Universidade Federal do Paraná
Setor de Ciências Exatas
Departamento de Estatística e Departamento de Informática
Programa de Especialização em *Data Science* e *Big Data*

IAGO HAUER CHIARELLI

ANÁLISE DOS INDICADORES CONTÁBEIS E BETAS DE MERCADO: ETL A PARTIR DE DADOS ABERTOS DA CVM E QUANTMOD

Curitiba
2025

Iago Hauer Chiarelli

ANÁLISE DOS INDICADORES CONTÁBEIS E BETAS DE MERCADO: ETL A PARTIR DE DADOS ABERTOS DA CVM E QUANTMOD

Artigo apresentado ao Programa de Especialização em *Data Science* e *Big Data* da Universidade Federal do Paraná como requisito parcial para a obtenção do grau de especialista.

Orientador: Prof. Dr. Paulo Justiniano Ribeiro Junior

Curitiba

2025

Análise dos indicadores contábeis e betas de mercado: ETL a partir de dados abertos da CVM e Quantmod

Título em inglês

Iago Hauer Chiarelli[1], Prof. Dr. Paulo Justiniano Ribeiro Junior[2]

[1] Aluno do programa de Especialização em Data Science & Big Data.

[2] Departamento de Estatística (DEST/UFPR), Setor de Ciências Exatas, Universidade Federal do Paraná (UFPR).

Resumo

O presente estudo investiga a capacidade explicativa de informações contábeis sobre o risco sistemático de empresas listadas na B3, mensurado por coeficientes beta, estimados no período subsequente à divulgação de tais demonstrações (betas futuros). O trabalho também documenta a concepção e implementação de um fluxo de ETL para extração e padronização de dados públicos do Portal Dados Abertos CVM, integrando rotinas em Python para extração e transformação das informações relevantes, armazenadas em arquivos CSV, e rotinas em R para reestruturação dos dados e cálculo dos indicadores. Os betas de mercado serviram de variável resposta, extraídos utilizando o Índice Ibovespa como proxy através do pacote quantmod do R. A amostra é composta por ações listadas na B3 que faziam parte do índice no período de 2018–2024. Antes da estimação de modelos, as variáveis contábeis foram transformadas nos seguintes formatos: *Trailing Twelve Months* (TTM), transformação logarítmica, robust scale e min–max. Os resultados indicam que determinadas métricas contábeis (principalmente o lucro líquido e o ROE) apresentam associação estatística consistente com o beta futuro, com R^2 ajustado na faixa aproximada de 0,14 a 0,16, mas também revelam heterogeneidade, formação de *clusters* e limitações de generalização. Este trabalho contribui ao estimular, e facilitar, a obtenção de informações contábeis por meio de fontes de dados abertos e ferramentas gratuitas. Adicionalmente, apresenta evidências empíricas sobre indicadores contábeis relevantes ao risco sistemático.
Palavras-chave: Indicadores contábeis, Risco sistemático, ETL, Python, Linguagem R.

Abstract

This study investigates the explanatory capacity of accounting information regarding the systematic risk of companies listed on B3, measured by betas coefficients, estimated in the period following the disclosure of financial statements (forward betas). The paper also documents the design and implementation of an ETL pipeline for extracting and standardizing public data from the CVM Open Data Portal, integrating Python routines for data extraction and transformation (stored in CSV files), as well as R routines for data restructuring and indicator calculations. Market betas, used as the response variable, were extracted using the Ibovespa Index as a proxy via R's quantmod package. The sample consists of stocks listed on B3 that were part of the index between 2018 and 2024. Prior to model estimation, accounting variables were transformed into the following formats: Trailing Twelve Months (TTM), logarithmic transformation, robust scaling, and min-max normalization. The results indicate that certain accounting metrics (primarily net income and ROE) exhibit a statistically consistent association with forward beta, with adjusted R^2 values ranging approximately between 0.14 and 0.16. However, the findings also reveal heterogeneity, the presence of clusters, and limitations in generalizability. The contributions of this work aim to encourage, and facilitate, to obtain accounting information through open data sources and free tools. Additionally, it provides empirical evidence about accounting indicators relevant to systematic risk.
Keywords: Accounting information. Systematic risk. ETL. Python. R Language.

1 Introdução

As informações contábeis são amplamente reconhecidas na literatura financeira como uma base fundamental para a análise de empresas. Assim como a história cumpre o papel de 'laboratório' das ciências sociais, oferecendo a base empírica para o estudo de fenômenos humanos, as informações contábeis funcionam como o registro experimental da realidade econômico-financeira de uma empresa. Escritas em sua linguagem própria, elas não apenas documentam o passado da organização, mas servem de fundamento material para análises e decisões futuras, tal como a história sustenta o conhecimento nas ciências sociais aplicadas. A base deste trabalho são as

Demonstrações Financeiras Padronizadas (DFP) e Informações Trimestrais (ITR), disponibilizadas pela Comissão de Valores Mobiliários (CVM) a partir do Portal Dados Abertos CVM (2025). Estas são informações periódicas, padronizadas e de divulgação obrigatória para as empresas listadas na B3 S.A. - Brasil, Bolsa, Balcão (B3).

A partir destas bases, é possível extrair informações contábeis necessárias para a formulação de índices econômico-financeiros, estes serão a variável explicativa. Enquanto a variável resposta é o risco sistemático, definido nos capítulos seguintes, extraído através do pacote quantmod do R (RYAN et al, 2025).

Nesse contexto, o presente trabalho combina duas frentes complementares: análise empírica da relação entre indicadores contábeis e risco sistemático (beta futuro) e desenvolvimento de um fluxo de *Extract, Transform, Load* (ETL), utilizando Python e R,[1] para tornar essa análise replicável a partir das bases públicas da CVM. Outros pacotes e programas fazem isso de forma automática, como o próprio pacote quantmod utilizado neste estudo, porém, com o objetivo de avaliar a disponibilidade dos dados oficiais, foi testado o uso dos arquivos diretamente da CVM, por serem fontes primárias para este tipo de estudo. Mas, devido a limitações no acesso aos dados dos preços de fechamento das ações estudadas, dados necessários para a formulação da variável resposta, nesta etapa optou-se pelo uso de dados secundários.

Este trabalho está dividido em 7 seções incluindo esta introdução. A seguir será detalhada brevemente a fundamentação teórica. Em seguida, discute os objetivos, as variáveis e as bases de dados utilizada. O quarto capítulo discute o processo de ETL descrevendo as principais funções elaboradas no Python e no R, assim como os desafios enfrentados nesta etapa. Em seguida, é descrita a parte de inferência estatística, utilizando o R. O penúltimo capítulo discute os resultados, seguido das conclusões finais.

2 Fundamentação Teórica

Este trabalho adota como referencial teórico a literatura que destaca o papel fundamental das informações contábeis na análise financeira (AMORIM et al, 2012; KOBORI, 2011; ROSS et al, 2013). Estas servem de base para alguns dos indicadores chamados fundamentalistas na análise financeira. Conforme evidenciado por Amorim, Lima e Murcia (2012, p. 199-200), "tais informações refletem a realidade econômico-financeira da empresa em um dado período". São dados de divulgação obrigatória e periódica para empresas diante da Receita Federal. Além disso, por exigência da B3 e CVM, as empresas listadas são obrigadas a divulgarem seus resultados de forma aberta (CVM, 2025). Inclusive, este foi um dos motivos para a opção do Portal Dados Abertos da CVM como base de dados. A contabilidade “é o idioma dos negócios” (WARREN BUFFETT, apud KOBORI, 2011, p. 68), tem o objetivo de ser uma linguagem padronizada, mas ainda assim, como todo idioma é um “idioma imperfeito” (ibid), com algumas limitações que serão apresentadas nos capítulos seguintes.

Essas informações contábeis fornecem a base indispensável para a avaliação de riscos, particularmente do risco sistemático, objeto central desta seção. A literatura acadêmica e a prática profissional reconhecem dois tipos de risco: o risco sistemático (não diversificável) e o risco não sistemático (diversificável).

Conforme descrito por Ross e outros autores (2013, p. 434), o risco sistemático é aquele que afeta um grande número de ativos, também denominado risco de mercado. Já o risco não sistemático refere-se a riscos inerentes a uma empresa ou ativo específico, podendo ser chamado de risco específico. Esse tipo de risco pode afetar um ou mais ativos, mas, à medida que a carteira de investimentos é diversificada, sua influência tende a diminuir.

Como a maioria das análises econômico-financeiras envolve um grande conjunto de empresas e ativos, visando maior poder de generalização, o estudo do risco sistemático é o mais abordado. Alguns autores assumem, inclusive, como pressuposto analítico, que o risco específico pode ser eliminado via diversificação, o que permite pesquisas mais abrangentes e didáticas (AMORIM et al, 2011; KOBORI, 2011). Embora amplamente discutido na teoria e na prática, mesmo que reduzido o risco nunca pode ser completamente eliminado. Contudo, o risco não sistemático é aquele efetivamente redutível pela diversificação.

Para mensurar o risco sistemático, um indicador amplamente utilizado na literatura é o Beta de mercado. Kobori (2011, p. 49) resume sua aplicação da seguinte forma:

“O risco de um ativo é a variância de seus resultados em relação à média do mercado. [...] é a variância nos preços da ação em relação aos preços do mercado, ou seja, o Índice Bovespa. Esse risco é expresso pelo beta (β).” (KOBORI, 2011, p. 49)

É importante destacar que o beta reflete médias observadas em um dado período. É inviável, senão impossível, prever o risco com exatidão. Aliando essa concepção à teoria da aversão ao risco (AMORIM et al, 2011; KOBORI, 2012; ROSS et al, 2013), é possível deduzir a existência de casos de betas não constantes

[1] A opção pelo uso das duas linguagens se insere no contexto do curso de Especialização em Data Science & Big Data da Universidade Federal do Paraná (UFPR). Ao longo do curso, ambas as linguagens foram amplamente utilizadas. A escolha teve por objetivo acompanhar a didática do curso no uso destas duas ferramentas.

para movimentos de alta e de baixa do mercado. Por exemplo, hipoteticamente, uma ação com beta baixo (mais estável) durante quedas poderia apresentar um beta ligeiramente maior em cenários de alta. O oposto ocorreria com ações mais voláteis (betas > 1), que poderiam ter um beta menor para subidas, considerando o pressuposto da aversão ao risco (em que empresas voláteis apresentam, em média, risco maior que as demais no mercado). Uma análise mais detalhada desses pressupostos pode ser objeto de estudos futuros, baseados nos algoritmos e na base de dados utilizados nesta pesquisa.

3 Objetivos, variáveis e base de dados

O presente estudo persegue objetivos complementares e interdependentes. Inicialmente, o objetivo foi investigar a capacidade explicativa de indicadores contábeis, de forma isolada, extraídos das demonstrações financeiras públicas, a partir dos dados da CVM, sobre o risco sistemático das empresas listadas na B3, mensurado por betas estimados no período subsequente à divulgação das informações (a seguir nomeado "betas futuros"). O segundo desafio, que demandou maior atenção neste trabalho, consiste em implementar e documentar o fluxo de extração, transformação e carregamento (ETL).[2] Este fluxo foi específico para extrair as informações das bases públicas da CVM, Nesta etapa, utilizou-se principalmente a linguagem Python, com o auxílio do editor Visual Studio Code e do ambiente Jupyter Notebook. Este conjunto de processos será detalhado adiante no texto.

A delimitação das variáveis obedeceu à lógica de isolar o efeito de informações contábeis que não incorporassem diretamente preços de mercado, evitando assim a "contaminação" do modelo por variáveis refletidas nas cotações diárias (por exemplo, flutuações cambiais, variações na taxa de juros, choques de commodities, eventos idiossincráticos e o "efeito manada"[3]). Dessa forma, as variáveis independentes foram construídas exclusivamente a partir de dados contábeis reportados nos formulários DFP e ITR de cada empresa, em seguida, foram organizadas a partir dos *tickers*.[4] Após extraídos, estes dados foram transformados em *Trailing Twelve Months* (TTM), esta etapa foi necessária para redução de sazonalidade, afinal, alguns setores podem ter maior ou menor atividade em dado período do ano, por exemplo, férias da maioria dos serviços no final do ano, enquanto há aumento das vendas do varejo no mesmo período devido às festividades e datas comemorativas.

Os dados foram obtidos a partir do Portal Dados Abertos CVM (CVM, 2025) que disponibiliza as Demonstrações Financeiras Padronizadas (DFP), Informações Trimestrais (ITR) e Informações Cadastrais. Estas informações são organizadas por ano e disponíveis em formato *Comma-Separated Values* (CSV).

Como o objetivo é identificar os efeitos dos indicadores sobre o risco sistemático, detalhado anteriormente, a variável resposta é este risco medido no Beta Futuro. Este foi calculado a partir do pacote quantmod que carrega e organiza as informações de várias fontes, utilizando o Yahoo como fonte de dados padrão (RYAN et al, 2025, p. 50).

A amostra de empresas foi selecionada a partir das empresas que faziam parte do Índice Ibovespa (BVSP) no período de análise (2018-2024), assim, ficava mais próximo do índice como um todo, embora certas empresas tenham pesos diferentes neste índice (B3, 2025a). Como o site oficial da B3 não fornece índices históricos, apenas os mais recentes, foi preciso utilizar a *Wayback Machine* da biblioteca The Internet Archive. A partir desses dados apenas algumas empresas estavam listadas em todo este período e também não foi possível importar as informações de todas, por isso, a amostra final resultou em 91 ações que representam um total de 85 empresas, pois algumas empresas podem possuir mais de um tipo de ação em circulação, como ordinária, preferencial ou unit.

4 ETL

O processo de ETL foi implementado em Python com as seguintes bibliotecas: `pandas` para manipulação dos dados em CSV e *Dataframe*; `ThreadPoolExecutor` combinado com `functools.partial` para paralelização das operações de extração; `os`, `re` e `unicodedata` para tratamento de caminhos de arquivos, expressões regulares e normalização de texto; `charset_normalizer` para detecção robusta de *encoding* nos arquivos CSV; e `logging` para monitoramento estruturado do pipeline, substituindo o uso convencional de `print()` por um sistema de

[2] ETL é a sigla comumente utilizada neste contexto, que significa Extract, Transform, Load, este é literalmente o termo em inglês do processo de extração, transformação e carregamento. O estudo utiliza esta sigla de forma a manter os padrão da nomenclatura acadêmica e profissional sobre o tema.

[3] Efeito Manada é um fenômeno estudado nas áreas de psicologia social, economia comportamental e teoria da decisão, que mostra como a pressão social e a busca por validação coletiva força individuos a tomarem decisões de forma quase inconsciente. Para mais informações, uma análise didática deste efeito pode ser observada neste artigo da B3 disponível em: <https://borainvestir.b3.com.br/objetivos-financeiros/organizar-as-contas/efeito-manada-como-ele-influencia-decisoes-e-afeta-investimentos-e-financas/> Acesso em 13/08/2025.

[4] Ticker é o nome resumido do código de negociação de uma empresa na bolsa. Também é o nome pelo qual o código é chamado no R, logo, para manter o padrão, este é o termo que será adotado ao longo do texto.

registros (*logs*) padronizados, facilitando a depuração; o controle do tempo de execução, e marcação de arquivos, foi controlado via `datetime`. Este *script* na íntegra está disponível no Anexo I.

A operação se iniciou com a configuração do logging para obter uma resposta no output sobre o funcionamento do *script*, assim como possíveis mensagens de erro. Este método teve o objetivo de facilitar o trabalho de correção de erros, manutenção e verificação do andamento do *script*. Para reconhecer o tempo de execução e facilitar verificações, também foi utilizado datetime do pacote 'os'.

