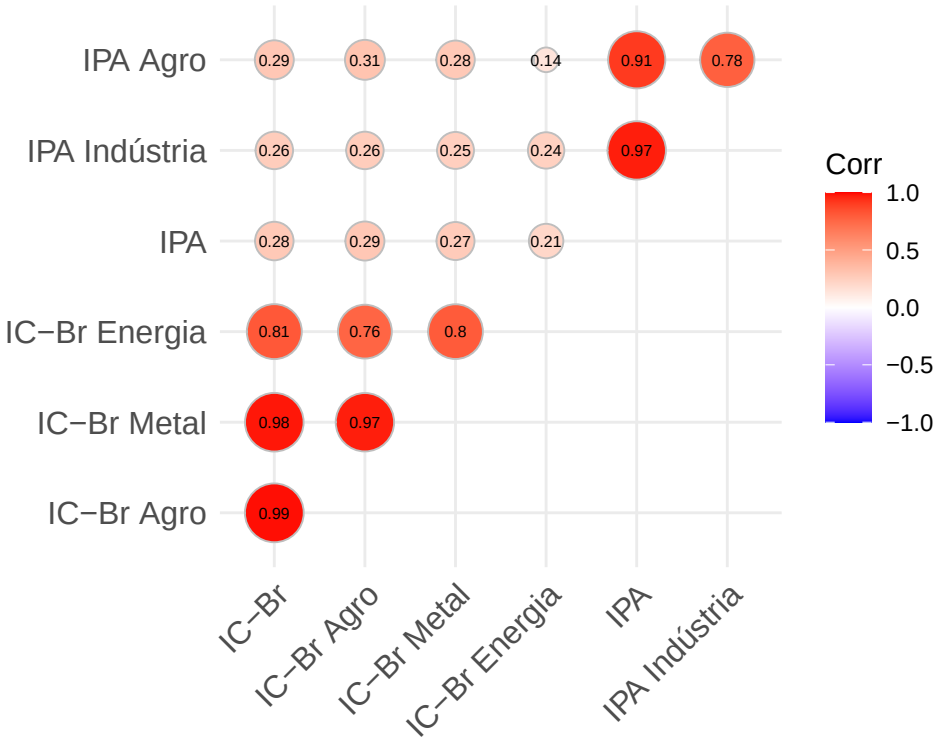
```
data %>%
  dplyr::select(date, IPA, `IPA Indústria`, `IPA Agro`) %>%
  tidyr::gather(variavel, valor, -date) %>%
  ggplot(aes(x=date, y=valor, colour=variavel))+
  geom_line(size=.8)+
  geom_hline(yintercept=0, linetype='dashed', colour='black')+
  geom_hline(yintercept=20, linetype='dashed', colour='black')+
  theme(legend.title = element_blank(),
        legend.position = c(.6,.85),
        legend.key.size = unit(.4, 'cm'),
        panel.background = element_rect(colour='white', fill='white'),
        axis.line.x.bottom = element_line(colour='black'),
        axis.line.y.left = element_line(colour='black'),
        plot.title = element_text(size=11, face='bold'))+
  scale_colour_manual(values=c('red', '#336666', '#91b8bd'))+
  scale_x_date(date_breaks = '2 year',
               date_labels = '%Y')+
  labs(x='', y='% a.a.',
       title='Índice de Preços ao Produtor Amplo (IPA)',
       caption='Fonte: analisemacro.com.br')
```



Fonte: analisemacro.com.br

9.4 Matriz de Correlação

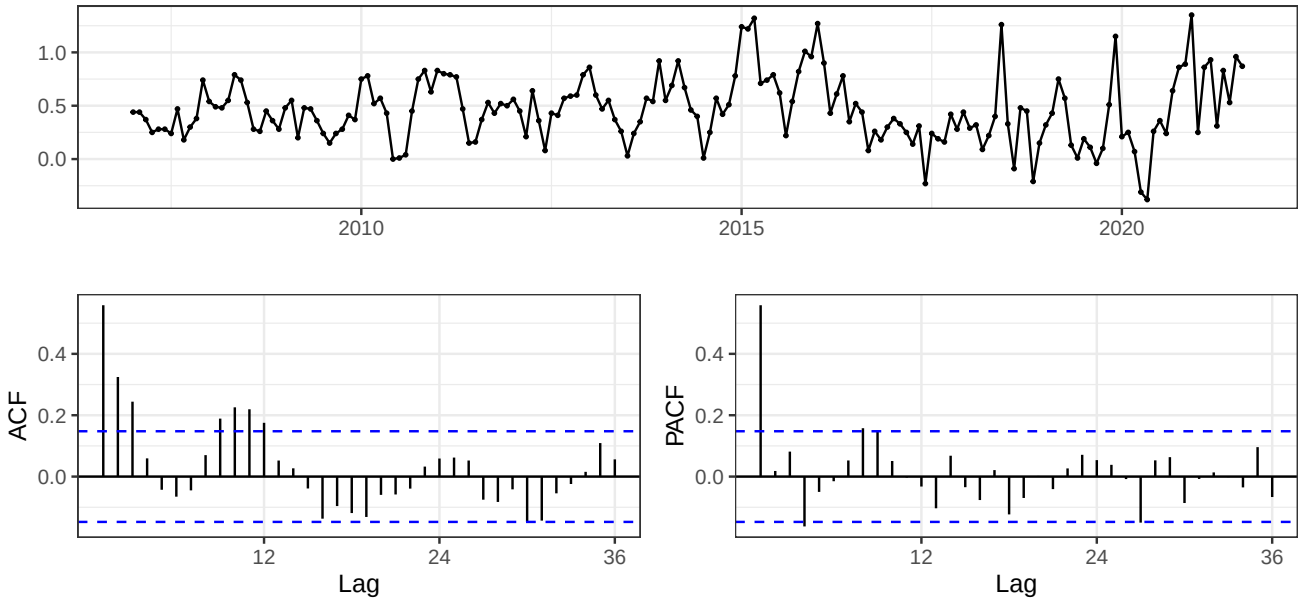
```
data %>%
  dplyr::select(-date) %>%
  drop_na() %>%
  cor() %>%
  ggcorrplot(method='circle',
             type='upper',
             show.diag = F,
             lab=TRUE,
             lab_col = 'black',
             lab_size = 2)
```


IPA Agro
IPA Indústria
IPA
IC−Br Energia
IC−Br Metal
IC−Br Agro
0.29
0.31
0.28
0.14
0.91
0.78
0.26
0.26
0.25
0.24
0.97
0.28
0.29
0.27
0.21
0.81
0.76
0.8
0.98
0.97
0.99
IC−Br
IC−Br Agro
IC−Br Metal
IC−Br Energia
IPA
IPA Indústria
Corr
1.0
0.5
0.0
−0.5
−1.0

10 Pandemia causou aumento da inércia inflacionária no Brasil

10.1 Inflação mensal

```
## Inflação
inflacao = BETSget(433, from='2007-01-01')
ggtsdisplay(inflacao, theme=theme_bw())
```



10.2 Estimar AR1

```
## Estimar AR1
ar1 = Arima(inflacao, order=c(1,0,0))
summary(ar1)
```

```
## Series: inflacao
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1    mean
##       0.5610  0.4643
## s.e.  0.0623  0.0435
##
## sigma^2 estimated as 0.06585:  log likelihood=-9.53
## AIC=25.05   AICc=25.19   BIC=34.57
##
```

```
## Training set error measures:
##                        ME      RMSE       MAE  MPE MAPE      MASE        ACF1
## Training set 0.0001013694 0.2551557 0.1942576 -Inf  Inf 0.6697131 -0.01022134
```