Os informes contábeis estavam organizados em pastas com subpastas que continham os arquivos em csv. A base dos dados brutos resultou em 7,27 GB de arquivos em CSV. Como apenas algumas informações seriam utilizadas neste trabalho, foi feito uma operação de *loop* que percorria as seguintes variáveis no Python:

- anos = [2018, 2019, 2020, 2021, 2022, 2023, 2024]
- informes = ['BPA','BPP','DRE']
- tipos = ['ind','con']
- itr_dfp = ['itr','dfp']

Estas buscavam os informativos: Balanço Patrimonial Ativo (BPA), Balanço Patrimonial Passivo (BPP), Demonstração de Resultado (DRE). Como as informações são trimestrais, elas são armazenadas em 2 formatos, 'ind' é o informe individual, que contabiliza apenas o trimestre de referência, enquanto 'con' é o informe consolidado, que captura todo o movimento desde o início do ano até a data de referência do trimestre. Os 3 primeiros trimestres estão disponíveis no Formulário de Informações Trimestrais (ITR), o último trimestre do ano é a informação mais completa e organizada, resultando no Formulário de Demonstrações Financeiras Padronizadas (DFP). Como o objetivo era capturar o TTM de cada ação, eram necessários completar 4 trimestres, esta operação exigia juntar ou o informe consolidado do quarto trimestre, ou os informes individuais no caso dos três primeiros trimestres. Para isso foram importados apenas os informes tipo 'ind' do ITR e os dois tipos no DFP.

Inicialmente, a pesquisa contava com a padronização do plano de contas referenciais do Sistema Público de Escrituração Digital (Sped) para buscar as contas necessárias para formar os índices (RECEITA FEDERAL DO BRASIL, 2025). Dependia então destas três variáveis para o próximo loop:

- `df['CD_CONTA']`: Coluna do código da conta contábil;
- informe: para buscar no respectivo informe a conta correspondente;
- `df['DS_CONTA']`: Coluna com descrição da conta contábil, para fins de controle e análise.

O *script* procurava os valores das seguintes contas contábeis das empresas: Ativo Total no BPA; Passivo Total e Patrimônio Líquido no BPP; Receita Bruta, Lucro Bruto, Despesas Operacionais e Lucro Líquido do DRE.

Com o *script* atual, é possível procurar outras contas contábeis adaptando alguns mapeamentos de busca. No entanto, em alguns casos, o código da conta contábil não era consistente, principalmente no DRE. O mapeamento não pôde ser realizado apenas via dicionário de variáveis, sendo necessário o uso de expressões regulares (REGEX) implementadas com o pacote '`re`'.

No caso do DRE, foi feito também um mapeamento dos níveis de conta via REGEX, uma vez que as contas necessárias estavam em subníveis específicos.

Alguns arquivos apresentavam erros de leitura, foi então necessário criar uma função de detectar encoding, através dos pacotes '`charset_normalizer`' e '`unicodedata`'.

A maioria dos arquivos csv estavam em encoding 'cp1252' mas alguns poucos estavam em 'utf-8'. Para automatizar a detecção e importação dos arquivos, a função '`detectar_encoding(path, sample_size)`' lia o caminho do arquivo, então criava a variável 'enc' que era utilizada como valor de um argumento nomeado passado para o parâmetro 'encoding' da função '`pd.read_csv`'.

Além disso era preciso normalizar os textos das variáveis em string, como os dados estão em língua portuguesa e a maioria dos programas utiliza textos em inglês, optou-se por remover acentos e símbolos e deixá-los em formato *uppercase*. Estes procedimentos foram feitos nas funções '`normalizar_texto(texto)`' '`remover_acentos(texto)`'. Com os dados padronizados e os problemas com encoding resolvidos, foram desenvolvidas funções para carregar as informações cadastrais das empresas selecionadas e dos respectivos *tickers*, em seguida estes dados seriam unidos via 'merge' do pacote 'pandas' e usando 'query' para filtragem dos dados.

Devido ao grande volume de pastas, subpastas e arquivos, o mapeamento em lista dos caminhos dos diretórios (PATH) foram paralelizados utilizando a função '`ThreadPoolExecutor(max_workers=4)`' e a lista concatenada em um único *dataframe*.

Por fim, a função '`processar_arquivo()`' fazia o processo completo de ETL a partir dos arquivos mapeados e do REGEX definido para obter as informações necessárias.

A parte do ETL dos betas futuros e os cálculos de indicadores foi realizado na linguagem R utilizando as

seguintes bibliotecas: `dplyr`, `quantmod`, `readr`, `scales`, `slider`, `tidyr`, `xts`. O conjunto destas operações está disponível no Anexo II.

A partir do csv gerado no ETL do Python, os dados totalizaram 26.642 linhas. Devido a estrutura do arquivo, os dados foram transpostos e reestruturados para a etapa do cálculo dos indicadores, de modo que cada conta contábil se tornasse uma coluna. Nesta etapa foi feita a transformação dos informes individuais e consolidados para o formato Trailing Twelve Months (TTM).
A partir de um conjunto de operações os seguintes indicadores foram calculados para estas variáveis TTM:

- EBIT: Lucro Bruto - Despesas operacionais;
- ROE: Lucro Líquido / Patrimônio Líquido;
- ROA: Lucro Líquido / Ativo Total.

Em outro *script* do R foi feito o ETL dos betas futuros a partir da biblioteca quantmod. A função '`obter_retornos(symbol, start_date, end_date)`' extrai as variações de preço diárias de cada ativo, e também do índice BVSP. Enquanto a função '`calcular_beta(ret_ativo, ret_mercado)`' calcula o beta de determinado ativo. Estas duas funções foram inseridas em uma função maior '`calcular_betas_trimestrais(lista_tickers, start_date, end_date)`' esta percorre um *loop* '`for`' para cada ticker de uma lista para o período desejado, retornando os betas trimestrais. Estes em seguida foram transformados em betas futuros através de uma função de deslocamento que opera em cada ativo em dado período. Assim o beta futuro (variável explicada) poderia ser comparado com os indicadores contábeis do período vigente (variáveis explicativas).

5 Estatística Inferencial

Para análise estatística, foi preciso primeiro padronizar os dados. As informações contábeis e econômicas possuem escalas muito heterogêneas: ROE e ROA são indicadores em escala decimal, muitas vezes apresentados na forma de porcentagem; Lucro Bruto e Líquido podem chegar a milhões de reais positivos, ou negativos em caso de prejuízo; Beta é uma relação entre 2 variâncias, então possui valor próximo de uma unidade, com variações decimais. Para trabalhar com estes tipos de dados a opção foi pela padronização em três técnicas simultâneas: logarítmica, escalonamento robusto (*Robust Scale*) e Min-Max.
Transformações logarítmicas reduzem a assimetria, o que é extremamente relevante para dados com muitos *outliers*. Importante destacar aqui que a transformação utilizada foi a log1p(x), que calcula o logaritmo natural de (1+x) esta variável é útil para valores próximos de 0 a 1 em módulo, como é o caso do ROE, ROA e EBIT. Como estamos lidando com valores que podem ser negativos e também existem dados faltantes, foi feito o tratamento destes valores da seguinte forma: Se o número não estiver disponível (NA), ele retorna NA; Se o valor for maior que 0, opera log1p(x); Se for menor ou igual a zero (restante dos casos), mede o log1p(x) do valor absoluto, para em seguida transformar em valor negativo.

Desta forma, os valores negativos são transformados em módulo para em seguida voltarem a ser negativos.
Robust Scale é uma transformação que utiliza a mediana e o Intervalo Interquartil (IQR) para tornar os dados mais robustos a outliers. Enquanto o escalonamento min-max transforma todas as variáveis em uma escala de 0 a 1, evitando valores muito discrepantes e facilitando a comparação entre variáveis em escalas distintas. Estas transformações foram aplicadas a partir de funções no R e foram testadas tanto simultâneamente quanto de forma isolada.

6 Análise dos Resultados

Nesta etapa, a análise foi feita principalmente utilizando os resultados dos resumos do R que sintetizam o conjunto de informações relevantes do modelo.

Na Figura 1 é possível perceber que o conjunto das variáveis selecionadas explica 0,1453% do modelo. A princípio é um valor baixo para modelos que buscam predição de valores, mas para fins analíticos, como o objetivo é verificar o impacto de indicadores contábeis é um resultado razoavelmente bom. Pois o modelo não considera outras variáveis não contábeis que são importantes, como conjuntura macroeconômica, conjuntura política e análise setorial.

Figura 1 - *Output* do R, função `summary()` do modelo inicial com variáveis destacadas na literatura

```
> # Modelo Simples:
> df <- testes_rodar[["completo"]]  %>% select(-"TICKER", -"DT_REFER_ttm")
> modelo_beta_futuro_simples <- lm(Beta_futuro_mm ~ PATRIMONIO_LIQUIDO_TTM_mm + EBIT_mm
+                                  + LUCRO_LIQUIDO_TTM_mm + ROA_mm + ROE_mm,
+                                  data = df, na.action = na.omit)
> summary(modelo_beta_futuro_simples)

Call:
lm(formula = Beta_futuro_mm ~ PATRIMONIO_LIQUIDO_TTM_mm + EBIT_mm +
    LUCRO_LIQUIDO_TTM_mm + ROA_mm + ROE_mm, data = df, na.action = na.omit)

Residuals:
     Min       1Q   Median       3Q      Max
-0.59438 -0.05766  0.00204  0.06178  0.33925

Coefficients:
                       Estimate Std. Error t value             Pr(>|t|)
(Intercept)            1.037604   0.047946  21.641 < 0.0000000000000002 ***
PATRIMONIO_LIQUIDO_TTM_mm -0.035539   0.014775  -2.405             0.016257 *
EBIT_mm                -0.024433   0.006309  -3.873             0.000111 ***
LUCRO_LIQUIDO_TTM_mm   -0.087756   0.010567  -8.305 < 0.0000000000000002 ***
ROA_mm                 -0.175441   0.056013  -3.132             0.001763 **
ROE_mm                 -0.343253   0.073789  -4.652            0.00000352 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.09813 on 1855 degrees of freedom
  (172 observations deleted due to missingness)
Multiple R-squared:  0.1476,    Adjusted R-squared:  0.1453
F-statistic: 64.22 on 5 and 1855 DF,  p-value: < 0.00000000000000022
```

Ao analisar o modelo completo, foi possível perceber um erro na seleção de variáveis, o Passivo Total é igual ao Ativo Total, o que é normal em fechamento de balanços. Esta variável foi descartada, inicialmente buscava o Passivo resultante da soma do Passivo

Circulante e Não Circulante. Utilizado na construção de outros indicadores, mas neste caso não foi utilizado. Com o conjunto restante, o modelo foi mais explicativo que o anterior, com um R Quadrado Ajustado de 0,1594 (Figura 2). Além disso, é interessante observar que o comportamento dos resíduos também é melhorado, afinal, tanto a mediana quanto as extremidades dos resíduos é reduzida. Indicando, novamente, que os indicadores contábeis possuem poder explicativo sobre o risco sistemático.

Figura 2 - Analise completa das variáveis

```
> mod_teste_completo <- lm(Beta_futuro_mm ~ ., data = df, na.action = na.omit)
> summary(mod_teste_completo)

Call:
lm(formula = Beta_futuro_mm ~ ., data = df, na.action = na.omit)

Residuals:
     Min       1Q   Median       3Q      Max
-0.59210 -0.05898  0.00108  0.06258  0.32786

Coefficients:
                              Estimate Std. Error t value             Pr(>|t|)
(Intercept)                    0.97456    0.05203  18.732 < 0.0000000000000002 ***
ATIVO_TOTAL_TTM_mm            -0.01601    0.01709  -0.937              0.349059
DESPESAS_OPERACIONAIS_TTM_mm -0.03347    0.01112  -3.009              0.002657 **
EBIT_mm                       -0.06684    0.01363  -4.904           0.00000103 ***
LUCRO_BRUTO_TTM_mm             0.08244    0.04720   1.747              0.080885 .
LUCRO_LIQUIDO_TTM_mm          -0.10823    0.01190  -9.092 < 0.0000000000000002 ***
PATRIMONIO_LIQUIDO_TTM_mm     -0.03527    0.01730  -2.039              0.041649 *
RECEITA_BRUTA_TTM_mm          -0.04689    0.03707  -1.265              0.206147
ROA_mm                        -0.10365    0.05971  -1.736              0.082735 .
ROE_mm                        -0.26742    0.07673  -3.485              0.000504 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1 on 1669 degrees of freedom
  (354 observations deleted due to missingness)
Multiple R-squared:  0.1639,    Adjusted R-squared:  0.1594
F-statistic: 36.36 on 9 and 1669 DF,  p-value: < 0.00000000000000022
```

Para testar a viabilidade das padronizações, foram feitos os testes com diferentes combinações de transformações.

Apenas utilizando a transformação logarítmica (Figura 3), verificou-se que a aplicação subsequente de escalonamentos não alterou significativamente a significância estatística das variáveis, os valores-p ou o R^2 ajustado. A principal diferença observada foi na magnitude dos resíduos, resultado esperado pela alteração da escala da variável resposta.

Figura 3 - Resultados da transformação logarítmica isolada.

```
> summary(mod_teste_log)

Call:
lm(formula = formula_beta_futuro, data = df, na.action = na.omit)

Residuals:
     Min       1Q   Median       3Q      Max
-1.38810 -0.13828  0.00252  0.14672  0.76862

Coefficients:
                                Estimate Std. Error t value             Pr(>|t|)
(Intercept)                    0.8842442  0.0792443  11.158 < 0.0000000000000002 ***
ATIVO_TOTAL_TTM_log           -0.0045321  0.0048384  -0.937              0.349059
DESPESAS_OPERACIONAIS_TTM_log -0.0021424  0.0007119  -3.009              0.002657 **
EBIT_log                      -0.0040626  0.0008284  -4.904           0.00000103 ***
LUCRO_BRUTO_TTM_log            0.0052022  0.0029784   1.747              0.080885 .
LUCRO_LIQUIDO_TTM_log         -0.0067446  0.0007418  -9.092 < 0.0000000000000002 ***
PATRIMONIO_LIQUIDO_TTM_log    -0.0020894  0.0010249  -2.039              0.041649 *
RECEITA_BRUTA_TTM_log         -0.0036498  0.0028859  -1.265              0.206147
ROA_log                       -0.0929352  0.0535318  -1.736              0.082735 .
ROE_log                       -0.1128150  0.0323690  -3.485              0.000504 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2345 on 1669 degrees of freedom
  (354 observations deleted due to missingness)
Multiple R-squared:  0.1639,    Adjusted R-squared:  0.1594
F-statistic: 36.36 on 9 and 1669 DF,  p-value: < 0.00000000000000022
```

Porém, destacam-se os testes com apenas as transformações *Robust Scale* (Figura 4) e com apenas a Min-Max (Figura 5). Estes resultaram em um modelo inferior aos demais. Em ambos os casos houve redução do R^2 Ajustado para valores próximos de 0,072. Além disso, em ambas as transformações, o lucro líquido se tornou uma variável com baixa significância. Estas observações podem indicar a influência de dados dispersos com muitos *outliers* o que é comum nos dados de ações e empresas listadas na bolsa de valores. Mesmo que o *Robust Scaler* seja menos sensível a *outliers*, apenas este método não é suficiente dada a natureza dos dados estudados.

Figura 4 - Resultados da transformação *Robust Scale* isolada.

```
> # Teste apenas transformação robust_scale
> mod_teste_robust <- rodar_modelos_beta(testes_rodar[["robust_scale"]], "_robust")
> summary(mod_teste_robust)

Call:
lm(formula = formula_beta_futuro, data = df, na.action = na.omit)

Residuals:
    Min      1Q  Median      3Q     Max
-3.6777 -0.5930 -0.0705  0.4561  5.0940

Coefficients: (1 not defined because of singularities)
                                 Estimate Std. Error t value      Pr(>|t|)
(Intercept)                      0.141995   0.024284   5.847 0.00000000600 ***
ATIVO_TOTAL_TTM_robust           0.119850   0.032693   3.666      0.000254 ***
DESPESAS_OPERACIONAIS_TTM_robust -0.009011   0.040060  -0.225      0.822046
EBIT_robust                      -0.001836   0.058336  -0.031      0.974899
LUCRO_BRUTO_TTM_robust                 NA         NA      NA            NA
LUCRO_LIQUIDO_TTM_robust          0.002434   0.033579   0.072      0.942228
PATRIMONIO_LIQUIDO_TTM_robust    -0.160458   0.031718  -5.059 0.00000046835 ***
RECEITA_BRUTA_TTM_robust         -0.018780   0.022707  -0.827      0.408324
ROA_robust                       -0.029110   0.004912  -5.927 0.00000000374 ***
ROE_robust                       -0.020659   0.003894  -5.305 0.00000012747 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.8897 on 1670 degrees of freedom
  (354 observations deleted due to missingness)
Multiple R-squared:  0.07619,   Adjusted R-squared:  0.07176
F-statistic: 17.22 on 8 and 1670 DF,  p-value: < 0.00000000000000022
```

Figura 5 - Resultados da transformação Min-Max isolada.

```
> # Teste apenas transformação min-max
> mod_teste_mm <- rodar_modelos_beta(testes_rodar[["mm"]], "_mm")
> summary(mod_teste_mm)

Call:
lm(formula = formula_beta_futuro, data = df, na.action = na.omit)

Residuals:
     Min       1Q   Median       3Q      Max
-0.41765 -0.06735 -0.00801  0.05180  0.57849

Coefficients: (1 not defined because of singularities)
                              Estimate Std. Error t value             Pr(>|t|)
(Intercept)                    1.20492    0.12326   9.776 < 0.0000000000000002 ***
ATIVO_TOTAL_TTM_mm             0.55559    0.15155   3.666              0.000254 ***
DESPESAS_OPERACIONAIS_TTM_mm -0.02956    0.13140  -0.225              0.822046
EBIT_mm                       -0.00934    0.29680  -0.031              0.974899
LUCRO_BRUTO_TTM_mm                 NA         NA      NA                    NA
LUCRO_LIQUIDO_TTM_mm           0.01549    0.21375   0.072              0.942228
PATRIMONIO_LIQUIDO_TTM_mm     -0.45343    0.08963  -5.059        0.00000046835 ***
RECEITA_BRUTA_TTM_mm          -0.07378    0.08920  -0.827              0.408324
ROA_mm                        -0.38883    0.06560  -5.927        0.00000000374 ***
ROE_mm                        -0.62371    0.11756  -5.305        0.00000012747 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.101 on 1670 degrees of freedom
  (354 observations deleted due to missingness)
Multiple R-squared:  0.07619,   Adjusted R-squared:  0.07176
F-statistic: 17.22 on 8 and 1670 DF,  p-value: < 0.00000000000000022
```

Outros testes foram feitos selecionando as variáveis através dos métodos *step-wise.* Nestes utilizou-se tanto o método *Akaike Information Criterion* (AIC) quanto o *Bayesian Information Criterion* (BIC), respectivamente nas figuras 6 e 7, estes testam uma seleção com diferentes critérios de penalidade sobre o número de variáveis.

Figura 6 - Teste *step-wise both* (método AIC)

```
Step:  AIC=-7723.8
Beta_futuro_mm ~ DESPESAS_OPERACIONAIS_TTM_mm + EBIT_mm + LUCRO_LIQUIDO_TTM_mm +
    PATRIMONIO_LIQUIDO_TTM_mm + ROA_mm + ROE_mm

                              Df Sum of Sq    RSS     AIC
<none>                                      16.733 -7723.8
+ LUCRO_BRUTO_TTM_mm            1   0.01353 16.719 -7723.2
- ROA_mm                        1   0.02640 16.759 -7723.1
+ ATIVO_TOTAL_TTM_mm           1   0.00503 16.728 -7722.3
+ RECEITA_BRUTA_TTM_mm         1   0.00234 16.731 -7722.0
- PATRIMONIO_LIQUIDO_TTM_mm    1   0.06375 16.797 -7719.4
- DESPESAS_OPERACIONAIS_TTM_mm 1   0.09027 16.823 -7716.8
- ROE_mm                       1   0.12657 16.860 -7713.1
- EBIT_mm                      1   0.27171 17.005 -7698.8
- LUCRO_LIQUIDO_TTM_mm         1   0.80008 17.533 -7647.4

Call:
lm(formula = Beta_futuro_mm ~ DESPESAS_OPERACIONAIS_TTM_mm +
    EBIT_mm + LUCRO_LIQUIDO_TTM_mm + PATRIMONIO_LIQUIDO_TTM_mm +
    ROA_mm + ROE_mm, data = df_limpo)

Residuals:
     Min       1Q   Median       3Q      Max
-0.59282 -0.05922  0.00047  0.06224  0.33080

Coefficients:
                             Estimate Std. Error t value             Pr(>|t|)
(Intercept)                   0.98877    0.05127  19.287 < 0.0000000000000002 ***
DESPESAS_OPERACIONAIS_TTM_mm -0.03331    0.01109  -3.003              0.002710 **
EBIT_mm                      -0.05971    0.01146  -5.211          0.000000212 ***
LUCRO_LIQUIDO_TTM_mm         -0.10334    0.01156  -8.941 < 0.0000000000000002 ***
PATRIMONIO_LIQUIDO_TTM_mm    -0.03861    0.01530  -2.524              0.011700 *
ROA_mm                       -0.09647    0.05939  -1.624              0.104530
ROE_mm                       -0.27236    0.07659  -3.556              0.000387 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1 on 1672 degrees of freedom
Multiple R-squared:  0.1619,    Adjusted R-squared:  0.1589
F-statistic: 53.83 on 6 and 1672 DF,  p-value: < 0.00000000000000022
```

Figura 7 - Teste *step-wise both* (método BIC)

```
Step:  AIC=-7689.45
Beta_futuro_mm ~ DESPESAS_OPERACIONAIS_TTM_mm + EBIT_mm + LUCRO_LIQUIDO_TTM_mm +
    PATRIMONIO_LIQUIDO_TTM_mm + ROE_mm

                              Df Sum of Sq    RSS     AIC
<none>                                      16.759 -7689.4
- DESPESAS_OPERACIONAIS_TTM_mm  1   0.09323 16.853 -7687.7
- PATRIMONIO_LIQUIDO_TTM_mm     1   0.11266 16.872 -7685.8
+ ROA_mm                        1   0.02640 16.733 -7684.5
+ LUCRO_BRUTO_TTM_mm            1   0.01248 16.747 -7683.1
+ ATIVO_TOTAL_TTM_mm            1   0.00313 16.756 -7682.1
+ RECEITA_BRUTA_TTM_mm          1   0.00193 16.757 -7682.0
- ROE_mm                        1   0.15677 16.916 -7681.4
- EBIT_mm                       1   0.25783 17.017 -7671.4
- LUCRO_LIQUIDO_TTM_mm          1   0.89935 17.659 -7609.3

Call:
lm(formula = Beta_futuro_mm ~ DESPESAS_OPERACIONAIS_TTM_mm +
    EBIT_mm + LUCRO_LIQUIDO_TTM_mm + PATRIMONIO_LIQUIDO_TTM_mm +
    ROE_mm, data = df_limpo)

Residuals:
     Min       1Q   Median       3Q      Max
-0.59197 -0.05896  0.00039  0.06242  0.32918

Coefficients:
                             Estimate Std. Error t value             Pr(>|t|)
(Intercept)                   0.94194    0.04241  22.211 < 0.0000000000000002 ***
DESPESAS_OPERACIONAIS_TTM_mm -0.03384    0.01109  -3.051              0.002319 **
EBIT_mm                      -0.05788    0.01141  -5.073          0.000000435 ***
LUCRO_LIQUIDO_TTM_mm         -0.10721    0.01131  -9.475 < 0.0000000000000002 ***
PATRIMONIO_LIQUIDO_TTM_mm    -0.04774    0.01424  -3.354              0.000816 ***
ROE_mm                       -0.29707    0.07509  -3.956          0.000079404 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1001 on 1673 degrees of freedom
Multiple R-squared:  0.1606,    Adjusted R-squared:  0.1581
F-statistic:    64 on 5 and 1673 DF,  p-value: < 0.00000000000000022
```

Foi possível perceber que a retirada o Lucro Líquido é a última das alterações possíveis em ambos os métodos de seleção, indicando que é uma variável importante no modelo. Nestes testes *step-wise* houve pouca redução do R^2 Ajustado e houve aumento da significância das variáveis selecionadas. Além disso a mediana dos resíduos ficou mais próxima de zero, indicando que houve melhora no modelo ao selecionar as variáveis mais relevantes. Porém um dos custos desta redução é um aumento modesto das extremidades dos resíduos, sugerindo a necessidade de variáveis adicionais para melhorar a acurácia se o objetivo for a predição.

Algumas variáveis se destacaram em todos os modelos, como o caso do Lucro Líquido que foi extremamente significativo em todos os testes. Também se destacam o EBIT e ROE que são indicadores comuns na análise de ações. Aqui é preciso tomar cuidado, pois com base na teoria, muito do risco sistemático está atribuído ao comportamento coletivo dos investidores, logo, se muitos consideram tal variável como importante, ela pode gerar o efeito das expectativas auto realizáveis. O Lucro Líquido é uma variável que é pouco analisada isoladamente em análises mais robustas, então o fato dele sozinho ser a variável extremamente significativa pode sinalizar este fenômeno, embora esta confirmação precisasse de uma análise mais detalhada.

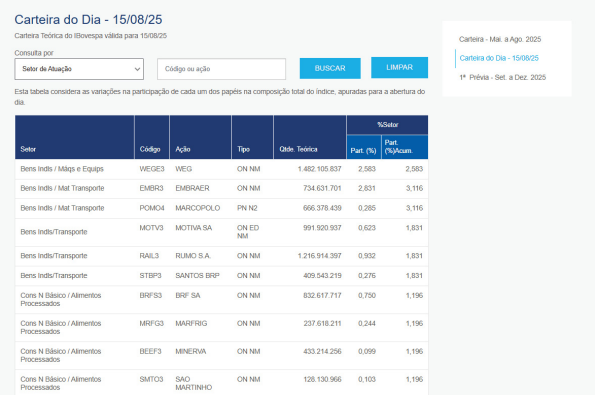
7 Conclusão

Os resultados mostram que determinados indicadores contábeis apresentam associação estatística significativa com o beta futuro na amostra analisada, com um R^2 ajustado em torno de 0,14–0,16, indicando poder explicativo moderado. Esses achados sugerem que informações contábeis fornecem sinais úteis sobre o comportamento do risco sistemático, embora expliquem apenas uma parcela da variação observada nos betas.

Porém, alguns diagnósticos de resíduos apontam heterogeneidade e presença de clusters, indicando subpopulações com padrões distintos. Isso limita a generalização dos resultados e sugere que efeitos podem variar conforme outras dimensões como setor, porte da empresa ou condições de mercado. Além disso, o desenho correlacional do estudo impede inferências causais, pois variáveis importantes foram excluídas do modelo com o objetivo de isolar os padrões contábeis. Variáveis macroeconômicas, setoriais ou de liquidez, não incluídas neste modelo, podem explicar parte das relações observadas.

A ponderação das ações no índice BVSP, calculada com base no volume negociado (B3, 2025a), é um fator relevante para pesquisas futuras, uma vez que o beta é estimado em relação a esse índice, utilizado como *proxy* do mercado como um todo. Conforme a metodologia apresentada pela B3 (2025a), o índice é uma carteira com as ações que resultam em cerca de 80% do número de negócios e do volume financeiro nesse mercado de capitais, além de alguns critérios de seleção adicionais. Esta metodologia revela que determinados setores da economia e determinadas empresas possuem pesos distintos na negociação (Figura 8), o que pode indicar variáveis idiossincráticas não contabilizadas no modelo estudado neste artigo.

Figura 8 - Exemplo de composição da Carteira do Dia 15/08/2025



Setor	Código	Ação	Tipo	Qtde. Teórica	%Setor	
					Part. (%)	Part. (%)Acum.
Bens Indls / Máqs e Equips	WEGE3	WEG	ON NM	1.482.105.837	2,583	2,583
Bens Indls / Mat Transporte	EMBR3	EMBRAER	ON NM	734.631.701	2,831	3,116
Bens Indls / Mat Transporte	POMO4	MARCOPOLO	PN N2	666.378.439	0,285	3,116
Bens Indls/Transporte	MOTV3	MOTIVA SA	ON ED NM	991.920.937	0,623	1,831
Bens Indls/Transporte	RAIL3	RUMO S.A.	ON NM	1.216.914.397	0,932	1,831
Bens Indls/Transporte	STBP3	SANTOS BRP	ON NM	409.543.219	0,276	1,831
Cons N Básico / Alimentos Processados	BRFS3	BRF SA	ON NM	832.617.717	0,750	1,196
Cons N Básico / Alimentos Processados	MRFG3	MARFRIG	ON NM	237.618.211	0,244	1,196
Cons N Básico / Alimentos Processados	BEEF3	MINERVA	ON NM	433.214.256	0,099	1,196
Cons N Básico / Alimentos Processados	SMTO3	SAO MARTINHO	ON NM	128.130.966	0,103	1,196

Fonte: B3, 2025b.

Esta própria distribuição ponderada revela inclusive mais um elemento: o risco não sistemático não deve ser ignorado em análises práticas. Afinal, mesmo com a diversificação, o peso relativo, favoritismos e variáveis idiossincráticas podem influenciar esse risco. É comum que, em documentos/exames da ANBIMA (2025), se enfatize que o risco nunca é excluído. Mesmo que possa ser reduzido por meio de técnicas, recomenda-se que o profissional sempre alerte potenciais clientes de que o risco nunca é eliminado, apenas mitigado. Por possuir um enfoque mais prático e profissional, esta abordagem difere do conceito amplamente destacado na literatura acadêmica que considera apenas o risco sistemático. Como foi detalhado anteriormente, esta é uma limitação importante considerada neste estudo: a exclusão de outras variáveis importantes para fins de recorte analítico.

Pesquisas subsequentes podem ampliar o recorte incorporando medidas de liquidez, variáveis macroeconômicas, análise setorial, *clusters* e análises estratificadas para avaliar a robustez e heterogeneidade dos efeitos aqui observados.

Por isso, a maior contribuição desta análise consiste em destacar as ferramentas disponíveis e suas limitações na análise e gestão do risco em carteiras diversificadas no mercado de ações. As variáveis contábeis mostraram-se estatisticamente significativas, mesmo com limitações no poder explicativo. O uso de ferramentas *Open Source* como o R e o Python, com suas respectivas bibliotecas, mas também a base de dados públicos disponível no Portal Dados Abertos CVM, reforçam não apenas a replicabilidade desta pesquisa, mas também podem estimular potenciais investidores a buscar informações confiáveis que auxiliem em suas tomadas de decisão.