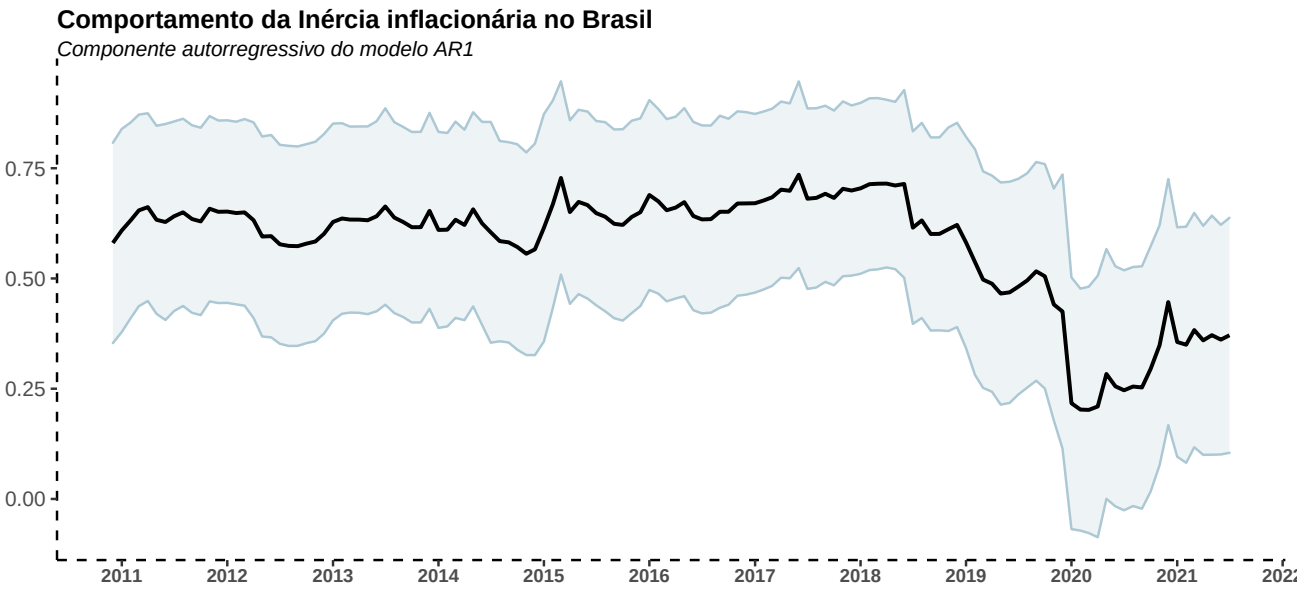
10.3 Rolling Regression

```
### Criando matrizes que guardarão coeficientes e desvios-padrões
p <- 2 # Parâmetros a serem guardados
janela <- 48 # número de meses da janela
coefs <- matrix(NA, ncol = p, nrow = length(inflacao)-janela)
dps <- matrix(NA, ncol = p, nrow = length(inflacao)-janela)
colnames(coefs) <- c('AR(1)', 'Intercepto')
colnames(dps) <- c('AR(1)', 'Intercepto')

### Loop para rodar o AR(1) com janela variável
for (i in 1:nrow(coefs)){
  ar1 <- Arima(inflacao[(1+i-1):(janela+i-1)],
               order=c(1,0,0))
  coefs[i,] <- coef(ar1)
  dps[i,] <- coeftest(ar1)[,2]
}

### Transformando as séries obtidas em séries de tempo
ar1 <- ts(coefs[,1],
          start=as.Date(time(inflacao)[janela]),
          freq=12)
dp <- ts(dps[,1],
         start=as.Date(time(inflacao)[janela]),
         freq=12)

### Intervalos de confiança
dpsup <- ar1+1.96*dp
dpinf <- ar1-1.96*dp
```

10.4 Visualização dos Dados

```
data = tk_tbl(ar1, preserve_index = TRUE, rename_index = 'date') %>%
  mutate(dpsup = dpsup,
```

```
         dpinf = dpinf,
         date = as.Date(date)) %>%
  rename(ar1 = value)


data %>%
  ggplot(aes(x=date))+
  geom_ribbon(aes(ymin=dpinf, ymax=dpsup), colour='#acc8d4', fill='#acc8d4',
              alpha=.2)+
  geom_line(aes(y=ar1), size=.8, collour='#244747')+
  scale_x_date(breaks = '1 year',
               labels = date_format('%Y'))+
  xlab('')+ylab('')+
  labs(title='Comportamento da Inércia inflacionária no Brasil',
       subtitle='Componente autorregressivo do modelo AR1',
       caption='Fonte: analisemacro.com.br')+
  theme(panel.background = element_rect(fill='white',
                                        colour='white'),
        axis.line = element_line(colour='black',
                                 linetype = 'dashed'),
        axis.line.x.bottom = element_line(colour='black'),
        axis.text.x = element_text(size=8, face='bold'),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        legend.position = 'right',
        legend.key.size = unit(0.3, "cm"),
        plot.margin=margin(5,5,15,5),
        plot.title=element_text(size=11, face='bold'),
        plot.subtitle=element_text(size=9, face='italic'))
```



Fonte: analisemacro.com.br

11 Coletando dados de comércio internacional com o R

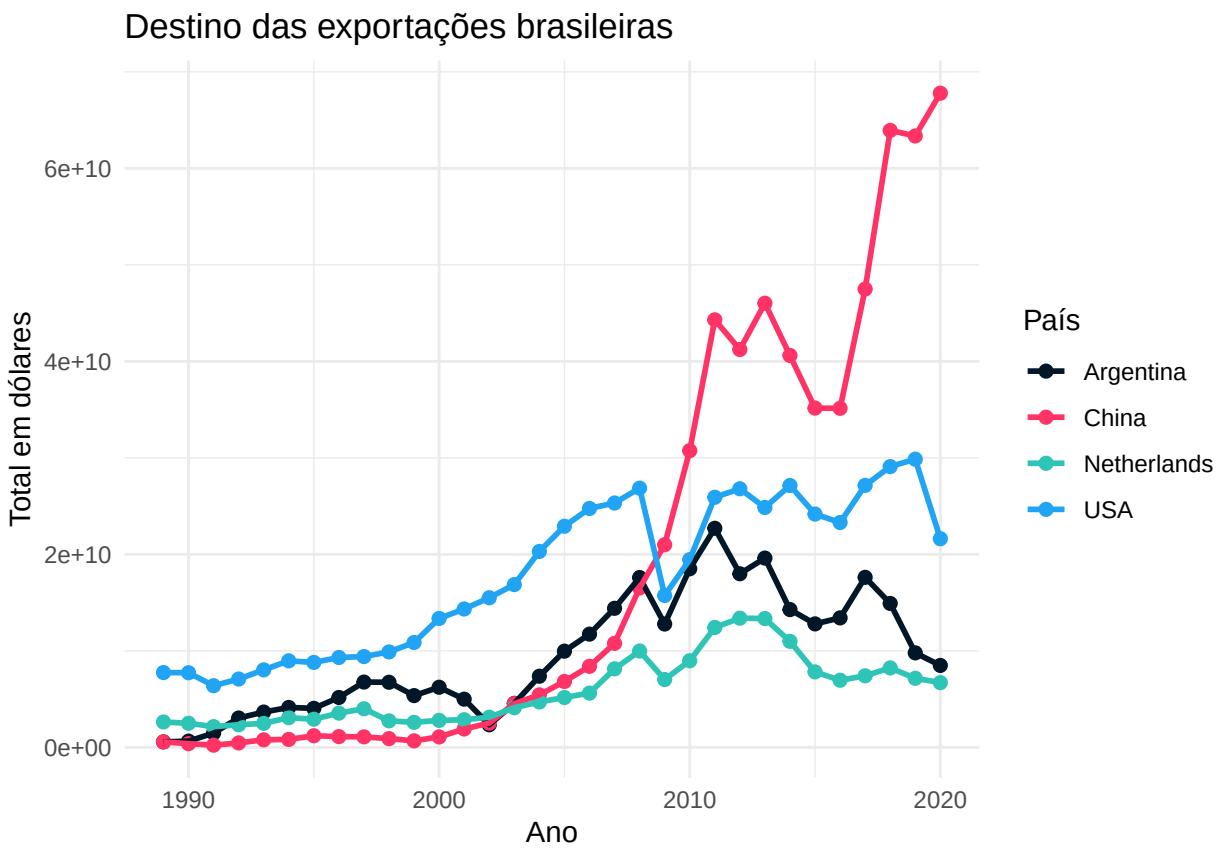
A base de dados `UN Comtrade` fornece acesso gratuito às informações de comércio global. É possível obter dados usando sua interface de extração (https://comtrade.un.org/) ou API. Por sorte, existe um pacote, chamado comtradr, que facilita o uso da API no R. Neste post, iremos mostrar um pouco da funcionalidade dele.

Primeiramente, iremos selecionar os dados de exportações brasileiras com quatro países: China, Estados Unidos, Argentina e Holanda. Como não selecionamos o período, todos os dados desde 1990 são retornados. Além disso, é possível filtrar por tipo de produto. Neste caso, estamos selecionando o total exportado.

```
braziltop <- ct_search(reporters = "Brazil",
                partners = c("USA", "China", "Argentina", "Netherlands"),
                trade_direction = "exports") %>%
  ct_use_pretty_cols()
```

Veja que o formato de saída torna muito simples fazer um gráfico com a evolução das exportações ao longo dos anos. Fica evidente a rápida expensão chinesa para se tornar principal destino dos produtos brasileiros.

```
ggplot(braziltop, aes(Year, `Trade Value usd`, color = factor(`Partner Country`))) +
  geom_point(size = 2) +
  geom_line(size = 1) +
  scale_color_manual(values = c("#011627", "#FF3366", "#2EC4B6", "#20A4F3"),
                     name = "País") +
  scale_shape_discrete(name = "País") +
  labs(title = "Destino das exportações brasileiras",
       y = "Total em dólares",
       x = "Ano") +
  theme_minimal()
```



Agora, ao invés de selecionarmos todos os produtos, iremos escolher apenas as exportações de peixe. Para fazer a filtragem, é preciso utilizar os códigos do sistema harmonizado (SH), que cataloga os produtos em categorias gerais e específicas. Essa filtragem é feita no parâmetro "commod_codes" da função. Iremos extrair apenas os dados referentes a 2020.