Referências

AMBIMA, **CURSO DE ATUALIZAÇÃO CPA-20**. Documento interno. 2025. Disponível em: <https://www.anbima.com.br/pt_br/educar/cursos/atualizacoes-de-certificacao/atualizacao-cpa-20.htm>
Acesso em: 15/08/2025.

AMORIM, A. L. G. C.; LIMA, I. S.; MURCIA, F. Dal-Ri. **Análise da Relação entre as Informações Contábeis e o Risco Sistemático no Mercado Brasileiro**. 2011 R Cont Fin — USP, São Paulo.

B3 S.A. BRASIL, BOLSA, BALCAO. **Índice Bovespa (Ibovespa B3)**. 2025a.
<https://www.b3.com.br/pt_br/market-data-e-indices/indices/indices-amplos/ibovespa-b3.htm>
Acesso em : 13/08/2025.

B3 S.A. BRASIL, BOLSA, BALCAO. **Índice Bovespa (Composição da carteira)**.
2025b.<https://www.b3.com.br/pt_br/market-data-e-indices/indices/indices-amplos/ibovespa-b3.htm>
Acesso em : 13/08/2025.

COMISSÃO DE VALORES MOBILIÁRIOS (CVM). **Cias Abertas: Documentos: Formulário de Demonstrações Financeiras Padronizadas (DFP)**.
<https://dados.cvm.gov.br/dataset/cia_aberta-doc-dfp>
Acesso em: 13/08/2025.

COMISSÃO DE VALORES MOBILIÁRIOS (CVM). **Cias Abertas: Documentos: Formulário de Informações Trimestrais (ITR)**.
<https://dados.cvm.gov.br/dataset/cia_aberta-doc-itr>
Acesso em: 13/08/2025.

COMISSÃO DE VALORES MOBILIÁRIOS (CVM) **Cias Abertas: Informação Cadastral**.
<https://dados.cvm.gov.br/dataset/cia_aberta-cad>
Acesso em: 13/08/2025.

COMISSÃO DE VALORES MOBILIÁRIOS (CVM). **CVMWeb Central de Sistemas.**
<cvmweb.cvm.gov.br/swb/sistemas/scw/cpublica/ciaab/formbuscaciaabordalf.aspx>
Acesso em 13/08/2025.

KOBORI, José. **Análise fundamentalista; como obter uma performance superior e consistente no mercado de ações**. Rio de Janeiro Elsevier, 2012.

RECEITA FEDERAL DO BRASIL. **Sistema Público de Escrituração Digital (Sped): Tabelas Dinâmicas e Planos de Contas Referenciais - Leiaute 11 (Atualização: 11/07/2025)**.
<http://sped.rfb.gov.br/arquivo/show/7627>
Acesso em: 13/08/2025.

ROSS, STEPHEN A, WESTERFIELD, RANDOLPH W., JORDAN, BRADFORD D., LAMB, ROBERTO. **Fundamentos de administração financeira [tradução: Leonardo Zilio, Rafaela Guimarães Barbosa]**. 9. ed — Dados eletrônicos. — Porto Alegre: AMGH, 2013.

RYAN, J. A.; ULRICH, J. M.; SMITH, E. B.; THIELEN, W.; TEETOR, P.; BRONDER, S. **quantmod: Quantitative Financial Modelling Framework. Versão 0.4.28. [S.l.]**. 2025. Pacote do R. Disponível em: <https://cran.r-project.org/web/packages/quantmod/quantmod.pdf> Acesso em: 13/08/2025.