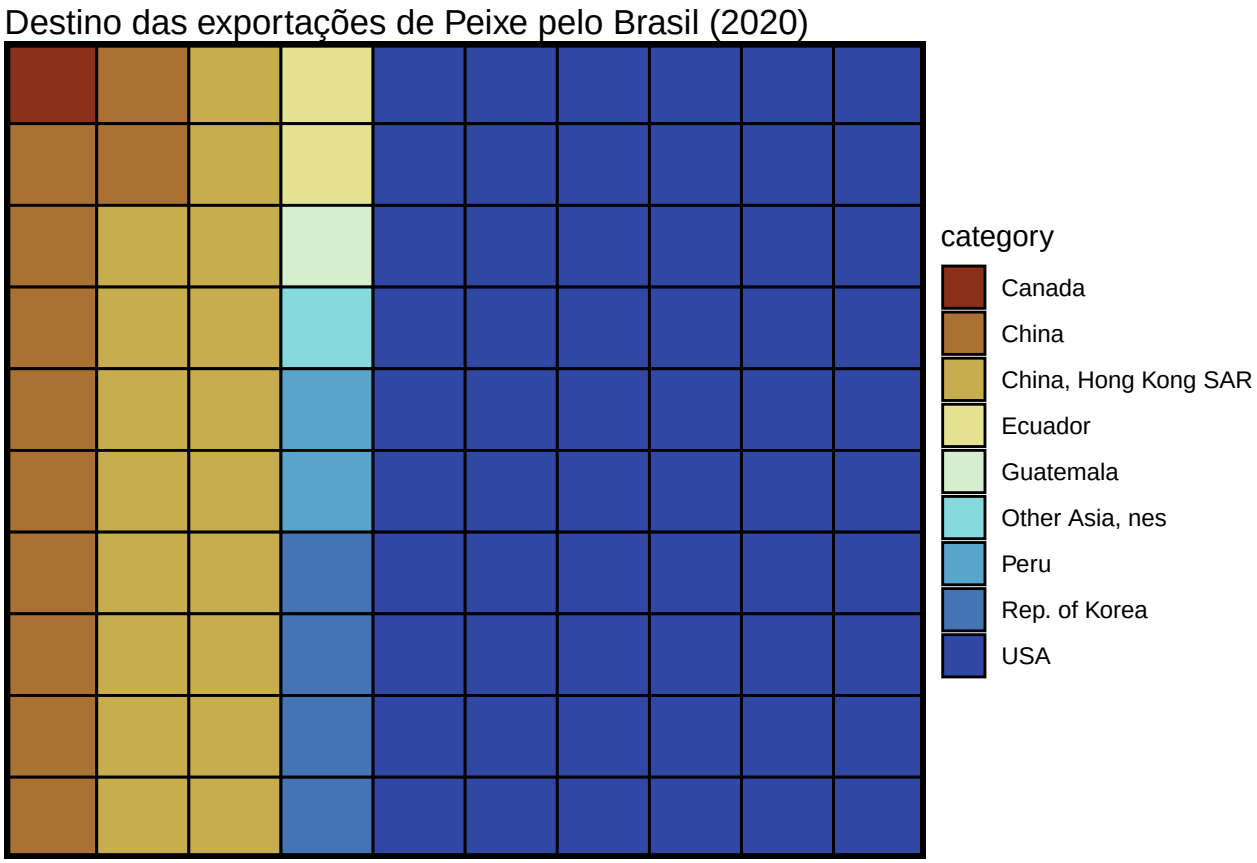
Como nós selecionamos os produtos, a API retorna o dado individual de cada categoria. Assim, iremos somar o valor exportado por país de destino.

```
export_peixe <- ct_search(reporters = "Brazil",
                partners = "All",
                trade_direction = "exports",
                start_date = 2020,
                end_date = 2020,
                commod_codes = c("0301","0302","0303","0304","0305")
                ) %>%
  ct_use_pretty_cols() %>%
  group_by(`Partner Country`) %>%
  summarise("Total" = sum(`Trade Value usd`)) %>%
```

```
  filter(`Partner Country` != "World")
```

A partir destes dados, iremos criar um gráfico que mostre a composição relativa de cada país. Para isso, é preciso antes fazer algumas alterações na formatação dos dados, de modo que o dataframe final tenha a o número de quadrantes ocupado por cada país e suas posições.

```
export_peixe$prop = 100*export_peixe$Total/sum(export_peixe$Total)
export_peixe = expandRows(export_peixe, "prop")
export_peixe <- rbind(export_peixe,
                      export_peixe[sample(nrow(export_peixe), 100- nrow(export_peixe)), ])

var = export_peixe$`Partner Country`
nrows <- 10
categ_table <- round(table(var) * ((nrows*nrows)/(length(var))))

base <- expand.grid(y = 1:nrows, x = 1:nrows)
base$category <- factor(rep(names(categ_table), categ_table))



ggplot(base, aes(x = x, y = y, fill = category)) +
    geom_tile(color = "black", size = 0.5) +
    scale_fill_manual(values = as.character(GetColors(n = 9,
                                                      scheme = "roma",
                                                      alpha = 0.9))) +
    theme_void() +
    labs(title = "Destino das exportações de Peixe pelo Brasil (2020)") +
    scale_x_continuous(expand = c(0, 0)) +
    scale_y_continuous(expand = c(0, 0), trans = 'reverse') +
    theme(panel.border = element_rect(size = 2, fill = NA))
```



Com isso, vemos uma predominância dos Estados Unidos como destino das exportações de peixe.

12 Datação de recessões e ciclos econômicos no R com o algoritmo de Harding-Pagan

Ao longo do tempo a economia apresenta o que se chama de ciclos econômicos, ou seja, períodos de expansão e recessão. Mas de que forma podemos saber em qual ponto do ciclo econômico a economia se encontra? Como sabemos se a economia está em recessão? Estas são perguntas de grande interesse para acadêmicos e profissionais da área, e neste breve exercício demonstramos como replicar a datação de ciclos econômicos que instituições como NBER (EUA) e CODACE (Brasil) tradicionalmente publicam.

De maneira prática, neste exercício replicamos o algoritmo de Harding & Pagan (2002) para datar o ciclos de negócios do Produto Interno Bruto (PIB) brasileiro. Em resumo, o método considera algumas regras impostas ao comportamento de uma série temporal para classificar picos e vales. Recessão é o período entro o pico da atividade econômica e seu subsequente vale, ou ponto mínimo. Entre o vale e o pico, diz-se que a economia está em expansão.

O método é bastante simples e poderoso, conseguindo praticamente replicar a cronologia de recessões desenvolvidas pelas instituições mencionadas acima.

12.1 BCDating

Para aplicar o algoritmo utilizaremos o pacote `BCDating` na linguagem R, criado por Majid Einian (Central Bank of Islamic Republic of Iran) e Franck Arnaud (National Institute of Statistics and Economic Studies, France).

12.2 Dados

Neste exercício utilizaremos a série do PIB a preços de mercado (série encadeada do índice de volume trimestral com ajuste sazonal, média de 1995 = 100), disponível no SIDRA/IBGE. Para coletar os dados via API pode-se usar o pacote `sidrar`, especificando o código de coleta. Além disso realizamos a preparação dos dados para utilização posterior:

```
# Coleta e tratamento de dados
pib <- sidrar::get_sidra(api = "/t/1621/n1/all/v/all/p/all/c11255/90707/d/v584%202") %>%
  dplyr::select("date" = `Trimestre (Código)`, "value" = `Valor`) %>%
  dplyr::mutate(value = value, date = lubridate::yq(date)) %>%
  dplyr::as_tibble()
```

```
## All others arguments are desconsidered when 'api' is informed
```

12.3 Algoritmo de Harding & Pagan (2002)

Para aplicar o algoritmo e obter as datações de ciclo de negócios, primeiro transformamos o objeto pro formato de série temporal e, em seguida, utilizamos a função `BBQ()` do pacote `BCDating`. Optamos por deixar com os valores predefinidos os demais argumentos da função, que servem para definir os valores mínimos de duração do ciclo (pico ao pico ou vale ao vale) e da fase do ciclo (pico ao vale ou vale ao pico).

```
# Obter datação de ciclo de negócios
bc_dates <- pib %>%
  timetk::tk_ts(select = value, start = c(1996, 1), frequency = 4) %>%
  BCDating::BBQ(name = "Ciclo de Negócios do PIB do Brasil")
```

12.4 Resultados

Como pode ser visto abaixo, o objeto retornado traz como resultado as datas (trimestres) de picos e vales, assim como a duração do ciclo.

```
# Exibir resultados
show(bc_dates)
```

```
##    Peaks Troughs Duration
## 1 2001Q1  2001Q4        3
## 2 2002Q4  2003Q2        2
## 3 2008Q3  2009Q1        2
## 4 2014Q1  2016Q4       11
## 5 2019Q4  2020Q2        2
```

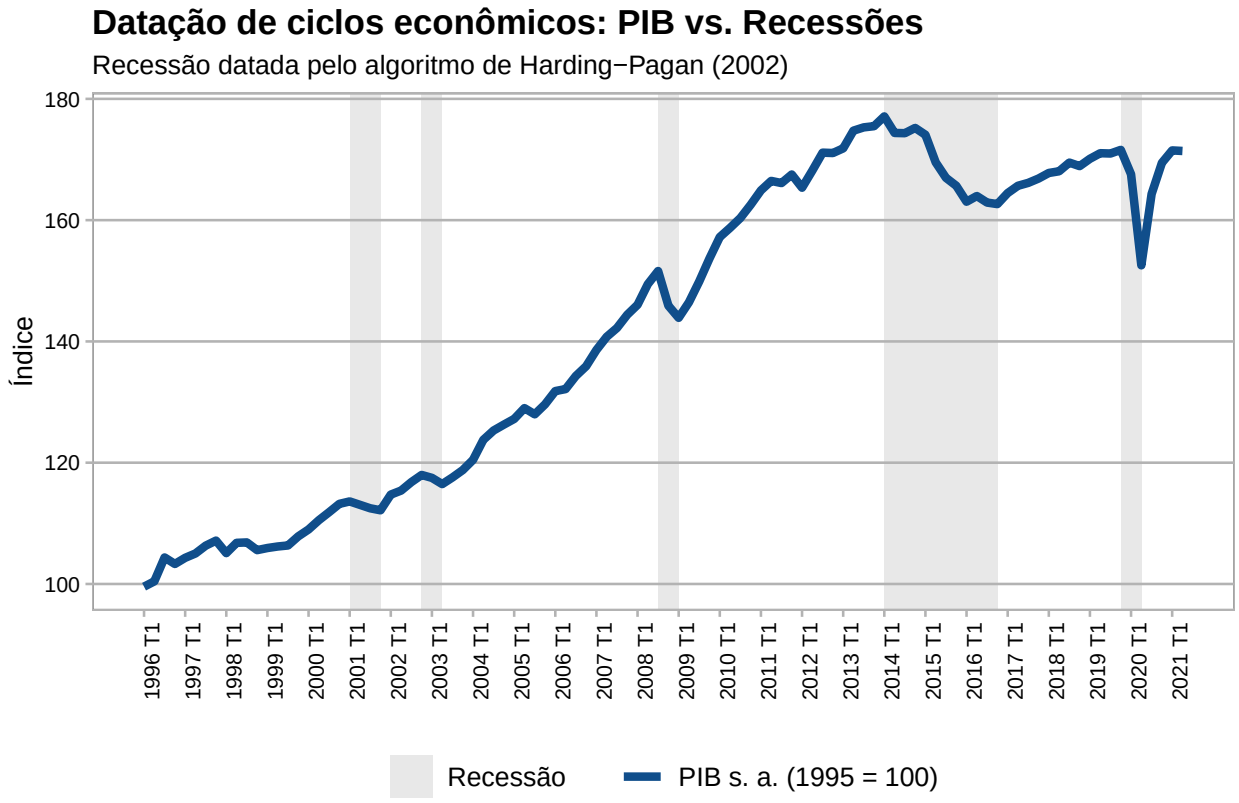
Outras informações podem ser obtidas com a função `summary()`:

```
# Informações adicionais
summary(bc_dates)
```

```
##        Phase ]Start  ;End] Duration LevStart LevEnd Amplitude
## 1  Expansion   <NA> 2001Q1       NA       NA    114        NA
## 2  Recession 2001Q1 2001Q4        3      114    112       1.5
## 3  Expansion 2001Q4 2002Q4        4      112    118       5.8
## 4  Recession 2002Q4 2003Q2        2      118    116       1.5
## 5  Expansion 2003Q2 2008Q3       21      116    152      35.1
## 6  Recession 2008Q3 2009Q1        2      152    144       7.7
## 7  Expansion 2009Q1 2014Q1       20      144    177      33.2
## 8  Recession 2014Q1 2016Q4       11      177    163      14.5
## 9  Expansion 2016Q4 2019Q4       12      163    172       8.9
## 10 Recession 2019Q4 2020Q2        2      172    153      19.0
## 11 Expansion 2020Q2   <NA>       NA      153     NA        NA
##
##           Amplitude Duration
## Exp=]T;P]      20.8     14.2
## Rec=]P;T]       8.8      4.0
```

Porém, o mais interessante é avaliar o resultado visualmente através de um gráfico. Para tal, fazemos um tratamento dos dados retornados pela função `BBQ()` e utilizamos o `ggplot2` para gerar o gráfico com as áreas sombreadas referente às datas de recessão que foram identificadas pelo algoritmo, acompanhadas do comportamento do PIB no período:

```
##    Peaks Troughs Duration
## 1 2001Q1  2001Q4        3
## 2 2002Q4  2003Q2        2
## 3 2008Q3  2009Q1        2
## 4 2014Q1  2016Q4       11
## 5 2019Q4  2020Q2        2
```



Fonte: analisemacro.com.br com dados de IBGE

13 Análise das Atas do COPOM com text mining

Mineração de texto, ou text mining, é um tópico muito interessante, pois há um potencial enorme de aplicações para obtenção de insights através dessa técnica envolvendo análise textual. Com a finalidade de demonstrar seu uso, neste post faremos uma breve e introdutória análise das atas do Comitê de Política Monetária - COPOM usando text mining com o auxílio do pacote `tidytext`.

As atas do COPOM são um caminho natural para qualquer economista em busca de uma fonte de dados para exercitar o text mining, já que a autoridade monetária realiza ajustes (mudanças) no texto a cada reunião, realizada a cada 45 dias. As atas são disponibilizadas publicamente neste link em arquivos PDFs (atualmente há 238 reuniões realizadas). Inicialmente vamos pegar o arquivo da última reunião publicada e importar os dados para o R, fazendo tratamentos e posterior análise. Por fim, vamos coletar os dados de todo o histórico de atas para refazer a análise de forma a incorporar o componente temporal.

Um ponto importante é que utilizaremos as versões em inglês das atas, apesar de o COPOM disponibilizar também em português. Isso se deve ao fato de que as ferramentas aqui utilizadas, para aplicar a técnica de text mining, funcionarem melhor com textos na língua inglesa.

13.1 Pacotes

Os pacotes utilizados neste exercício podem ser instalados/carregados com o gerenciador `pacman`, todos provenientes do CRAN:

```
# Instalar/carregar pacotes
pacman::p_load(
  "tidytext",
  "pdftools",
  "stopwords",
  "textdata",
  "dplyr",
  "tidyr",
  "lubridate",
  "magrittr",
  "knitr",
  "ggplot2",
```

```
  "ggthemes",
  "jsonlite",
  "purrr",
  "stringr",
  "scales",
  "forcats"
  )
```

13.2 Text mining de uma ata do COPOM

13.2.1 Dados

Vamos importar a última ata do COPOM para o R. Primeiro criamos um objeto para armazenar o link para o arquivo e, na sequência, usamos a função `pdf_text` do pacote `pdftools` para ler cada página do arquivo PDF e transformar os dados em um vetor de caracteres de tamanho igual ao número de páginas.

```
# URL da última ata do COPOM
www <- "https://www.bcb.gov.br/content/copom/copomminutes/MINUTES%20238.pdf"

# Ler arquivo PDF convertendo para caracter
raw_copom_last <- pdftools::pdf_text(www)
```

13.2.2 Tratamento de dados

Um primeiro olhar sobre os dados mostrará que há uma série de caracteres especiais "\n" e "\r" que indicam quebras de linhas. Portanto, vamos tratar de forma a obter um objeto `tibble` com uma coluna (text) com os dados do texto de cada página da ata, outra coluna (meeting) indicando o mês da reunião e a última coluna (page) que informará a página referente ao texto.

```
# Tratamento de dados
copom_last_clean <- dplyr::tibble(
    text = unlist(strsplit(raw_copom_last, "\r"))
    ) %>%
  dplyr::mutate(
    meeting = "May 2021",
    page = dplyr::row_number(),
```

```
    text = gsub("\n", "", text)
    )
```

13.2.3 Text mining

Agora estamos com os dados quase prontos para uma análise! O que faremos agora é começar a aplicar algumas das técnicas de text mining provenientes do livro Text Mining with R escrito pela Julia Silge e David Robinson.