MATERIAL SUPLEMENTAR

Anexo I - *Script* em Python

```
##################################################################################################
######## MEGA SCRIPT DEEPSEEK + GPT- usar com cuidado e verificar várias vezes depois ############
##################################################################################################
######## MEGA SCRIPT DEEPSEEK + GPT usar com cuidado e verificar várias vezes depois ############
##################################################################################################
###################### Feito para speedrun de TCC perto do prazo lol ##############################
##################################################################################################
#############                                                                    #############
############# Soma tudo que foi discutido ao longo dos dias 16/06/2025 a 01/07/2025 #############
#############                                                                    #############
##################################################################################################
############# ETL Dados CVM - Processamento de Arquivos ITR / Iago.HC ############################
##################################################################################################



'''
python


dados cadastrais das empresas
https://dados.cvm.gov.br/dataset/cia_aberta-cad


Códigos da CVM (aqui fui buscar na força bruta, ctrl+a + ctrl+c + alt+tab + ctrl+v + alt+tab lol)
https://cvmweb.cvm.gov.br/swb/sistemas/scw/cpublica/ciaab/FormBuscaCiaAbOrdAlf.aspx?LetraInicial=A


Base de dados das Informações Trimestrais (ITR)
https://dados.cvm.gov.br/dataset/cia_aberta-doc-itr


Base de dados das Demonstrações Financeiras Padronizadas (DFP)
https://dados.cvm.gov.br/dataset/cia_aberta-doc-dfp


Leiaute da numeração utilizada na numeração das contas contábeis ('CD_CONTA' do .csv)
http://sped.rfb.gov.br/arquivo/show/5973


'''


##################################################################################################


# 1) Imports e configurações iniciais
from charset_normalizer import from_bytes
from concurrent.futures import ThreadPoolExecutor
from datetime import datetime
from functools import partial
import logging
import os
import pandas as pd
import re
import unicodedata
```

```
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
logger = logging.getLogger()
logger.info('Configurações iniciais...')
current_datetime = datetime.now().strftime("%Y%m%d_%H%M")


BASE_DIR   = "base_dados_cvm"
OUTPUT_DIR = "dados_testes"
os.makedirs(OUTPUT_DIR, exist_ok=True)




# Lista de arquivos
anos    = [2018, 2019, 2020, 2021, 2022, 2023, 2024]
informes = ['BPA','BPP','DRE']
tipos   = ['ind','con'] # 'ind','con'
itr_dfp =  ['itr','dfp'] # Informes Trimestrais ou Demonstrativos Financeiros Padronizados
arquivos = []


for ano in anos:
    for info in informes:
        # ITR: apenas individual do trimestre (consolidado aqui não vale a pena usar)
        arquivos.append(f"{BASE_DIR}/ITRs/itr_cia_aberta_{ano}/itr_cia_aberta_{info}_ind_{ano}.csv")


        # DFP: individual e consolidado (consolidado puxa os dados do ano todo e é auditado! ind é apenas
verificado!)
        for tipo in tipos:
            arquivos.append(f"{BASE_DIR}/DFPs/dfp_cia_aberta_{ano}/dfp_cia_aberta_{info}_{tipo}_{ano}.csv")




DTYPE_MAP = {col: str for col in ['CNPJ_CIA','CD_CVM','CD_CONTA', 'DS_CONTA', 'ORDEM_EXERC']} # Lembrar de
capturar apenas o 'ORDEM_EXERC'== 'ULTIMO' após normalizar
DTYPE_MAP['VL_CONTA'] = float


# 2) Mapeamentos e palavras-chave
MAPEAMENTO_CONTAS = {
    'BPA': {
        '1': ('1', 'ATIVO_TOTAL', r'^ATIVO TOTAL$'),
        # '1.01': ('1.01', 'ATIVO_CIRCULANTE', r'^ATIVO CIRCULANTE$'), # Não utilizado
        # '1.02': ('1.02', 'ATIVO_NAO_CIRCULANTE', r'(ATIVO NAO CIRCULANTE$|ATIVO REALIZAVEL A LONGO
PRAZO)') # Não utilizado
    },
    'BPP': {
        '2': ('2', 'PASSIVO_TOTAL', r'^PASSIVO TOTAL$'),
        # '2.01': ('2.01', 'PASSIVO_CIRCULANTE', r'^PASSIVO CIRCULANTE$'), # Não utilizado
        # '2.02': ('2.02', 'PASSIVO_NAO_CIRCULANTE', r'^PASSIV.*N.*CIRCULANTE'), # Não utilizado
        '2.03': ('2.03', 'PATRIMONIO_LIQUIDO_ITR',
r'^PATRIM.*L.*QUIDO(?:\s+(?:TOTAL|CONSOLIDADO|DO\s+PERÍODO))?$')
    },
    'DRE': {
        '3.01': ('3.01', 'RECEITA_BRUTA', r'\b(?:RECEITA|VENDA|FORNECIMENTO)\b(?:.*\bBRUT[AO]?\b)?'),
        '3.03': ('3.03', 'LUCRO_BRUTO', r'^(RESULTADO|LUCRO).*BRUTO'),
        '3.04': ('3.04', 'DESPESAS_OPERACIONAIS',
r'(?!.*(?:EQUIVALEN|PATRIMON)).*?DESPESA.*?(?:ADMIN.*|GERA.*|OPERACION.*).*?'),
        '3.11': ('3.11', 'LUCRO_LIQUIDO',
r'^(?:LUCRO.*|PREJU.*).*?(?:.*QUIDO|CONSOLIDAD.*).*(?:PERI.*|EXERC.*).*'),
```

```
        '3.99': ('3.99', 'LPA_CVM', r'LUCRO*.POR*.ACAO')
    }
}



LEVELS_DRE_3_01 = [
    r'^3\.01$',            # Conta principal 3.01
    r'^3\.\d{2}\.\d{2}$',     # Subcontas 3.01.XX
    r'^3\.\d{2}\.\d{2}\.\d+'  # Subcontas detalhadas
]


logger.info(f"{LEVELS_DRE_3_01}")
logger.info("até aqui está ok! só mexer nos dicionários se precisar")
```

```
# 3.1) Funções utilitárias que estão funcionando ok! Não mexer a não ser que realmente precise!!!
logger.info('Definindo as funções...')
# Detectar encoding e setar para 'cp1252' se for necessário
def detectar_encoding(path, sample_size=9000000):
    with open(path, 'rb') as f:
        raw = f.read(sample_size)
    best = from_bytes(raw).first()


    # Se encoding não for utf-8 ou cp1252, forçar cp1252 como padrão
    if best and best.encoding.lower() in ['utf_8', 'cp1252', 'iso-8859-1']:
        logger.info(f"Encoding detectado: {best.encoding} | Idioma: {best.language}")
        return best.encoding
    else:
        logger.warning(f"Encoding incomum detectado ({best.encoding if best else 'None'}), forçando
cp1252.")
        return 'cp1252'



# Aqui remove acentos
def remover_acentos(texto):
    """Remove acentos mantendo a string original"""
    if not isinstance(texto, str):
        return texto
    nfkd_form = unicodedata.normalize('NFKD', texto)
    return ''.join([c for c in nfkd_form if not unicodedata.combining(c)])



# Aqui usam as funções anteriores para normalizar o texto
def normalizar_texto(texto):
    """Normaliza texto para análise"""
    texto = str(texto)
    texto = remover_acentos(texto)
    texto = texto.upper().strip()
    return texto
```

```
logger.info("Concluído esta parte das funções =)")
```

```
# TEMPORARIO

path =
detectar_encoding
df = pd.read_csv()
df['DS_CONTA'] = df['DS_CONTA'].map(normalizar_texto)
```

```
# 3.2) Funções que mexem com mapeamento, aqui tomar cuidado e verificar!!!!
logger.info('Definindo as funções de mapeamento...')


def buscar_mapeamento(df, tipo):
    """Retorna DF apenas com linhas que casam algum pattern do seu MAPEAMENTO_CONTAS."""
    resultados, unmapped_idx = [], set(df.index)


    # Contas (BPA, BPP, DRE exceto 3.01)
    for tipo_map, entries in MAPEAMENTO_CONTAS.items():
        if tipo_map != tipo:
            continue

        for _, (cd, ds_norm, rx) in entries.items():
            # pula o DRE 3.01 nesta fase
            if tipo == 'DRE' and cd == '3.01':
                continue


            pattern = re.compile(rx, re.IGNORECASE)
            if tipo == 'DRE' and cd in {'3.04', '3.11'}:
                try:
                    # testa primeiro se a cd_conta tem o dígito certinho
                    logger.info(f"Fazendo busca dos 3.04 e 3.11...")
                    sel = df['CD_CONTA'].str.contains(r'^(?:3\.04$|3\.11$)', regex=True, na=False) & \
                    df['DS_CONTA'].str.contains(pattern, regex=True, na=False)
                except:
                    # se não, ignora completamente CD_CONTA, filtra apenas por DS_CONTA
                    logger.info(f"Não deu, tentar na força bruta lol...")
                    sel = df['DS_CONTA'].str.contains(pattern, regex=True, na=False)
            else:
                # filtro padrão (usa índice ainda não mapeado)
                sel = df.index.isin(unmapped_idx) & \
                    df['DS_CONTA'].str.contains(pattern, regex=True, na=False)


            if sel.any():
                temp = df[sel].copy()
                temp[['CD_NORM','DS_NORM']] = cd, ds_norm
                resultados.append(temp)
                unmapped_idx -= set(temp.index)


    # DRE: receita bruta em múltiplos níveis
    if tipo == 'DRE':
        cd, ds_norm, rx = MAPEAMENTO_CONTAS['DRE']['3.01']
        pattern_bruta = re.compile(rx, re.IGNORECASE)
```

```
        pattern_geral = re.compile(r'^(?:RECEITAS.*|RECEITA.*|VENDA.*|FORNECIMENTO.*)', re.IGNORECASE)
        # print(f"teste {LEVELS_DRE_3_01}")
        for lvl_regex in LEVELS_DRE_3_01:
            nivel = df['CD_CONTA'].str.match(lvl_regex, na=False)
            logger.info(f"Fazendo busca dos 3.01...")
            # Busca contas com "BRUTA" no nome
            temp_bruta = df[nivel & df['DS_CONTA'].str.contains(pattern_bruta, regex=True, na=False)]
            if not temp_bruta.empty:
                # ATRIBUA O MAPEAMENTO
                temp_bruta = temp_bruta.copy()
                temp_bruta['CD_NORM'] = cd
                temp_bruta['DS_NORM'] = ds_norm
                resultados.append(temp_bruta)
                unmapped_idx -= set(temp_bruta.index)
                break


            # Busca contas genéricas de receita
            logger.info(f"Não deu, tentar na força bruta lol...")
            temp_geral = df[df['DS_CONTA'].str.contains(pattern_geral, regex=True, na=False)]
            if not temp_geral.empty:
                temp_geral = temp_geral.copy()
                temp_geral['CD_NORM'] = cd
                temp_geral['DS_NORM'] = ds_norm
                resultados.append(temp_geral)
                unmapped_idx -= set(temp_geral.index)
                break


    # 3.2.3) concat e retorno
    if resultados:
        out = pd.concat(resultados, ignore_index=True)
    else:
        out = pd.DataFrame(columns=df.columns.tolist()+['CD_NORM','DS_NORM'])


    # opcional: juntar não mapeados se precisar
    # df_rest = df.loc[unmapped_idx].assign(CD_NORM='ND', DS_NORM='ND')
    # out = pd.concat([out, df_rest], ignore_index=True)


    return out.reindex(columns=df.columns.tolist() + ['CD_NORM', 'DS_NORM'])


# 4) Processamento de arquivos
def processar_arquivo(informes, path: str) -> pd.DataFrame:
    basename = os.path.basename(path).lower()
    tipo = next(info for info in informes if f"_{info}_" in path)
    enc  = detectar_encoding(path)


    match = re.search(r'_(ind|con)_\d{4}\.csv', basename)
    if not match:
        logger.warning(f"Não foi possível determinar 'ind' ou 'con' em {basename}")
    ind_con = match.group(1)
    logger.info(f"Processando arquivo {basename}: \n"
                f"tipo '{tipo}_{ind_con}' \n"
                f"encoding {enc}")

    try:
        if not tipo == 'DRE':
            cols = ['CNPJ_CIA', 'CD_CVM', 'DT_REFER', 'DT_FIM_EXERC', 'ORDEM_EXERC', 'CD_CONTA',
'DS_CONTA', 'VL_CONTA']
        else:
```

```
            cols = ['CNPJ_CIA', 'CD_CVM', 'DT_REFER', 'DT_INI_EXERC', 'DT_FIM_EXERC', 'ORDEM_EXERC',
'CD_CONTA', 'DS_CONTA', 'VL_CONTA']


        df = pd.read_csv(
            path, sep=';', encoding=enc,
            usecols=cols,
            parse_dates=['DT_REFER'],
            dtype=DTYPE_MAP
        )
    except Exception as e:
        logger.error(f"Erro ao ler {basename}: {str(e)}")
        return pd.DataFrame()


    logger.info(f"Iniciando busca - {len(df)} linhas")
    df = df.query("CD_CVM in @codigos_cvm").copy()
    logger.info(f"Iniciando busca empresas listadas - {len(df)} linhas")


    df['INFO_IND_CON'] = ind_con
    df['CD_CONTA'] = df['CD_CONTA'].str.strip()
    df['DS_CONTA'] = df['DS_CONTA'].map(normalizar_texto)
    df['ORDEM_EXERC'] = df['ORDEM_EXERC'].map(normalizar_texto)


    ordens = df['ORDEM_EXERC'].unique().tolist()
    logger.info(f"Exercicios listados: {ordens}")
    df = df.query("ORDEM_EXERC == 'ULTIMO'").copy()




    logger.info(f"Iniciando busca mapeada {tipo} - {len(df)} linhas")
    df_filt = buscar_mapeamento(df, tipo)
    print(f"Contas encontradas: {len(df_filt)}")


    # Auditoria: verifica contas essenciais
    contas_essenciais = set(MAPEAMENTO_CONTAS[tipo].keys())
    contas_encontradas = set(df_filt['CD_NORM'])
    faltantes = contas_essenciais - contas_encontradas


    if faltantes:
        logger.warning(f"Contas essenciais faltantes em {basename}: {', '.join(faltantes)}")


    logger.info(f"Busca concluída '{tipo}_{ind_con}' Dataset {basename} - Linhas resultantes:
{len(df_filt)}")


    return df_filt


logger.info("Concluído esta parte das funções =)")
```

```
# 5) Carregar funções dos cadastros e mais filtros
def carregar_cadastro(path) -> pd.DataFrame:
    """Carrega e filtra o cadastro de empresas ativas"""
    logging.debug('abrindo arquivo de dados cadastrais: "cad_cia_aberta.csv"')
```

```
    enc  = detectar_encoding(path)
    logger.info(f"{os.path.basename(path)} → encoding {enc}")

    df = pd.read_csv(
        path,
        sep=';',
        encoding=enc,  # corrige bug de não ler arquivos aqui
        usecols=['CNPJ_CIA', 'CD_CVM', 'DT_REG', 'DENOM_SOCIAL', 'SETOR_ATIV', 'TP_MERC', 'SIT'],
        parse_dates=['DT_REG'],
        dtype={'CNPJ_CIA': str, 'CD_CVM': str}
    )

    df['DENOM_SOCIAL'] = df['DENOM_SOCIAL'].map(normalizar_texto)
    df['SETOR_ATIV'] = df['SETOR_ATIV'].map(normalizar_texto)
    logging.debug("Filtrando registros ativos (SIT == 'ATIVO')...")
    df_filtrado = df.query("SIT == 'ATIVO'").drop(columns=['SIT'])

    logging.debug("Filtrando empresas anteriores a 2016 (DT_REG < '2016-01-01')...")
    df_filtrado = df_filtrado.query("DT_REG < '2018-01-01'")

    logging.debug("Filtrando apenas empresas listadas na B3 (TP_MERC == 'BOLSA')...")
    df_filtrado = df_filtrado.query("TP_MERC == 'BOLSA'").drop(columns=['TP_MERC'])
    df_filtrado['CD_CVM'] = df_filtrado['CD_CVM'].astype(str).str.zfill(6)

    df_filtrado.to_csv('teste_cadastros.csv', index=False)
    logging.info(f'Cadastro carregado: {len(df_filtrado)} empresas ativas')

    return df_filtrado

# Etapa 4: Composição de capital (Não utilizado no final)
def carregar_quant_cap(cadastro, path):
    logging.info('Calc quantidades de ações...')
    # path = f'base_dados_cvm/itr_cia_aberta_{ano}/itr_cia_aberta_composicao_capital_{ano}.csv'
    enc  = detectar_encoding(path)
    logger.info(f"{os.path.basename(path)} → encoding {enc}")


    capital = pd.read_csv(
        path,
        sep=',',
        encoding=enc,  # corrige bug de não ler arquivos aqui
        usecols=['CNPJ_CIA', 'DT_REFER', 'QT_ACAO_TOTAL_CAP_INTEGR'],
        parse_dates=['DT_REFER'],
        dtype={'CNPJ_CIA': str}
    )

    logger.info(f"Iniciando busca quant ttl ações - {len(capital)} linhas")
    cadastro_reduzido = cadastro[['CNPJ_CIA', 'CD_CVM']]

    logger.info(f"Total de registros cadastro_reduzido - {len(cadastro_reduzido)} linhas")
    capital = capital.merge(cadastro_reduzido, on='CNPJ_CIA', how='left')
    capital = capital.query("CD_CVM in @codigos_cvm").copy()
    logger.info(f"Total de registros quant ttl ações - {len(capital)} linhas")
    capital.to_csv('teste_qnt_acoes.csv', index=False)
    n_missing = capital['CD_CVM'].isna().sum()

    if n_missing > 0:
        logger.warning(f"{n_missing} registros sem correspondência de CD_CVM.")

    return capital

# def executar_etl(df_filt):
# logging.info('Pivotando tabela...')
```

```
# Deixar pra pivotar no R!

logger.info("Funções ok!")
```

```
# Execução dos programas acima, aqui testar!!!
# Esta parte está rodando ok também, não mexer a não ser que seja extremamente necessário


# Lista de arquivos
# ano = 2024
# anos = [2024]
# informes= ['BPA','BPP','DRE']
# # informes= ['DRE']
# tipos   = ['ind','con']
# Comentado para fora, definido lá em cima
# arquivos = [
#     f"{BASE_DIR}/itr_cia_aberta_{ano}/itr_cia_aberta_{info}_{tipo}_{ano}.csv"
#     for ano in anos for info in informes for tipo in tipos
# ]
# Usar este para poucas empresas (para testar só)
# codigos_cvm=['001023', '009512', '007617', '002453', '020087']

# Usar este para várias empresas
logging.info('Carregar base com tickers...')
tickers_e_cvm= pd.read_csv('tickers_com_status_cvm.csv',
                        sep=",",
                        dtype={'CD_CVM': str, 'ATIVO_CVM': str, 'TICKER': str})
codigos_cvm = tickers_e_cvm['CD_CVM'].to_list()


# 6) Carregar cadastros
logging.info('Carregar dados básicos...')
current_datetime = datetime.now().strftime("%Y%m%d_%H%M")
cadastro = carregar_cadastro('base_dados_cvm/cad_cia_aberta.csv')

# print(cadastro.sample(n=5))

logging.info('Amostra do cadastro.')
cadastro = cadastro.query("CD_CVM in @codigos_cvm").copy()
# print(cadastro.sample(n=5))


# 6) Carregar cadastros e quantidades de ações
# cria função que já leva cadastro "pré-carregado"
func = partial(carregar_quant_cap, cadastro)

import os

# anos = [2018, 2019, 2020, 2021, 2022, 2023, 2024]
arquivos_capital = []
anos_ausentes = []

for ano in anos:
    encontrados = False

    # ITR
    itr_path = f"{BASE_DIR}/ITRs/itr_cia_aberta_{ano}/itr_cia_aberta_composicao_capital_{ano}.csv"
    if os.path.exists(itr_path):
        arquivos_capital.append(itr_path)
        encontrados = True

    # DFP
```

```
    dfp_path = f"{BASE_DIR}/DFPs/dfp_cia_aberta_{ano}/dfp_cia_aberta_composicao_capital_{ano}.csv"
    if os.path.exists(dfp_path):
        arquivos_capital.append(dfp_path)
        encontrados = True

    if not encontrados:
        anos_ausentes.append(ano)

if anos_ausentes:
    logger.info(f"Aviso: Arquivo de composição de capital não encontrado para os anos: {anos_ausentes}")


with ThreadPoolExecutor(max_workers=4) as executor:
    lista = list(executor.map(func, arquivos_capital))

# opcional: concatenar tudo num único DF
df_quant = pd.concat(lista, ignore_index=True)
logger.info(f"Arquivos carregados df_quant: {len(df_quant)}")
```

```
# 7) Leitura paralela, concat, dedup, sample e save BPA, BPP, DRE
logger.info('Início da paralelização e filtragens...')
func_contas = partial(processar_arquivo, informes)

with ThreadPoolExecutor(max_workers=4) as executor:
    lista = list(executor.map(func_contas, arquivos))



dfs_validos = [df for df in lista if isinstance(df, pd.DataFrame)]
df_total = pd.concat(dfs_validos, ignore_index=True)
logger.info(f"Arquivos carregados df_total: {len(df_total)}")
```

```
# JUNTANDO E SALVANDO TUDO
# # Remover duplicatas considerando CD_CONTA e TIPO
# df_total = df_total.drop_duplicates()
logger.info("Juntando e salvando tudo")

# CD_CVM deve ser string de 6 dígitos
df_total['CD_CVM'] = df_total['CD_CVM'].astype(str).str.zfill(6)
df_quant['CD_CVM'] = df_quant['CD_CVM'].astype(str).str.zfill(6)

# CNPJ deve ser string com zeros à esquerda
df_total['CNPJ_CIA'] = df_total['CNPJ_CIA'].astype(str).str.zfill(14)
df_quant['CNPJ_CIA'] = df_quant['CNPJ_CIA'].astype(str).str.zfill(14)

# Datas devem ter mesmo tipo -- o mais seguro é usar datetime
df_total['DT_REFER'] = df_total['DT_REFER'].astype(str)
df_quant['DT_REFER'] = df_quant['DT_REFER'].astype(str)


df_total = df_total.merge(df_quant, on=['CD_CVM', 'CNPJ_CIA', 'DT_REFER'], how='left')
periodo = "2021-2024"
# df_total.rename(columns = {'DT_REFER_x':'DT_REFER'})
# df_total.drop(columns=['DT_REFER_y'])
# Salvar resultado final
out_path = os.path.join(OUTPUT_DIR, f"Amostra_empresas_{periodo}_{current_datetime}.csv")
df_total.to_csv(out_path, index=False, sep=';', encoding='cp1252')
logger.info(f"Resultado salvo em {out_path} ({len(df_total)} linhas)")
```

```
# Separar arquivo pelos anos
```

```
# Colunas:
#
CNPJ_CIA;DT_REFER;CD_CVM;ORDEM_EXERC;DT_FIM_EXERC;CD_CONTA;DS_CONTA;VL_CONTA;INFO_IND_CON;CD_NORM;DS_NORM;D
T_INI_EXERC;QT_ACAO_TOTAL_CAP_INTEGR
print("iniciando separação dos anos")

OUTPUT_DIR = "dados_testes"
os.makedirs(OUTPUT_DIR, exist_ok=True)
path = 'dados_testes/Amostra_empresas_2021-2024_20250702_1144.csv'

# Data e hora atual para o nome do arquivo
current_datetime = datetime.now().strftime("%Y%m%d_%H%M")

# ler arquivo completo
df = pd.read_csv(
        path,
        sep=';',
        encoding='cp1252',
        usecols=['CNPJ_CIA', 'DT_REFER', 'CD_CVM', 'DT_INI_EXERC', 'DT_FIM_EXERC', 'CD_CONTA', 'DS_CONTA',
'VL_CONTA', 'INFO_IND_CON', 'CD_NORM', 'DS_NORM', 'QT_ACAO_TOTAL_CAP_INTEGR'],
        parse_dates=['DT_REFER', 'DT_INI_EXERC', 'DT_FIM_EXERC'],
        dtype={'CNPJ_CIA': str, 'CD_CVM': str, 'CD_CONTA': str, 'DS_CONTA': str, 'INFO_IND_CON': str,
'CD_NORM': str, 'DS_NORM': str}
)

# Extrair anos únicos de DT_REFER
series = pd.Series(df['DT_REFER'])
lista_anos = sorted(series.dt.year.unique().tolist())
print(f"Anos encontrados: {lista_anos}")

# Loop por ano
for ano in lista_anos:
    df_temp = df[df['DT_REFER'].dt.year == ano].copy()
    out_path = os.path.join(OUTPUT_DIR, f"Amostra_empresas_{ano}_{current_datetime}.csv")
    df_temp.to_csv(out_path, index=False, sep=';', encoding='cp1252')



print("concluído")
```