O primeiro passo é "tokenizar" nossos dados, de forma a obter "unidades do texto" que servirão para a análise. Isso facilitará para realizarmos a contagem de palavras frequentes, filtrar palavras em específico, etc. O processo de "tokenizar" é definido no livro como:

"*A token is a meaningful unit of text, most often a word, that we are interested in using for further analysis, and tokenization is the process of splitting text into tokens.*"

Para utilizar a técnica, usamos a função `unnest_tokens` do pacote `tidytext` e na sequência realizamos a contagem das palavras encontradas:

```
# Text mining - Criar tokens
copom_last <- copom_last_clean %>%
  tidytext::unnest_tokens(word, text)

# Contar palavras
copom_last %>%
  dplyr::count(word, sort = TRUE) %>%
  dplyr::slice_head(n = 6) %>%
  knitr::kable()
```

word	n
the	174
of	76
in	35
and	31
to	31

word	n
for	24

Como podemos observar, as palavras mais frequentes são palavras comuns como “the,” “of,” “in,” etc. que não servirão muito para nossa análise. Essas palavras são chamadas de “stop words” no mundo do text mining. Vamos fazer uma tratativa para remover essas palavras usando como base o objeto `stop_words` proveniente do pacote `tidytext` e, antes, removemos também números que foram apontados como “palavras”:

```
copom_last_sw <- copom_last %>%
  # Remover palavras comuns (stop words)
  dplyr::anti_join(stop_words)%>%
  # Remover números
  dplyr::mutate(word = gsub("[^A-Za-z ]", "", word)) %>%
  # Contar palavras
  dplyr::count(word, sort = TRUE) %>%
  dplyr::filter(word != "")
```

```
## Joining, by = "word"
```

```
copom_last_sw %>%
  dplyr::slice_head(n = 6) %>%
  knitr::kable()
```

word	n
inflation	24
copom	19
monetary	18
economic	17
meeting	16
committee	15

Agora sim temos os dados prontos para uma análise de sentimento!

13.2.4 Análise de sentimento

Nosso foco agora é usar ferramentas de análise para tirar informação desses dados, ou seja, queremos saber o que as palavras das atas do COPOM podem indicar com base na tentativa de apontarmos um "score" para cada uma delas, usando datasets e bibliotecas de "sentimento do texto" do `tidytext` para isso.

Vamos ver quais são as palavras negativas e positivas usadas com mais frequência. A função `get_sentiments` do `tidytext` fará isso pra gente, bastando apontar um `lexicon`, que pode ser "bing," "afinn," "loughran" ou "nrc."

```
# Obter análise de sentimento das palavras com base em uma biblioteca
copom_last_sw %>%
  dplyr::inner_join(tidytext::get_sentiments("bing")) %>%
  dplyr::slice_head(n = 10) %>%
  knitr::kable()
```

```
## Joining, by = "word"
```

word	n	sentiment
risks	13	negative
slack	5	negative
recovery	4	positive
risk	3	negative
robust	3	positive
challenging	2	negative
achievement	1	positive
aggravate	1	negative
assure	1	positive
bias	1	negative

Como resultado, "risks" é a palavra negativa que aparece mais vezes (13 no total) nessa ata do COPOM, e "recovery," uma palavra positiva, é usada 4 vezes no total. Um segundo olhar sobre esse resultado pode levantar uma suspeita, pois "recovery" é apontada como uma palavra positiva

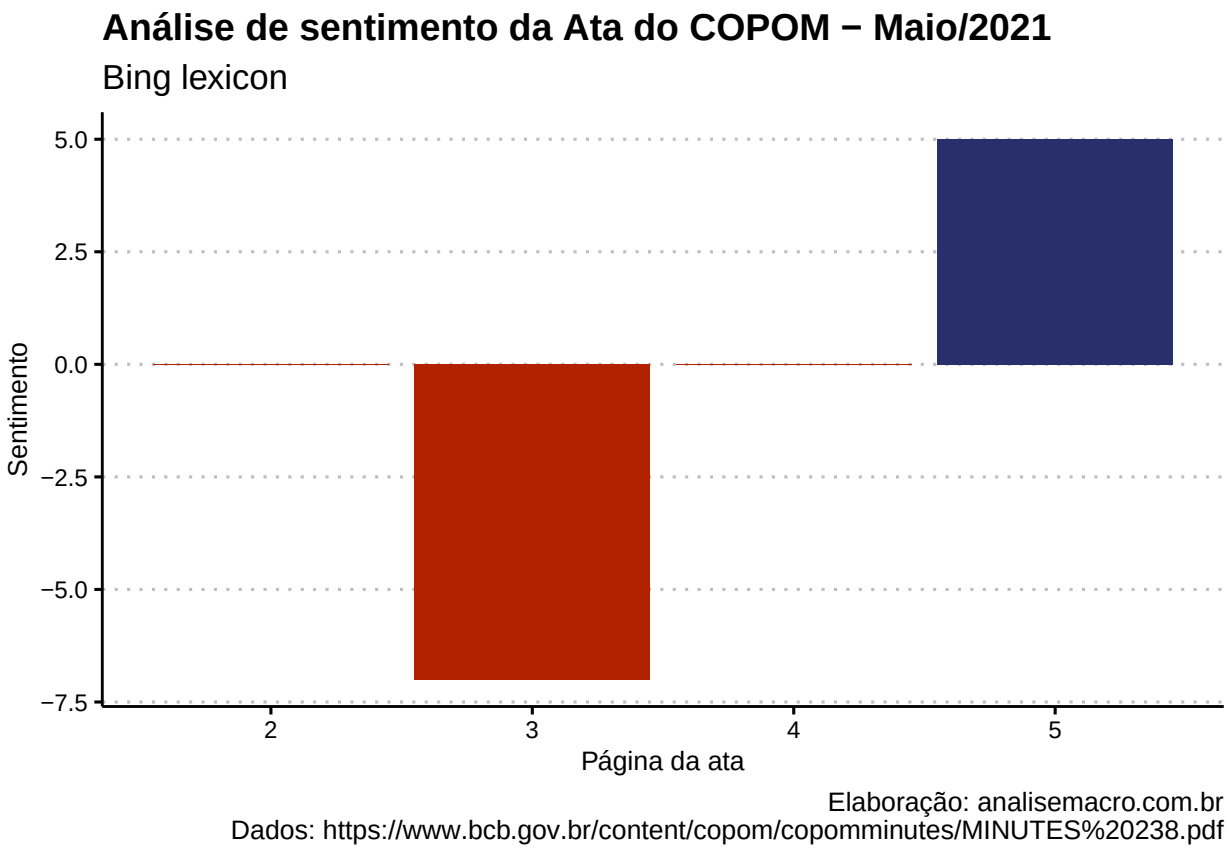
(recuperação), no entanto, é razoável supor que pode estar também associada com uma situação anterior negativa. Isso é um possível problema da análise que o leitor pode investigar.

Agora vamos explorar a análise graficamente:

```
# Análise de sentimento final da ata de Maio/2021
copom_sentiment <- copom_last %>%
  dplyr::inner_join(tidytext::get_sentiments("bing")) %>%
  dplyr::count(meeting, page, sentiment) %>%
  tidyr::pivot_wider(
    id_cols     = c(meeting, page),
    names_from  = sentiment,
    values_from = n,
    values_fill = 0
    ) %>%
  dplyr::mutate(sentiment = positive - negative)
```

```
## Joining, by = "word"
```

```
# Gerar gráfico
copom_sentiment %>%
  ggplot2::ggplot(ggplot2::aes(page, sentiment, fill = sentiment > 0)) +
  ggplot2::geom_col(show.legend = FALSE) +
  ggplot2::scale_fill_manual(values = c("#b22200", "#282f6b")) +
  ggplot2::labs(
    x        = "Página da ata",
    y        = "Sentimento",
    title    = "Análise de sentimento da Ata do COPOM - Maio/2021",
    subtitle = "Bing lexicon",
    caption  = paste0("Elaboração: analisemacro.com.br\nDados: ", www)
    )
```



O gráfico conta uma história interessante. O texto começou negativo na página 3 (sobre atualização de conjuntura, cenário e riscos), mas depois foi "neutro" na página 4 (sentimento positivo - negativo = 0) - onde é discutido a condução da política monetária e -, por fim, na última página o texto fica positivo com a decisão da reunião do COPOM.