Anexo II - Script em R

```
############################################################
#####      O script a seguir faz o seguinte:         #####
#####      Torna os dados trimestrais em anuais      #####
###  puxando infos a partir dos últimos 12 meses (TTM) ###
###    também  calcula os indicadores e                ###
#####       faz a normalização e padronização,       #####
#####   utilizando log -> robustscaler -> minmax     #####
############################################################

# Se ainda não instalado #
# install.packages("quantmod")
# install.packages("xts")
# install.packages("devtools")
# install.packages("GetDFPData2")
# install.packages("slider")
# install.packages("scales")

# Carregar bibliotecas
{ library(dplyr)
  library(quantmod)  # Para getSymbols e cálculos financeiros
  library(readr)
  library(scales)
  library(slider)
  library(tidyr)
  library(xts)       # Para manipulação de séries temporais (opcional, ver depois se vale a pena)
}


options(scipen = 999)  # Desativa notação científica para números grandes

# Path PC Trabalho
# C:/[...]/6 - PosGrads/DSBD/TCC - DSBD/TCC
2025/TCC-enviar-prof/Amostra_empresas_2021-2024_20250702_1144.csv
# Path casa
# C:/-/Documents/AULAS_CURSOS/DSBD/TCC - DSBD/BASE_DADOS_TCC 02-07-2025/
# colocar o caminho do arquivo 'BASE_DADOS_TCC'
path <- "C:/-----/TCC - DSBD/BASE_DADOS_TCC 02-07-2025/"

# Define os tipos das colunas (como no DTYPE_MAP)
col_types <- cols(
  CNPJ_CIA = col_character(),
  CD_CVM = col_character(),
  CD_CONTA = col_character(),
  DS_CONTA = col_character(),
  VL_CONTA = col_double(),
  DT_REFER = col_date(format = "%Y-%m-%d"),  # Formato da data
  DT_FIM_EXERC = col_date(format = "%Y-%m-%d"),
  .default = col_guess()
)

df_com_tudo_junto <- read_delim(
  file.path(path, "df_com_tudo_junto.csv"), # 'BASE_DADOS_TCC'
  delim = ",",
  locale = locale(decimal_mark = ".", encoding = "latin1"),
  col_types = col_types
)


glimpse(df_com_tudo_junto)
colMeans(is.na(df_com_tudo_junto)) * 100  # percentual de NAs por coluna
```

```
# Aqui eu vi que dá para excluir as outras colunas com datas
# df_com_tudo_junto %>%
#   filter(DT_REFER != DT_FIM_EXERC) %>%
#   summarise(n = n())
# Deu zerado

##############################
# Pivotando pra criar indicadores


# --- Etapa 1: pivotar contas principais ---
df_wide <- df_completo %>%
  select(TICKER, INFO_IND_CON, DT_REFER, DS_NORM, VL_CONTA) %>%
  group_by(TICKER, DT_REFER, INFO_IND_CON, DS_NORM) %>%
  summarise(VL_CONTA = max(VL_CONTA, na.rm = TRUE), .groups = "drop") %>%
  pivot_wider(names_from = DS_NORM, values_from = VL_CONTA)

# 1) Veja quais tickers ficaram com NA em LUCRO_LIQUIDO
empresas_com_NA <- df_wide %>%
  filter(is.na(LUCRO_LIQUIDO)) %>%
  distinct(TICKER, DT_REFER, INFO_IND_CON)


# Constrói o caminho para salvar o arquivo
out_path <- file.path(path, paste0("empresas_com_NA.csv"))
write.csv(empresas_com_NA, out_path, row.names = FALSE)

colMeans(is.na(df_wide)) * 100  # percentual de NAs por coluna


glimpse(df_wide)
colnames(df_wide)[colnames(df_wide) == "PATRIMONIO_LIQUIDO_ITR"] <- "PATRIMONIO_LIQUIDO" # Corrige aqui um
erro na nomeação antiga
##############################
# QT_ACAO_TOTAL_CAP_INTEGR e
# ORGANIZANDO PARA O FORMATO TTM (Trailing Twelve Months) este para pegar infos anualizadas, reduzindo
impactos de sazonalidade

glimpse(df_wide)
colnames(df_wide)[colnames(df_wide) == "PATRIMONIO_LIQUIDO_ITR"] <- "PATRIMONIO_LIQUIDO" # Corrige aqui um
erro na nomeação antiga


df_tt0 <- df_wide %>%
  filter(DT_REFER >= as.Date("2018-10-01"))


df_ind <- df_tt0 %>% filter(INFO_IND_CON == "ind")
df_con <- df_tt0 %>% filter(INFO_IND_CON == "con")

df_ind_ttm <- df_ind %>%
  arrange(TICKER, DT_REFER) %>%
  group_by(TICKER) %>%
  mutate(
    across(
      where(is.numeric),
      ~ slide_dbl(.x, sum, .before = 3, .complete = TRUE),
      .names = "{.col}_TTM"
    )
  ) %>%
  ungroup()

df_ind_ttm <- df_ind_ttm %>%
  ungroup() %>%
```

```
  filter(
    rowSums(across(matches("_TTM$"), ~ !is.na(.))) > 0
  )


# renomeia colunas para manter padrão _TTM
df_con_ttm <- df_con %>%
  transmute(
    TICKER, DT_REFER, INFO_IND_CON,
    across(where(is.numeric), ~ ., .names = "{.col}_TTM")
  )



df_ttm_final <- bind_rows(df_ind_ttm, df_con_ttm) %>%
  arrange(TICKER, DT_REFER) %>%
  # select(-QT_ACAO_TOTAL_CAP_INTEGR_TTM) %>% # Quantidade de ações no dado período não se soma
  select(-RECEITA_BRUTA, -ATIVO_TOTAL, -DESPESAS_OPERACIONAIS,
         -LPA_CVM, -LUCRO_BRUTO, -LUCRO_LIQUIDO,
         -PASSIVO_TOTAL, -PATRIMONIO_LIQUIDO) # Aqui remove os indicadores que não vamos mais usar


glimpse(df_ttm_final)



df_indicadores <- df_ttm_final %>%
  mutate(
    # Cálculo do EBIT
    EBIT = LUCRO_BRUTO_TTM - DESPESAS_OPERACIONAIS_TTM,

    # Rentabilidade
    ROE = ifelse(
      PATRIMONIO_LIQUIDO_TTM > 0,
      LUCRO_LIQUIDO_TTM / PATRIMONIO_LIQUIDO_TTM,
      0
    )
    ,
    ROA = LUCRO_LIQUIDO_TTM / ATIVO_TOTAL_TTM,
    # LPA = LUCRO_LIQUIDO_TTM / QT_ACAO_TOTAL_CAP_INTEGR # não mais utilizado
    )

glimpse(df_indicadores)

# Normalizando os dados:
# Log, RobustScaler e MinMaxScaler em R

# vetor de colunas a padronizar
colunas <- c(
  "ATIVO_TOTAL_TTM", "DESPESAS_OPERACIONAIS_TTM", "LUCRO_BRUTO_TTM",
  "LUCRO_LIQUIDO_TTM", "PASSIVO_TOTAL_TTM", "PATRIMONIO_LIQUIDO_TTM",
  "RECEITA_BRUTA_TTM", "EBIT", "ROE", "ROA"
)

# Funções de transformação
log_transform <- function(x) {
  case_when(
    is.na(x)  ~ NA_real_, # aqui se for NA
    x > 0     ~ log1p(x),
    TRUE      ~ -log1p(abs(x)) # Aqui se for negativo ou 0
  )
}
```

```
robust_scale <- function(x) {
  med <- median(x, na.rm = TRUE)
  iqr <- IQR(x,   na.rm = TRUE)
  (x - med) / iqr
}

minmax_scale <- function(x) {
  # utiliza scales::rescale
  scales::rescale(x, to = c(0, 1))
}

# (1-3) Loop for aplicando as três etapas em cada coluna
for (col in colunas) {
  # nomes das novas colunas
  col_log   <- paste0(col, "_log")
  col_rob   <- paste0(col, "_robust")
  col_mm    <- paste0(col, "_mm")

  # 1) Log
  df_indicadores[[col_log]] <- log_transform(df_indicadores[[col]])

  # 2) RobustScaler
  df_indicadores[[col_rob]] <- robust_scale(df_indicadores[[col_log]])

  # 3) MinMax
  df_indicadores[[col_mm]]  <- minmax_scale(df_indicadores[[col_rob]])
}

# Salvar tudo até aqui
# Obtém data e hora no formato desejado
current_datetime <- format(Sys.time(), "%Y%m%d_%H%M")

# Constrói o caminho para salvar o arquivo
out_path <- file.path(path, paste0("df_indicadores_", current_datetime, ".csv"))
write.csv(df_indicadores, out_path, row.names = FALSE)
```

```
############################################################
#####      O script a seguir faz o seguinte:         #####
#####  Extrai as informações dos preços (Nosso Y)    #####
#####       Testa as variáveis uma a uma             #####
#####  Seleciona as variáveis por step-wise          #####
#####           Faz os testes...                     #####
############################################################

# Se ainda não instalado #
# install.packages("quantmod")
# install.packages("xts")
# install.packages("devtools")
# install.packages("GetDFPData2")
# install.packages("slider")
# install.packages("scales")

# Carregar bibliotecas
{ library(dplyr)
  library(future) # Estes para paralelizar
```

```
  library(future.apply) # Estes para paralelizar
  library(lubridate)
  library(PerformanceAnalytics) # Estes para paralelizar
  library(quantmod)  # Para getSymbols e cálculos financeiros
  library(readr)
  library(scales)
  library(slider)
  library(tidyr)
  library(xts)       # Para manipulação de séries temporais (opcional, ver depois se vale a pena)
}



options(scipen = 999)  # Desativa notação científica para números grandes

# Path PC Trabalho
# C:/---
# Path casa
# C:/----/AULAS_CURSOS/DSBD/TCC - DSBD/BASE_DADOS_TCC 02-07-2025/
# colocar o caminho do arquivo 'BASE_DADOS_TCC'
path <- "C:/------/TCC - DSBD/BASE_DADOS_TCC 02-07-2025/"

# Define os tipos das colunas (como no DTYPE_MAP)
{col_types <- cols(
  CNPJ_CIA = col_character(),
  CD_CVM = col_character(),
  CD_CONTA = col_character(),
  DS_CONTA = col_character(),
  VL_CONTA = col_double(),
  DT_REFER = col_date(format = "%Y-%m-%d"),  # Formato da data
  DT_FIM_EXERC = col_date(format = "%Y-%m-%d"),
  .default = col_guess()
)
}
arquivo <- "df_indicadores_20250704_0101.csv"
df_indicadores <- read_delim(
  file.path(path, arquivo), # 'BASE_DADOS_TCC'
  delim = ",",
  locale = locale(decimal_mark = ".", encoding = "latin1"),
  col_types = col_types
)


glimpse(df_indicadores)

#########################################################

# Baixar dados do Ibovespa e da ação dentro do período definido
# Atenção aqui, estes códigos já armazenam as variáveis, ver ali do lado o nome delas pra acessar depois!
# Para evitar de poluir o ambiente global, joguei tudo para dentro de uma função, aí já puxa os totais
# Ver depois aonde consegue um dicionário desses códigos aí (concluído :D)

# Função que baixa e retorna os retornos diários de um ativo (ou benchmark)


# Função que baixa e retorna os retornos diários de um ativo (ou benchmark)
obter_retornos <- function(symbol, start_date, end_date) {
  xts_obj <- getSymbols(symbol, from = start_date, to = end_date,
                        auto.assign = FALSE, warnings = FALSE, timeout = 30)
  na.omit(dailyReturn(Cl(xts_obj)))
}

# Função que calcula beta num único intervalo
calcular_beta <- function(ret_ativo, ret_mercado) {
```

```
  dados <- merge(ret_ativo, ret_mercado, join = "inner")
  colnames(dados) <- c("Ret_Ativo", "Ret_Mercado")
  if (nrow(dados) < 5) return(NA_real_)
  cov(dados$Ret_Ativo, dados$Ret_Mercado) / var(dados$Ret_Mercado)
}

# Função principal: itera por trimestres e tickers
calcular_betas_trimestrais <- function(lista_tickers, start_date, end_date) {
  # baixar benchmark e calcular retornos
  ret_benchmark <- obter_retornos("^BVSP", start_date, end_date)

  # gerar lista de trimestres exatos via endpoints()
  datas_all <- index(ret_benchmark)
  idx_q     <- endpoints(datas_all, on = "quarters")
  periodos  <- data.frame(
    Data_Inicio = datas_all[idx_q[-length(idx_q)] + 1],
    Data_Fim    = datas_all[idx_q[-1]]
  )

  # criar df de resultados vazio
  colunas  <- c("TICKER", "Data_Inicio", "Data_Fim", "Beta")
  resultados <- setNames(data.frame(matrix(ncol = length(colunas), nrow = 0)), colunas)

  # loop por ticker
  for (ticker in lista_tickers) {
    simbolo <- paste0(ticker, ".SA")
    # loop por cada trimestre
    for (i in seq_len(nrow(periodos))) {
      di <- periodos$Data_Inicio[i]
      df <- periodos$Data_Fim[i]

      # tenta baixar retornos e calcular beta
      beta_val <- tryCatch({
        ret_ativo <- obter_retornos(simbolo, di, df)
        b <- calcular_beta(ret_ativo, window(ret_benchmark, start = di, end = df))

        # Corrigir: força escalar e remove nomes herdados
        b <- as.numeric(round(b, 6))
        names(b) <- NULL
        b
      }, error = function(e) {
        NA_real_
      })

      # montagem da linha
      nova_linha <- data.frame(
        TICKER      = ticker,
        Data_Inicio = di,
        Data_Fim    = df,
        Beta        = beta_val,
        stringsAsFactors = FALSE
      )

      resultados <- rbind(resultados, nova_linha)
    }
  }

  resultados
}

# Exemplo de uso
{
start_date <- as.Date("2022-01-31")
end_date   <- as.Date("2024-12-31")
```

```
remover    <- c("ARZZ3", "BRML3", "BTOW3", "CIEL3", "IGTA3", "LAME4", "VIVT4")
tickers    <- setdiff(lista_tickers, remover)
tickers <- c("ABEV3", "VALE3", "PETR4", "ITSA4", "BBDC4")

betas_trimestrais <- calcular_betas_trimestrais(tickers, start_date, end_date)
}

# Função para normalizar datas
normalizar_datas <- function(df, colunas_dat, sufixo = "_norm") {
  # Verifica se o dataframe existe
  if (missing(df)) {
    stop("O argumento 'df' é obrigatório")
  }

  # Verifica se as colunas existem no dataframe
  colunas_inexistentes <- setdiff(colunas_dat, names(df))
  if (length(colunas_inexistentes) > 0) {
    stop(paste("As seguintes colunas não existem no dataframe:",
               paste(colunas_inexistentes, collapse = ", ")))
  }

  # Aplica a conversão para cada coluna especificada
  for (coluna in colunas_dat) {
    nova_coluna <- paste0(coluna, sufixo)
    df <- df %>%
      mutate(!!nova_coluna := format(as.Date(.data[[coluna]]), "%Y-%m"))
  }

  return(df)
}




#########################


# Se já tiver o csv betas, pode pular essas daqui
# Carregar os tickers
{
lista_tickers <- df_indicadores[["TICKER"]]
lista_tickers <- unique(lista_tickers)
# Colocar aqui as que dão problema... Torcer q não seja lista longa lol
remover <- c("ALOS3", "ARZZ3", "ASAI3","BRML3", "BTOW3",
             "CMIN3" ,"CIEL3", "IGTA3", "IGTI11",
             "LAME4", "LWSA3", "NTCO3", "PETZ3", "VIVT4")
lista_nova <- lista_tickers[!lista_tickers %in% remover]

# Definir período de análise (expandir depois)
# ajustar depois pra tornar dinamico)
# Data inicial (ver conforme o primeiro balanço/DRE do csv)
# Data final (ver conforme o último balanço/DRE do csv)
# Bloco 1 00:40 -
start_date <- as.Date("2019-01-01")
end_date   <- as.Date("2020-12-31")
remover <- c("ALOS3", "ASAI3", "CMIN3", "IGTI11",
             "LWSA3", "NTCO3", "PETZ3", "RAIZ4",
             "RDOR3", "RECV3", "VAMO3")
lista_nova2 <- lista_nova[!lista_nova %in% remover]
df_betas_1 <- calcular_betas_trimestrais(lista_nova2, start_date, end_date)


# Tickers com erro de download ou cálculo neste período:
#   [1] "ALOS3"  "ASAI3"  "CMIN3"  "IGTI11" "LWSA3"  "NTCO3"  "PETZ3"  "RAIZ4"  "RDOR3"
```