Vale enfatizar que a primeira página, referente a capa, não é considerada na análise. Já a página 2, da contracapa, possui igual número de palavras negativas e positivas.

13.3 Text mining com todas as atas do COPOM

Agora vamos aplicar a técnica apresentada acima para um conjunto maior de dados, desta vez vamos pegar todas as atas disponíveis no site do Banco Central do Brasil - BCB através deste link.

Vamos expandir nossa análise capturando o texto de cada Ata do COPOM desde sua 42ª reunião, totalizando 198 atas para nossa análise. Faremos a comparação da frequência relativa de palavras e tópicos e veremos como o sentimento (conforme explorado acima) varia entre os relatórios.

13.3.1 Dados

Infelizmente, os links para as atas em PDF seguem um padrão irregular, mas felizmente para você, preparei um web-scrapping que automatizará o processo de captura dos links e dados. O código a seguir coletará os arquivos PDF e os deixará prontos para a mineração com o `tidytext`.

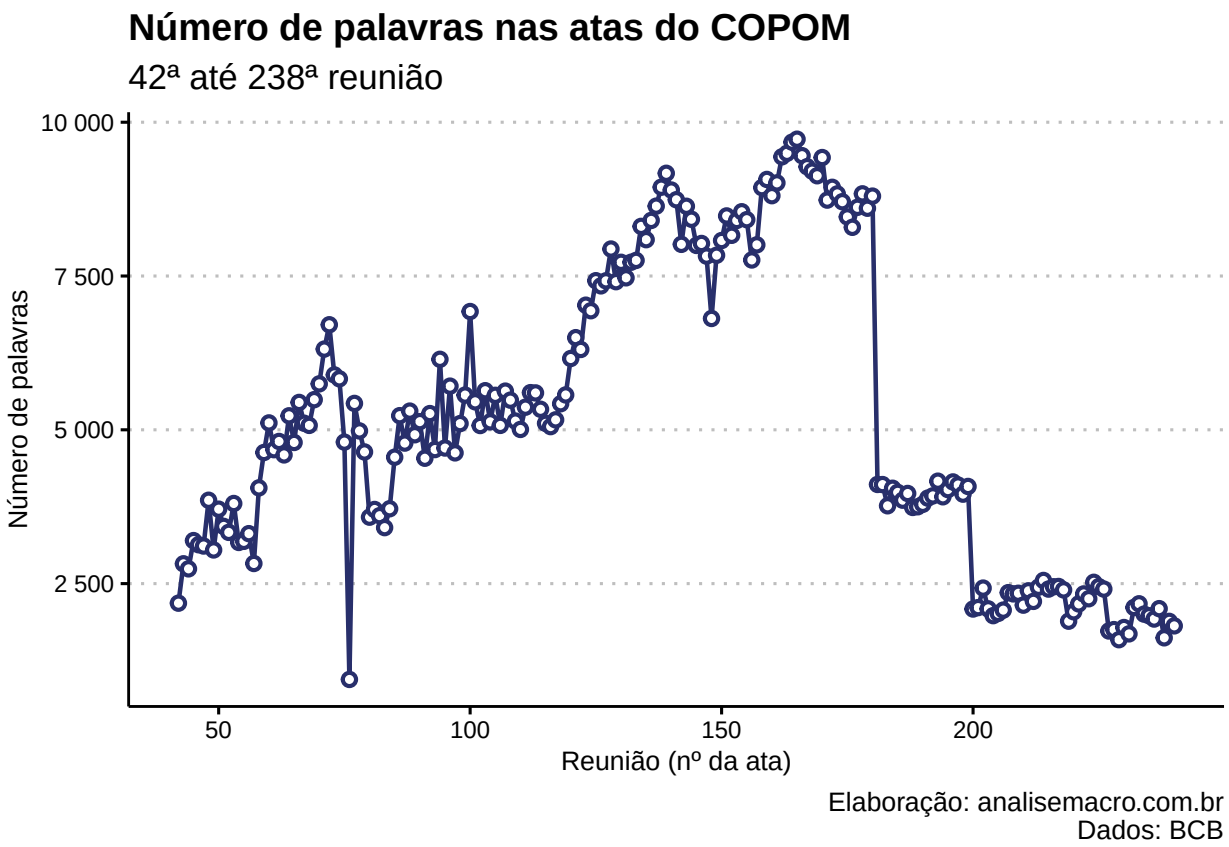
```
# URL para página com JSON dos links das atas
www_all <- "https://www.bcb.gov.br/api/servico/sitebcb/copomminutes/ultimas?quantidade=2000&fil

# Raspagem de dados
raw_copom <- jsonlite::fromJSON(www_all)[["conteudo"]] %>%
  dplyr::as_tibble() %>%
  dplyr::select(meeting = "Titulo", url = "Url") %>%
  dplyr::mutate(url = paste0("https://www.bcb.gov.br", url)) %>%
  dplyr::mutate(text = purrr::map(url, pdftools::pdf_text))

# Tratamento de dados
copom_clean <- raw_copom  %>%
  tidyr::unnest(text) %>%
  dplyr::filter(!meeting == "Changes in Copom meetings") %>%
  dplyr::group_by(meeting) %>%
  dplyr::mutate(
    page = dplyr::row_number(),
    text = strsplit(text, "\r") %>% gsub("\n", "", .),
    meeting = stringr::str_sub(meeting, 1, 3) %>%
      stringr::str_remove("[:alpha:]") %>%
      as.numeric()
    ) %>%
  dplyr::ungroup() %>%
  tidyr::unnest(text) %>%
  dplyr::arrange(meeting)
```

13.3.2 Estatística básica dos dados

Vamos ver o que conseguimos obter calculando algumas estatísticas básicas dos textos.

13.3.3 Número de palavras por ata

```
# Frequência de palavras por ata
copom_words <- copom_clean %>%
  tidytext::unnest_tokens(word, text) %>%
  dplyr::count(meeting, word, sort = TRUE) %>%
  dplyr::ungroup()

# Total de palavras por ata
total_words <- copom_words %>%
  dplyr::group_by(meeting) %>%
  dplyr::summarize(total = sum(n))

# Gerar gráfico
total_words %>%
  ggplot2::ggplot(ggplot2::aes(x = meeting, y = total)) +
  ggplot2::geom_line(color = "#282f6b", size = 0.8)+
  ggplot2::geom_point(
    shape  = 21,
    fill   = "white",
    color  = "#282f6b",
    size   = 1.6,
    stroke = 1.1
    ) +
  ggplot2::scale_y_continuous(labels = scales::number_format()) +
  ggplot2::labs(
    x        = "Reunião (nº da ata)",
    y        = "Número de palavras",
    title    = "Número de palavras nas atas do COPOM",
    subtitle = "42ª até 238ª reunião",
    caption  = "Elaboração: analisemacro.com.br\nDados: BCB"
    )
```



Percebe-se algumas mudanças ao longo do tempo, especialmente entre as reuniões número 180 e 181, onde houve remoção considerável de seções do comunicado na gestão Tombini. No mesmo sentido, de redução do total de palavras por comunicado, o início da gestão de Illan Goldfajn marcou mudança no layout e estrutura das seções da ata, alterações essas que permanecem em vigor até hoje.

A redução do tamanho dos comunicados, de forma geral, é certamente um ponto interessante que merece investigação em se tratando de qualidade de comunicação da política monetária.

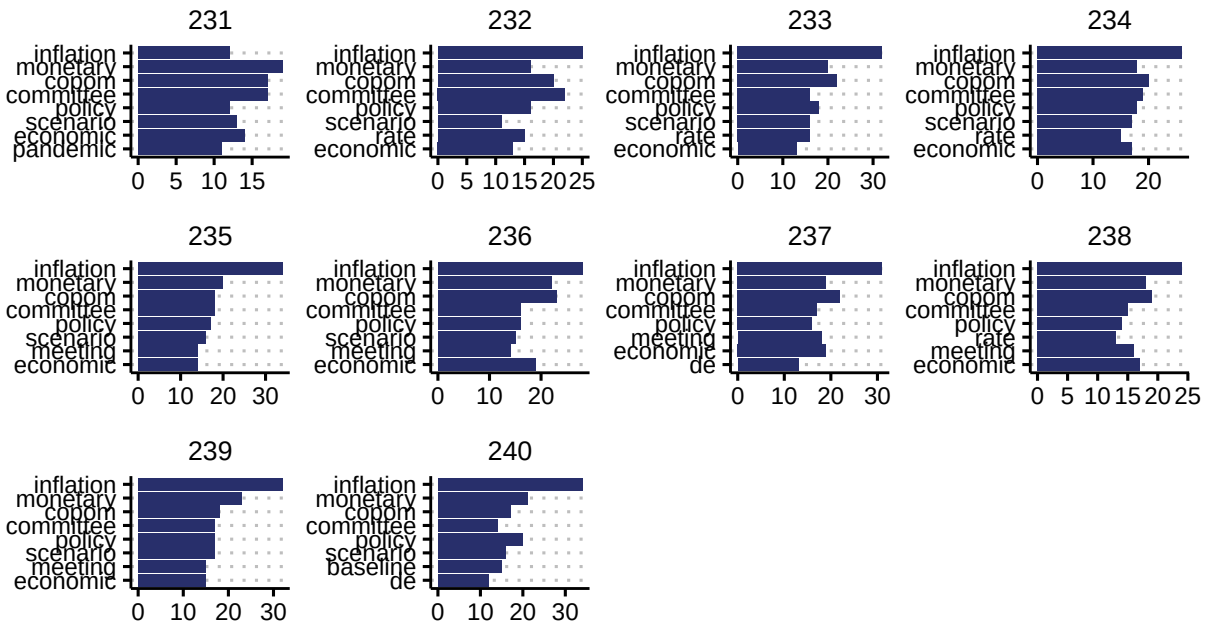
13.3.4 Sobre o que os diretores discutiram nas reuniões?

Vamos compilar uma lista das palavras usadas com maior frequência em cada ata. Como antes, vamos omitir as "stop words."

```
# Palavras por ata
copom_text <- copom_clean %>%
  dplyr::select(meeting, page, text) %>%
  tidytext::unnest_tokens(word, text)
```

```r
# Gerar gráfico
copom_text %>%
  dplyr::mutate(word = gsub("[^A-Za-z ]", "", word)) %>%
  dplyr::filter(word != "") %>%
  dplyr::anti_join(stop_words) %>%
  dplyr::group_by(meeting) %>%
  dplyr::count(word, sort = TRUE) %>%
  dplyr::mutate(rank = dplyr::row_number()) %>%
  dplyr::ungroup() %>%
  dplyr::arrange(rank, meeting) %>%
  dplyr::filter(rank < 9, meeting > 230) %>%
  ggplot2::ggplot(ggplot2::aes(y = n, x = forcats::fct_reorder(word, n))) +
  ggplot2::geom_col(fill = "#282f6b") +
  ggplot2::facet_wrap(~meeting, scales = "free", ncol = 4) +
  ggplot2::coord_flip() +
  ggplot2::labs(
    x        = "",
    y        = "",
    title    = "Palavras mais frequentes nas atas do COPOM",
    subtitle = "Excluídas palavras comuns (stop words) e números.",
    caption  = "Elaboração: analisemacro.com.br\nDados: BCB"
    )
```

```
## Joining, by = "word"
```



Como esperado, muitas discussões sobre inflação. Vamos tentar encontrar algo mais informativo com esses dados.

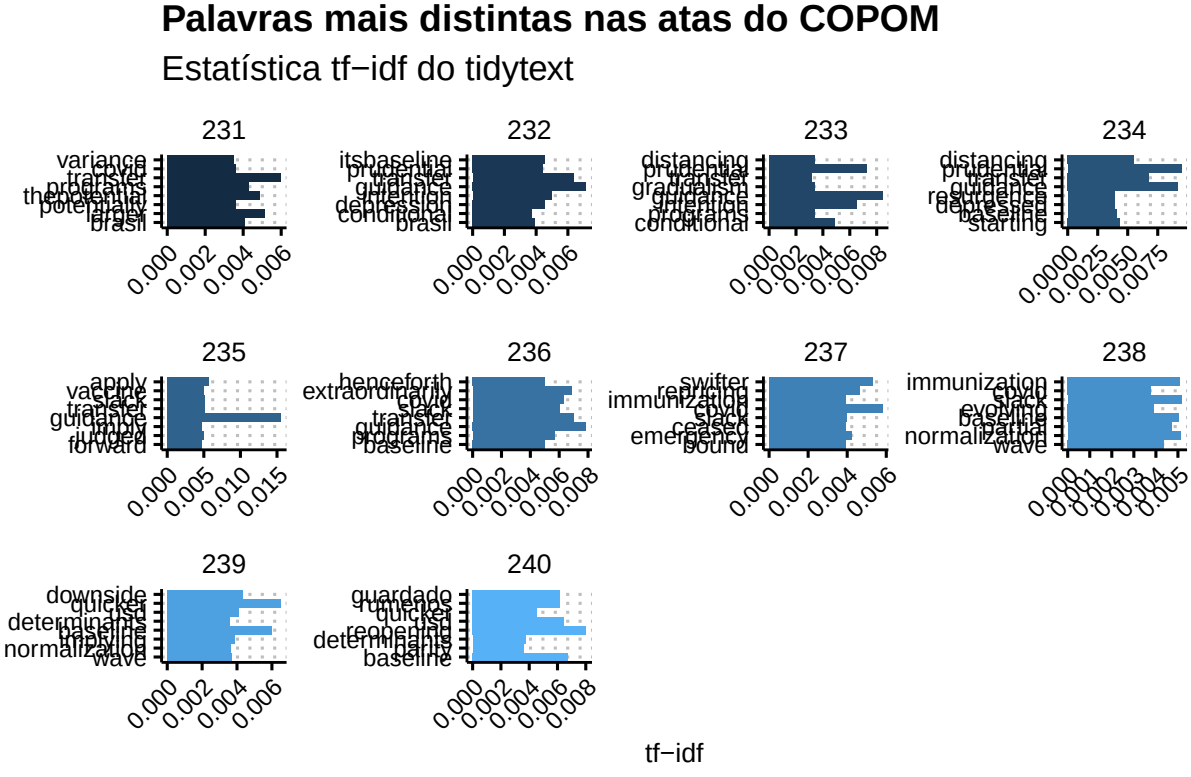
Conforme Silge e Robinson, podemos usar a função `bind_tf_idf` para juntar a frequência do termo (tf) e a frequência inversa do documento (idf) ao nosso conjunto de dados. Essa estatística diminuirá o peso em palavras muito comuns e aumentará o peso em palavras que só aparecem em algumas atas. Em essência, extrairemos o que há de especial em cada ata. As atas do COPOM sempre falarão muito sobre inflação e juros, e a estatística "tf-idf" pode nos dizer algo sobre o que é diferente em cada ata.

Também limparemos alguns termos adicionais que o `pdftools` captou (abreviações e palavras estranhas ou fragmentadas), aumentando nossa lista de "stop words."

```
# Stop words personalizadas
custom_stop_words <- dplyr::bind_rows(
  dplyr::tibble(
    word = c(
      tolower(month.abb),
```

```
      tolower(month.name),
      "one","two","three","four","five","six","seven","eight","nine","ten",
      "eleven","twelve", "wkh", "ri", "lq", "month", "wr", "dqg",
      "hdu", "jurzwk", "zlwk", "zlwk", "hfhpehu", "dqxdu", "kh", "sulfh", "dv",
      "kh", "prqwk", "hdu", "shulrg", "dv", "jurzwk", "wkdw", "zdv", "iru", "dw",
      "wkdw", "jrrgv", "xqh", "eloolrq", "eloolrq", "iluvw", "dq", "frqvxphu",
      "prqwk", "udwh", "sulo", "rq", "txduwhu", "vhfwru", "pandemic",
      "dffxpxodwhg", "hg", "kdyh", "sdqghg", "sulfhv", "rq", "sdqvlrq",
      "percent", "forvhg", "frpsduhg", "lqgh", "ryhpehu", "wklv", "kdv", "prqwkv",
      "bcbgovbr", "banco", "head"
      ),
    lexicon = c("custom")
    ),
  stop_words
  )

# Palavras por ata
copom_text_refined <- copom_text %>%
  dplyr::mutate(word = gsub("[^A-Za-z ]", "", word)) %>%
  dplyr::filter(word != "") %>%
  dplyr::group_by(meeting) %>%
  dplyr::count(word, sort = TRUE) %>%
  tidytext::bind_tf_idf(word, meeting, n) %>%
  dplyr::arrange(desc(tf_idf))

# Gerar gráfico
copom_text_refined %>%
  dplyr::anti_join(custom_stop_words, by = "word") %>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  dplyr::group_by(meeting) %>%
  dplyr::mutate(id = dplyr::row_number()) %>%
  dplyr::ungroup() %>%
  dplyr::filter(id < 9, meeting > 230) %>%
```

```
ggplot2::ggplot(ggplot2::aes(y = tf_idf, x = word, fill = meeting)) +
ggplot2::geom_col(show.legend = FALSE) +
ggplot2::facet_wrap(~meeting, scales = "free", ncol = 4) +
ggplot2::coord_flip() +
ggplot2::labs(
  x        = "",
  y        = "tf-idf",
  title    = "Palavras mais distintas nas atas do COPOM",
  subtitle = "Estatística tf-idf do tidytext",
  caption  = "Elaboração: analisemacro.com.br\nDados: BCB"
  ) +
ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1))
```



Esse gráfico já mostram uma história interessante dessa amostra que escolhemos. Podemos observar preocupações com o BREXIT, reformas e agenda econômica, o surgimento do termo "covid" a partir da ata nº 231 e subsequente adoção do *forward guidance* e, por fim, também como destaque, a mudança da política para uma "normalização parcial" (termos "partial" e "normalization").