```
# [10] "RECV3"  "VAMO3"

# Bloco 2 23:46 - 23:52 (~6min)
start_date <- as.Date("2021-01-31")
end_date   <- as.Date("2024-12-31")
df_betas_2 <- calcular_betas_trimestrais(lista_nova2, start_date, end_date)

# Combinar resultados
df_betas_final <- bind_rows(df_betas_1, df_betas_2)




# Salvar o arquivo até aqui
current_datetime <- format(Sys.time(), "%Y%m%d_%H%M")
out_path <- file.path(path, paste0("df_betas_final", current_datetime, ".csv"))
write.csv(df_betas_final, out_path, row.names = FALSE)

}
# strip ano-mes-dia para ano-mes apenas
# converter para data e trocar formato
# facilita o join depois!

####################


arquivo <- "df_betas_final20250703_0102 --- usar este.csv"
df_betas <- read_csv(file.path(path, arquivo), # 'BASE_DADOS_TCC'
                     col_types = cols(
  Data_Inicio = col_date(format = "%Y-%m-%d"),
  Data_Fim = col_date(format = "%Y-%m-%d")
))
# "TICKER","Data_Inicio","Data_Fim","Beta"


# Passo 1: Preparar os dataframes

glimpse(df_indicadores) # DT_REFER
glimpse(df_betas) # Data_Inicio e Data_Fim

df_betas <- normalizar_datas(df_betas, c("Data_Inicio", "Data_Fim"))
df_indicadores <- normalizar_datas(df_indicadores, c("DT_REFER"))

glimpse(df_indicadores) # DT_REFER -> DT_REFER_norm
glimpse(df_betas) # Data_Inicio e Data_Fim -> Data_Fim_norm

# Selecionar colunas relevantes da tabela_principal
tabela_reduzida <- df_indicadores %>%
  select(TICKER, DT_REFER_norm, INFO_IND_CON,
         ATIVO_TOTAL_TTM_mm, DESPESAS_OPERACIONAIS_TTM_mm, EBIT_mm,
         LUCRO_BRUTO_TTM_mm, LUCRO_LIQUIDO_TTM_mm, PASSIVO_TOTAL_TTM_mm,
         PATRIMONIO_LIQUIDO_TTM_mm, RECEITA_BRUTA_TTM_mm, ROA_mm, ROE_mm)

# Passo 3: Fazer o left join
dados_completos_1 <- df_betas %>%
  select(TICKER, Data_Inicio_norm, Data_Fim_norm, Beta) %>%
  left_join(
    tabela_reduzida,
    by = c(
      "TICKER" = "TICKER",
      "Data_Fim_norm" = "DT_REFER_norm"  # Mesclar por ticker e data (mas ver depois de como incluir um
delay para pegar efeitos posteriores)
    )
  )
```

```
# Para análise em betas futuros
incluir_betas_futuros <- function(df_base, df_betas, n_trimestres = 1) {
  betas_temp <- df_betas %>%
    select(TICKER, Data_Inicio_norm, Data_Fim_norm, Beta) %>%
    rename(Beta_futuro = Beta) %>%
    mutate(
      Data_temp = ymd(paste0(Data_Fim_norm, "-01")),
      Data_deslocada = Data_temp %m+% months(-3 * n_trimestres),
      Data_base_norm = format(Data_deslocada, "%Y-%m")
    ) %>%
    select(TICKER, Data_base_norm, Beta_futuro)


  inner_join(df_base, betas_temp,
             by = c("TICKER", "Data_Fim_norm" = "Data_base_norm"))
}

# Exemplo de uso:
dados_completos_1 <- incluir_betas_futuros(dados_completos_1, df_betas)  # 1 trimestre de delay

colMeans(is.na(dados_completos_1)) * 100  # percentual de NAs por coluna

colunas_teste <- c("ATIVO_TOTAL_TTM_mm", "DESPESAS_OPERACIONAIS_TTM_mm", "EBIT_mm",
                   "LUCRO_BRUTO_TTM_mm", "LUCRO_LIQUIDO_TTM_mm", "PASSIVO_TOTAL_TTM_mm",
                   "PATRIMONIO_LIQUIDO_TTM_mm", "RECEITA_BRUTA_TTM_mm", "ROA_mm", "ROE_mm")


glimpse(dados_completos_1)

# percent_na <- colMeans(is.na(dados_completos_1[, colunas_teste]))
# # Manter apenas variáveis com menos de 40% de NA
# colunas_validas <- names(percent_na[percent_na < 0.4])
#
# # Subset dos dados completos sem NA nas variáveis válidas e na variável dependente
# dados_modelo <- dados_completos_1 %>%
#   select(Beta, all_of(colunas_validas)) %>%
#   drop_na()

# Construir fórmula e rodar o modelo
formula_modelo <- as.formula(paste("Beta ~", paste(colunas_teste, collapse = " + ")))
modelo <- lm(formula_modelo, data = dados_completos_1,
                   na.action = na.omit)

formula_modelo <- as.formula(paste("Beta_futuro ~", paste(colunas_teste, collapse = " + ")))
modelo_beta_futuro <- lm(formula_modelo, data = dados_completos_1,
                   na.action = na.omit)

# Ver resultado
summary(modelo)
summary(modelo_beta_futuro)
##################

# Trabalho que falta ver:

# Ver quais infos estão faltando do df principal e por que;
# Confirmar as datas dos informes (problema do ULTIMO/PENULTIMO nos dados da CVM) <<< Este consigo ver no
do Python, ver depois
# Qual abordagem utilizar para delay (indicador da empresa i no período j-1 comparar com beta da empresa i
no período j)
# Como tratar os dados faltantes (após refinar mais a busca?)
# Trabalhar e Torcer que dê tudo certo a tempo =)
```

```
# Organizar em TTM para as análises sendo a soma das variáveis quando INFO_IND_CON = "ind" ou apenas o
consolidado quando INFO_IND_CON = "con"
# Em seguida, para as variáveis explicativas, quero fazer com defasagem, o TTM do 4 trimestre de 2020 deve
comparar com o 1 trimestre de 2021 e assim por diante.
# Log, RobustScaler e MinMaxScaler em R
################## stepwise #################


### stepwise. Primeiramente fixando k = 2, estamos definindo o AIC como critério de
### seleção, temos:

### Método backward
step_back_AIC <- step(modelo_beta_futuro, direction = "backward", data = dados_completos_1, k = 2)
summary(step_back_AIC)

### Método forward. Para o método forward devemos definir o escopo da seleção
### (menor e maior modelo). O menor seria o modelo nulo (apenas com o intercepto),
### enquanto o maior seria o modelo com todas as covariáveis.


# Aqui deu problema com os NA então precisei excluir nas formulas, mas não exclui no teste,
# Será que dá problema?
# Seleciona as colunas necessárias para o modelo completo
vars_modelo <- all.vars(formula_modelo)
dados_completos <- dados_completos_1 %>%
  select(all_of(vars_modelo)) %>%
  drop_na()

# Agora adicione a variável dependente
dados_completos <- dados_completos_1 %>%
  filter(complete.cases(select(., all_of(vars_modelo))))

aj_lower <- lm(Beta_futuro~1, data = dados_completos)
aj_upper <- lm(formula_modelo, data = dados_completos)
formula(aj_upper)

step_for_AIC <- step(aj_lower, direction = "forward", scope=formula(aj_upper),
                     data = dados_completos_1, k = 2)
summary(step_for_AIC)

### Finalmente, o algoritmo que considera tanto exclusão quanto inclusão de
### covariáveis a cada passo
step_both_AIC <- step(modelo_beta_futuro, direction = "both", data = dados_completos_1, k = 2)
summary(step_both_AIC)

formula(step_both_AIC)

### Ajustes 1, 2, 3
# Ajuste 1 apenas o ROE, ROA, EBIT, PATRIMONIO (São variáveis que amplamente utilizam na literatura)

colunas_teste <- c("PATRIMONIO_LIQUIDO_TTM_mm", "EBIT_mm", "ROA_mm", "ROE_mm")

ajuste1 <- as.formula(paste("Beta_futuro ~", paste(colunas_teste, collapse = " + ")))
modelo_ajuste1 <- lm(ajuste1, data = dados_completos_1,
                         na.action = na.omit)
summary(modelo_ajuste1)


# Ajuste2 adiciona LUCRO_LIQUIDO_TTM_mm

colunas_teste <- c("PATRIMONIO_LIQUIDO_TTM_mm", "EBIT_mm", "ROA_mm", "ROE_mm", "LUCRO_LIQUIDO_TTM_mm")

ajuste2 <- as.formula(paste("Beta_futuro ~", paste(colunas_teste, collapse = " + ")))
```

```
modelo_ajuste2 <- lm(ajuste2, data = dados_completos_1,
                     na.action = na.omit)
summary(modelo_ajuste2)

### Vamos comparar os ajustes.
data.frame(compareCoefs(modelo_ajuste2, modelo_ajuste1, modelo_beta_futuro, step_back_AIC, step_for_AIC,
step_both_AIC))

# Para o primeiro modelo (step_both_AIC)
y_hat_step_both_AIC <- predict(step_both_AIC)

# Para o segundo modelo (ajuste4_sbothAIC)
y_hat_ajuste4_sbothAIC <- predict(step_both_AIC)


# Gráfico dos resíduos
plot(modelo_ajuste2$residuals, main = "Gráfico dos Resíduos",
     xlab = "Observações", ylab = "Resíduos", pch = 19)
abline(h = 0, col = "red", lty = 2)

# Gráficos de diagnóstico padrão
par(mfrow = c(2, 2))  # Configura a área de gráficos para 2x2
plot(modelo_ajuste2)




# Carregar bibliotecas
{ library(dplyr)
  library(quantmod)  # Para getSymbols e cálculos financeiros
  library(readr)
  library(scales)
  library(slider)
  library(tidyr)
  library(xts)       # Para manipulação de séries temporais
  library(purrr)
  library(stringr)
  options(scipen = 999)  # Desativa notação científica para números grandes
}


# Path PC Trabalho
# C:/----/-----/Amostra_empresas_2021-2024_20250702_1144.csv
# Path casa
# C:/-------/BASE_DADOS_TCC 02-07-2025/
# colocar o caminho do arquivo 'BASE_DADOS_TCC'
path <- "C:/------/BASE_DADOS_TCC 02-07-2025/"

# Define os tipos das colunas (como no DTYPE_MAP)
col_types <- cols(
  CNPJ_CIA = col_character(),
  CD_CVM = col_character(),
  CD_CONTA = col_character(),
  DS_CONTA = col_character(),
  VL_CONTA = col_double(),
  DT_REFER = col_date(format = "%Y-%m-%d"),  # Formato da data
  DT_FIM_EXERC = col_date(format = "%Y-%m-%d"),
  .default = col_guess()
)

empresas_com_NA <- read_delim(
  file.path(path, "empresas_com_NA.csv"), # 'BASE_DADOS_TCC'
  delim = ",",
```

```
  locale = locale(decimal_mark = ".", encoding = "latin1"),
  col_types = col_types
)

df_base <- empresas_com_NA %>%
  left_join(
    tabela_reduzida %>% select(TICKER, CNPJ_CIA, CD_CVM),
    by = "TICKER"
  )

df_base <- unique(df_base)
glimpse(df_base)
colMeans(is.na(df_base)) * 100  # percentual de NAs por coluna

out_path <- file.path(path, paste0("df_dre_filtrado_faltantes_.csv"))
write.csv(df_dre_filtrado, out_path, row.names = FALSE)

ler_dre_e_filtrar <- function(df_base, pasta_path) {
  # lista só arquivos que contenham "_cia_aberta_DRE_"
  arquivos <- list.files(pasta_path, pattern = "_cia_aberta_DRE_", full.names = TRUE)

  # para cada arquivo, lê e filtra
  purrr::map_dfr(arquivos, function(arquivo) {
    df <- read_delim(arquivo, delim = ";",
                     locale = locale(encoding = "latin1"),
                      col_types = cols_only(
                        CNPJ_CIA      = col_character(),
                        CD_CVM        = col_character(),
                        DT_REFER      = col_date(format = "%Y-%m-%d"),
                        DT_FIM_EXERC  = col_date(format = "%Y-%m-%d"),
                        CD_CONTA      = col_character(),
                        DS_CONTA      = col_character(),
                        VL_CONTA      = col_double(),
                        # ORDEM_EXERC   = col_character()
                      ))

    df %>%
      # # só os ORDEM_EXERC que sejam "ULTIMO" ou "ÚLTIMO" (case-insensitive)
      # filter(str_to_upper(ORDEM_EXERC) == "ULTIMO") %>%
      # DT_REFER igual a DT_FIM_EXERC
      filter(DT_REFER == DT_FIM_EXERC) %>%
      # junta com df_base pra manter só CNPJ_CIA/CD_CVM de interesse
      inner_join(df_base %>% select(TICKER, CNPJ_CIA, CD_CVM, INFO_IND_CON, DT_REFER),
                 by = c("CNPJ_CIA", "CD_CVM", "DT_REFER"),
                 relationship = "many-to-many")
  }) -> df_dre_filtrado

  return(df_dre_filtrado)
}

caminho_2 <- "C:/-------/TCC - DSBD/BASE_DADOS_TCC 02-07-2025/dados_faltantes/"
df_dre_filtrado <- ler_dre_e_filtrar(df_base, caminho_2)


glimpse(df_dre_filtrado)
df_dre_filtrado <- read_delim(
  file.path(path, "df_dre_filtrado_faltantes_norm.csv"), # 'BASE_DADOS_TCC'
  delim = ",",
  locale = locale(decimal_mark = "."),
  col_types = col_types
)

glimpse(df_dre_filtrado)
verif <- df_dre_filtrado %>% filter(CD_CONTA == "3.11") %>% select(DS_CONTA)
```

```
verif <- unique(verif)
verif <- df_dre_filtrado %>%
  filter(
    str_starts(CD_CONTA, "3.11") | str_starts(CD_CONTA, "3.13"),
    str_detect(DS_CONTA, regex("lucro", ignore_case = TRUE))
  )


verif <- verif %>%
  mutate(DS_CONTA = if_else(
    str_detect(DS_CONTA, regex("lucro.*", ignore_case = TRUE)),
    "LUCRO_LIQUIDO",
    DS_CONTA
  ))
verif <- verif %>% rename(DS_NORM = DS_CONTA) %>% select(TICKER, INFO_IND_CON, DT_REFER, DT_FIM_EXERC,
DS_NORM, VL_CONTA)

glimpse(verif)
df_completo <- bind_rows(df_com_tudo_junto, verif)



############################################################
#####      O script a seguir faz o seguinte:         #####
#####  Extrai as informações dos preços (Nosso Y)    #####
#####       Testa as variáveis uma a uma             #####
#####  Seleciona as variáveis por step-wise          #####
#####           Faz os testes...                     #####
############################################################


############################################################
#####       Carregar informações básicas             #####
############################################################

# Se ainda não instalado #
# install.packages("quantmod")
# install.packages("xts")
# install.packages("devtools")
# install.packages("GetDFPData2")
# install.packages("slider")
# install.packages("scales")

# Carregar bibliotecas
{ library(dplyr)
  library(future) # Estes para paralelizar
  library(future.apply) # Estes para paralelizar
  library(lubridate)
  library(PerformanceAnalytics) # Estes para paralelizar
  library(quantmod)  # Para getSymbols e cálculos financeiros
  library(readr)
  library(scales)
  library(slider)
  library(tidyr)
  library(xts)       # Para manipulação de séries temporais (opcional, ver depois se vale a pena)
}

options(scipen = 999)  # Desativa notação científica para números grandes

# colocar o caminho do arquivo 'BASE_DADOS_TCC_final'
# Path PC Trabalho
path <- "C:/--------/DSBD/TCC - DSBD/TCC 2025/BASE_DADOS_TCC_final/"
# Path casa
# path <- "C:/-----/TCC - DSBD/BASE_DADOS_TCC 02-07-2025/"
```

```
# Define os tipos das colunas (como no DTYPE_MAP)
{col_types <- cols(
  CNPJ_CIA = col_character(),
  CD_CVM = col_character(),
  CD_CONTA = col_character(),
  DS_CONTA = col_character(),
  VL_CONTA = col_double(),
  DT_REFER = col_date(format = "%Y-%m-%d"),  # Formato da data