13.3.5 Comparando o sentimento entre as atas

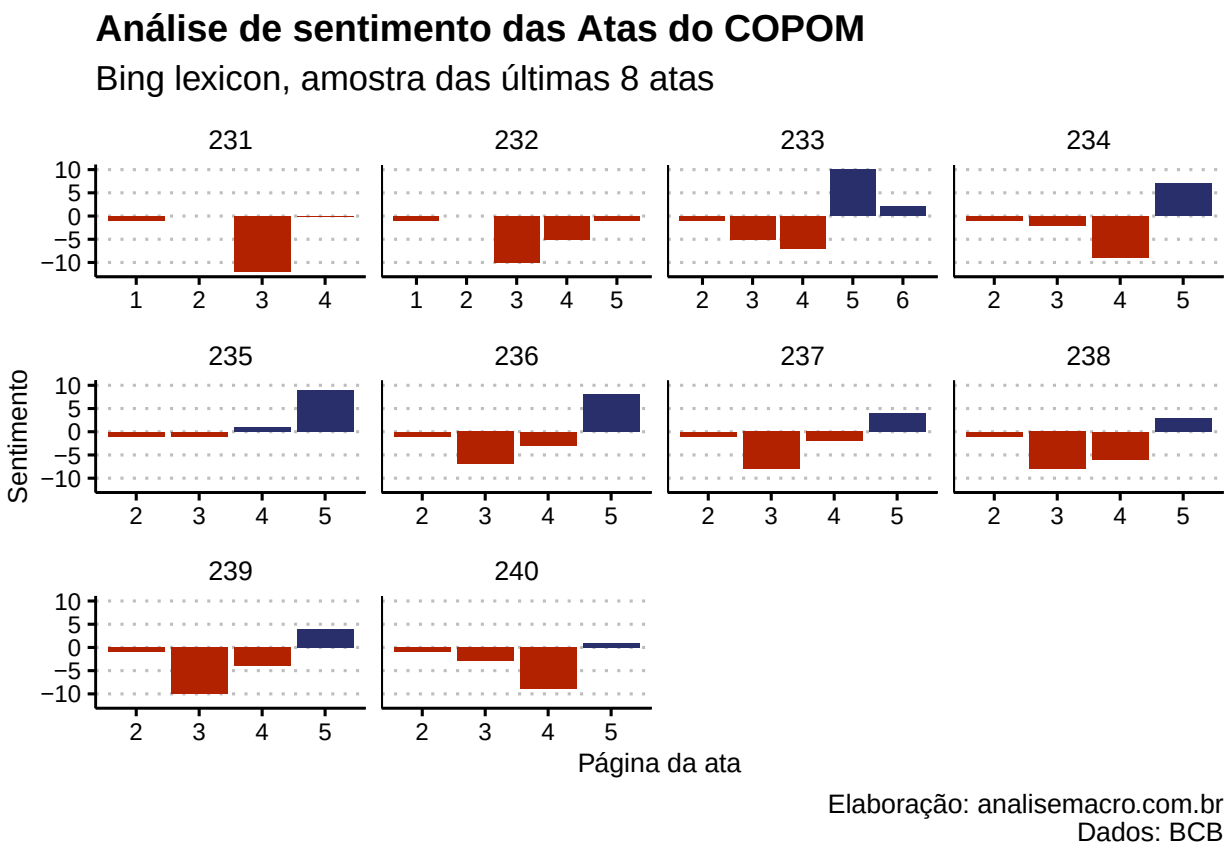
Como o sentimento variou entre as atas? Vamos usar a abordagem que usamos no início para a ata de maio/2021 e aplicá-la a cada ata.

```
# Análise de sentimento das atas
copom_sentiment_all <- copom_text %>%
  dplyr::anti_join(stop_words) %>%
  dplyr::inner_join(tidytext::get_sentiments("bing")) %>%
  dplyr::count(meeting, page, sentiment) %>%
  tidyr::pivot_wider(
    id_cols     = c(meeting, page),
    names_from  = sentiment,
    values_from = n,
    values_fill = 0
    ) %>%
  dplyr::mutate(sentiment = positive - negative)
```

```
## Joining, by = "word"
## Joining, by = "word"
```

```
# Gerar gráfico
copom_sentiment_all %>%
  dplyr::filter(meeting > 230) %>%
  ggplot2::ggplot(ggplot2::aes(page, sentiment, fill = sentiment > 0)) +
  ggplot2::geom_col(show.legend = FALSE) +
  ggplot2::scale_fill_manual(values = c("#b22200", "#282f6b")) +
  ggplot2::facet_wrap(~meeting, ncol = 4, scales = "free_x") +
  ggplot2::annotate("segment", x = -Inf, xend = Inf, y = -Inf, yend = -Inf)+
  ggplot2::annotate("segment", x = -Inf, xend = -Inf, y = -Inf, yend = Inf) +
  ggplot2::labs(
    x        = "Página da ata",
    y        = "Sentimento",
    title    = "Análise de sentimento das Atas do COPOM",
    subtitle = "Bing lexicon, amostra das últimas 8 atas",
```

```
    caption  = "Elaboração: analisemacro.com.br\nDados: BCB"
    )
```



O resultado mostra que o sentimento mudou consideravelmente no texto das atas da amostra das últimas 8 reuniões, com forte predominância "negativa" desde o advento da pandemia da Covid-19.

13.4 Conclusão

Isso é tudo por hoje. Essas técnicas são muito interessantes e promissoras, merecendo exploração aprofundada. Certamente o exercício aqui apresentado possui falhas e pontos de melhoria, mas serve como uma introdução ao tema. Espero que goste, caro leitor!

Possui interesse no tema de política monetária? A Análise Macro disponibiliza o curso de **Teoria da Política Monetária** dentro da temática de *Central Banking*. Aproveite!

14 Como continuar aprendendo

Ao longo dos exercícios desse `Ebook`, procuramos mostrar como *deve ser* o comportamento de um **economista quantitativo** ao lidar com dados. O processo de extração, limpeza, análise e apresentação deve ser concentrado em uma plataforma unificada como o R, onde todas as etapas do *ciclo do dado* conversam e se retroalimentam entre si.

O ganho de produtividade ao construir esse tipo de metodologia é enorme para o profissional e para a empresa onde o mesmo trabalha. Análises que levariam dias podem ficar prontas em poucos minutos. Além disso, a *reprodutibilidade* das análises é garantida a partir de scripts que podem ser compartilhados entre diferentes membros da equipe.

Ao longo dos exercícios, você viu uma gama de exemplos de como coletar dados a partir de fontes diversas. Hoje, o R conta com uma infinidade de pacotes de coleta de dados, interagindo com praticamente todas as fontes de dados e outros programas livres ou proprietários (que dependem de licença para uso). Todos os dias, diga-se, novos pacotes são adicionados à linguagem, tornando o R uma referência para quem lida com análise de dados.

Há dentro do ecosistema, ainda, uma infinidade de pacotes que lidam com o processo de limpeza do dado. Filtrar, selecionar, criar novas variáveis, tratar dados faltantes, etc, etapas rotineiras de um `economista quantitativo`, podem ser facilmente implementadas com a linguagem.

O `ciclo do dado` é completado com as etapas de modelagem e apresentação dos dados. No R, encontra-se disponível uma gama enorme de modelos, que pode ser facilmente implementado a partir de poucas linhas de código. Além disso, você pode utilizar o `RMarkdown` para produzir um relatório ou mesmo um `Ebook` como esse que você está tendo acesso.

Como dito na introdução, a origem desses exercícios que você teve acesso são o Clube AM, o espaço de compartilhamento de códigos da Análise Macro. Lá, compartilhamos exercícios semanalmente com os Membros em nossa plataforma exclusiva, além de fomentarmos, via grupo fechado no Whatsapp, a interação e networking entre eles.

Se você quer `se tornar um economista quantitativo` e praticar todos os dias novos exercícios de análise de dados, como esses que viu aqui no `Ebook`, seja muito bem-vindo a conhecer o Clube. Será um prazer tê-lo conosco!

15 Referências

Grolemund, G., and H. Wickham. 2017. *R for Data Science*. O'Reilly Media.