  DT_FIM_EXERC = col_date(format = "%Y-%m-%d"),
  .default = col_guess()
)
}

arquivo <- "df_indicadores_contabeis.csv"
df_indicadores <- read_delim(
  file.path(path, arquivo), # 'BASE_DADOS_TCC'
  delim = ",",
  # locale = locale(decimal_mark = ".", encoding = "latin1"), # caso esteja com outro encoding
  col_types = col_types
)

arquivo <- "df_betas_final.csv"
df_betas <- read_csv(file.path(path, arquivo), # 'BASE_DADOS_TCC'
                     col_types = cols(
                       Data_Inicio = col_date(format = "%Y-%m-%d"),
                       Data_Fim = col_date(format = "%Y-%m-%d")
                     ))


glimpse(df_indicadores)
glimpse(df_betas)

############################################################
########## Funções e Transformações            ##########
############################################################

# Função para normalizar datas
normalizar_datas <- function(df, colunas_dat, sufixo = "_norm") {
  # Verifica se o dataframe existe
  if (missing(df)) {
    stop("O argumento 'df' é obrigatório")
  }

  # Verifica se as colunas existem no dataframe
  colunas_inexistentes <- setdiff(colunas_dat, names(df))
  if (length(colunas_inexistentes) > 0) {
    stop(paste("As seguintes colunas não existem no dataframe:",
               paste(colunas_inexistentes, collapse = ", ")))
  }

  # Aplica a conversão para cada coluna especificada
  for (coluna in colunas_dat) {
    nova_coluna <- paste0(coluna, sufixo)
    df <- df %>%
      mutate(!!nova_coluna := format(as.Date(.data[[coluna]]), "%Y-%m"))
  }

  return(df)
}

# Para análise em betas futuros
```

```
incluir_betas_futuros <- function(df_base, df_betas, n_trimestres = 1) {
  betas_temp <- df_betas %>%
    select(TICKER, Data_Inicio_norm, Data_Fim_norm, Beta) %>%
    rename(Beta_futuro = Beta) %>%
    mutate(
      Data_temp = ymd(paste0(Data_Fim_norm, "-01")),
      Data_deslocada = Data_temp %m+% months(-3 * n_trimestres),
      Data_base_norm = format(Data_deslocada, "%Y-%m")
    ) %>%
    select(TICKER, Data_base_norm, Beta_futuro)


  inner_join(df_base, betas_temp,
             by = c("TICKER", "Data_Fim_norm" = "Data_base_norm"))
}

# Passo 1: Preparar os dataframes

glimpse(df_indicadores) # DT_REFER
glimpse(df_betas) # Data_Inicio e Data_Fim

df_betas <- normalizar_datas(df_betas, c("Data_Inicio", "Data_Fim"))
df_indicadores <- normalizar_datas(df_indicadores, c("DT_REFER"))

glimpse(df_indicadores) # DT_REFER -> DT_REFER_norm
glimpse(df_betas) # Data_Inicio e Data_Fim -> Data_Fim_norm

# Selecionar colunas relevantes da tabela_principal
tabela_reduzida <- df_indicadores %>%
  select(-matches("(_log$|_robust$|_mm$)")) %>%
  select(-INFO_IND_CON)


# Passo 3: Fazer o left join
dados_completos_1 <- df_betas %>%
  select(TICKER, Data_Inicio_norm, Data_Fim_norm, Beta) %>%
  left_join(
    tabela_reduzida,
    by = c(
      "TICKER" = "TICKER",
      "Data_Fim_norm" = "DT_REFER_norm"  # Mesclar por ticker e data (mas ver depois de como incluir um
delay para pegar efeitos posteriores)
    )
  )



# incluir betas futuros
dados_completos_1 <- incluir_betas_futuros(dados_completos_1, df_betas)  # 1 trimestre de delay

# Reordenando aqui
tabela_reduzida <- dados_completos_1 %>%
  select(TICKER, Data_Fim_norm, Beta, Beta_futuro, everything()) %>%
  rename(DT_REFER_ttm = Data_Fim_norm) %>%
  select(-DT_REFER, -Data_Inicio_norm, -LPA_CVM_TTM)

glimpse(tabela_reduzida)
colMeans(is.na(tabela_reduzida)) * 100  # percentual de NAs por coluna

# Salvar tudo até aqui
# Obtém data e hora no formato desejado
current_datetime <- format(Sys.time(), "%Y%m%d_%H%M")
out_path <- file.path(path, paste0("df_dados_limpinhos_e_trabalhados__", current_datetime, "__.csv"))
write.csv(tabela_reduzida, out_path, row.names = FALSE)
```

```r
# Normalizando os dados:
# Log, RobustScaler e MinMaxScaler em R

# vetor de colunas a padronizar
colunas_teste <- c("Beta", "Beta_futuro",
                   "ATIVO_TOTAL_TTM", "DESPESAS_OPERACIONAIS_TTM", "EBIT",
                   "LUCRO_BRUTO_TTM", "LUCRO_LIQUIDO_TTM", "PASSIVO_TOTAL_TTM",
                   "PATRIMONIO_LIQUIDO_TTM", "RECEITA_BRUTA_TTM", "ROA", "ROE")

# Funções de transformação

log_transform <- function(x) {
  case_when(
    is.na(x)  ~ NA_real_, # aqui se for NA
    x > 0     ~ log1p(x),
    TRUE      ~ -log1p(abs(x)) # Aqui se for negativo ou 0
  )
}

robust_scale <- function(x) {
  med <- median(x, na.rm = TRUE)
  iqr <- IQR(x,   na.rm = TRUE)
  (x - med) / iqr
}

minmax_scale <- function(x) {
  # utiliza scales::rescale
  scales::rescale(x, to = c(0, 1))
}


# Função para padronizar colunas com flags de transformação
padronizar_cols <- function(data_frame, colunas_teste,
                            log = TRUE, robust = TRUE, mm = TRUE) {
  df <- data_frame

  for (col in colunas_teste) {
    x <- df[[col]]

    # 1) Log-transform se requisitado
    if (log)    x <- log_transform(x)
    # 2) Robust-scale se requisitado
    if (robust) x <- robust_scale(x)
    # 3) MinMax se requisitado
    if (mm)     x <- minmax_scale(x)

    # definir sufixo da coluna resultante
    sufixo <- if (mm) {
      "_mm"
    } else if (robust) {
      "_robust"
    } else if (log) {
      "_log"
    } else {
      ""
    }

    nome_novo <- paste0(col, sufixo)
    df[[nome_novo]] <- x
  }

  # Remover as colunas originais que foram transformadas
```

```
  df <- df[, setdiff(names(df), colunas_teste)]
  df
}


############################################################
########## Transformações e análises           ##########
############################################################

# Aplicando normalizações conforme de diferentes formas

# Teste padrão com tudo junto
teste_completo <- padronizar_cols(tabela_reduzida, colunas_teste, log = TRUE, robust = TRUE, mm = TRUE)

# Testes com normalizações isoladas
{
teste_log <- padronizar_cols(tabela_reduzida, colunas_teste, log = TRUE, robust = FALSE, mm = FALSE)

teste_robust_scale <- padronizar_cols(tabela_reduzida, colunas_teste, log = FALSE, robust = TRUE, mm =
FALSE)

teste_mm <- padronizar_cols(tabela_reduzida, colunas_teste, log = FALSE, robust = FALSE, mm = TRUE)
}

# Testes com log+min-max ou robust+min-max
{
teste_log_mm <- padronizar_cols(tabela_reduzida, colunas_teste, log = TRUE, robust = FALSE, mm = TRUE)

teste_rs_mm <- padronizar_cols(tabela_reduzida, colunas_teste, log = FALSE, robust = TRUE, mm = TRUE)
}

glimpse(tabela_reduzida)

testes_rodar <- list(
  completo      = teste_completo,
  log           = teste_log,
  robust_scale  = teste_robust_scale,
  mm            = teste_mm,
  log_mm        = teste_log_mm,
  rs_mm         = teste_rs_mm
)



############################################################
########## Transformações e análises           ##########
############################################################

# rodar modelos conforme a padronização e conforme os betas

rodar_modelos_beta <- function(nome_modelo, df, sufixo) {

  # nomes a excluir, montados dinamicamente
  colunas_excluir <- c("TICKER", "DT_REFER_ttm",
                       paste0("Beta", sufixo),
                       paste0("Beta_futuro", sufixo))

  # colunas independentes: tudo menos as de exclusão
  colunas_teste <- setdiff(names(df), colunas_excluir)

  # checagem: não rodar se não houver variáveis independentes
  if (length(colunas_teste) == 0) {
    stop(paste("Nenhuma variável independente encontrada em", nome_modelo))
  }
```

```
  # monta fórmulas dinâmicas
  formula_beta         <- as.formula(paste0("Beta", sufixo, " ~ ", paste(colunas_teste, collapse = " + ")))
  formula_beta_futuro  <- as.formula(paste0("Beta_futuro", sufixo, " ~ ", paste(colunas_teste, collapse = "
+ ")))

  # ajusta os modelos
  modelo_beta <- lm(formula_beta, data = df, na.action = na.omit)
  modelo_beta_futuro <- lm(formula_beta_futuro, data = df, na.action = na.omit)

  # saída nomeada
  list(
    nome = nome_modelo,
    modelo_beta = modelo_beta,
    modelo_beta_futuro = modelo_beta_futuro
  )
}

glimpse(testes_rodar[["completo"]])
mod_teste_completo <- rodar_modelos_beta("teste_completo", testes_rodar[["completo"]], "_mm")
names(mod_teste_completo)
summary(mod_teste_completo[["nome"]])

# Salvando Resultados

{
  # Supondo que mod seja seu objeto (ex: mod_teste_completo)
  mod <- mod_teste_completo
  arquivo <- "resumo_modelo_teste_completo.txt"

  # Abre conexão de escrita
  sink(arquivo)

  # Cabeçalho com nome do modelo
  cat("# Modelo testado:", mod[["nome"]], "\n\n")

  # Resumo de Beta
  cat("# Beta summary:\n")
  print(summary(mod[["modelo_beta"]]))
  cat("\n")

  # Resumo de Beta Futuro
  cat("# Beta Futuro summary:\n")
  print(summary(mod[["modelo_beta_futuro"]]))
  cat("\n")

  # Fecha conexão
  sink()

}
export_modelo_txt <- function(mod, arquivo) {
  sink(arquivo)
  cat("# Modelo testado:", mod[["nome"]], "\n\n")
  cat("# Beta summary:\n")
  print(summary(mod[["modelo_beta"]]))
  cat("\n# Beta Futuro summary:\n")
  print(summary(mod[["modelo_beta_futuro"]]))
  sink()
}

# Exemplo de uso
export_modelo_txt(mod_teste_completo, "resumo_teste_completo.txt")
export_modelo_txt(resultados$log,       "resumo_teste_log.txt")
export_modelo_txt(resultados$mm,        "resumo_teste_mm.txt")
```

```
mod_teste_completo <- rodar_modelos_beta("teste_mm", testes_rodar[["mm"]], "_mm")

# Construir fórmula e rodar o modelo
formula_modelo <- as.formula(paste("Beta ~", paste(colunas_teste, collapse = " + ")))
modelo <- lm(formula_modelo, data = dados_completos_1,
             na.action = na.omit)

formula_modelo_f <- as.formula(paste("Beta_futuro ~", paste(colunas_teste, collapse = " + ")))
modelo_beta_futuro <- lm(formula_modelo, data = dados_completos_1,
                         na.action = na.omit)

# Ver resultado
summary(modelo)
summary(modelo_beta_futuro)
##################

# Trabalho que falta ver:

# Ver quais infos estão faltando do df principal e por que;
# Confirmar as datas dos informes (problema do ULTIMO/PENULTIMO nos dados da CVM) <<< Este consigo ver no
do Python, ver depois
# Qual abordagem utilizar para delay (indicador da empresa i no período j-1 comparar com beta da empresa i
no período j)
# Como tratar os dados faltantes (após refinar mais a busca?)
# Trabalhar e Torcer que dê tudo certo a tempo =)



# Organizar em TTM para as análises sendo a soma das variáveis quando INFO_IND_CON = "ind" ou apenas o
consolidado quando INFO_IND_CON = "con"
# Em seguida, para as variáveis explicativas, quero fazer com defasagem, o TTM do 4 trimestre de 2020 deve
comparar com o 1 trimestre de 2021 e assim por diante.
# Log, RobustScaler e MinMaxScaler em R
################## stepwise ##################


### stepwise. Primeiramente fixando k = 2, estamos definindo o AIC como critério de
### seleção, temos:

### Método backward
step_back_AIC <- step(modelo_beta_futuro, direction = "backward", data = dados_completos_1, k = 2)
summary(step_back_AIC)

### Método forward. Para o método forward devemos definir o escopo da seleção
### (menor e maior modelo). O menor seria o modelo nulo (apenas com o intercepto),
### enquanto o maior seria o modelo com todas as covariáveis.


# Aqui deu problema com os NA então precisei excluir nas formulas, mas não exclui no teste,
# Será que dá problema?
# Seleciona as colunas necessárias para o modelo completo
vars_modelo <- all.vars(formula_modelo)
dados_completos <- dados_completos_1 %>%
  select(all_of(vars_modelo)) %>%
  drop_na()

# Agora adicione a variável dependente
dados_completos <- dados_completos_1 %>%
  filter(complete.cases(select(., all_of(vars_modelo))))

aj_lower <- lm(Beta_futuro~1, data = dados_completos)
```

```
aj_upper <- lm(formula_modelo, data = dados_completos)
formula(aj_upper)

step_for_AIC <- step(aj_lower, direction = "forward", scope=formula(aj_upper),
                     data = dados_completos_1, k = 2)
summary(step_for_AIC)

### Finalmente, o algoritmo que considera tanto exclusão quanto inclusão de
### covariáveis a cada passo
step_both_AIC <- step(modelo_beta_futuro, direction = "both", data = dados_completos_1, k = 2)
summary(step_both_AIC)

formula(step_both_AIC)

### Ajustes 1, 2, 3
# Ajuste 1 apenas o ROE, ROA, EBIT, PATRIMONIO (São variáveis que amplamente utilizam na literatura)

colunas_teste <- c("PATRIMONIO_LIQUIDO_TTM_mm", "EBIT_mm", "ROA_mm", "ROE_mm")

ajuste1 <- as.formula(paste("Beta_futuro ~", paste(colunas_teste, collapse = " + ")))
modelo_ajuste1 <- lm(ajuste1, data = dados_completos_1,
                     na.action = na.omit)
summary(modelo_ajuste1)


# Ajuste2 adiciona LUCRO_LIQUIDO_TTM_mm

colunas_teste <- c("PATRIMONIO_LIQUIDO_TTM_mm", "EBIT_mm", "ROA_mm", "ROE_mm", "LUCRO_LIQUIDO_TTM_mm")

ajuste2 <- as.formula(paste("Beta_futuro ~", paste(colunas_teste, collapse = " + ")))
modelo_ajuste2 <- lm(ajuste2, data = dados_completos_1,
                     na.action = na.omit)
summary(modelo_ajuste2)

### Vamos comparar os ajustes.
data.frame(compareCoefs(modelo_ajuste2, modelo_ajuste1, modelo_beta_futuro, step_back_AIC, step_for_AIC,
step_both_AIC))

# Para o primeiro modelo (step_both_AIC)
y_hat_step_both_AIC <- predict(step_both_AIC)

# Para o segundo modelo (ajuste4_sbothAIC)
y_hat_ajuste4_sbothAIC <- predict(step_both_AIC)


# Gráfico dos resíduos
plot(modelo_ajuste2$residuals, main = "Gráfico dos Resíduos",
     xlab = "Observações", ylab = "Resíduos", pch = 19)
abline(h = 0, col = "red", lty = 2)

# Gráficos de diagnóstico padrão
par(mfrow = c(2, 2))  # Configura a área de gráficos para 2x2
plot(modelo_ajuste2)
```