UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
GRADUAÇÃO EM ESTATÍSTICA

Gustavo Almeida Silva

Modelos para Séries Temporais de Contagem: implementação dos modelos GLARMA e INGARCH no Software R

Juiz de Fora

2025

Gustavo Almeida Silva

Modelos para Séries Temporais de Contagem: implementação dos modelos GLARMA e INGARCH no Software R

Trabalho de conclusão de curso apresentado ao Departamento de Estatística da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de bacharel em Estatística

Orientador: Professor Doutor Marcel de Toledo Vieira

Juiz de Fora

2025

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF com os dados fornecidos pelo(a) autor(a)

Almeida Silva, Gustavo.
Modelos para Séries Temporais de Contagem: implementação dos modelos GLARMA e INGARCH no Software R / Gustavo Almeida Silva. – 2025.
127 f. : il.

Orientador: Marcel de Toledo Vieira
Trabalho de Conclusão de Curso – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Graduação em Estatística, 2025.

1. Séries Temporais. 2. Modelos Generalizados. 3. Dados de Contagem. 4. Programação em R I. Vieira, Marcel, orient. II. Modelos para Séries Temporais de Contagem: implementação dos modelos GLARMA e INGARCH no Software R.

Gustavo Almeida Silva

Modelos para Séries Temporais de Contagem: implementação dos modelos GLARMA e INGARCH no Software R

Trabalho de conclusão de curso apresentado ao Departamento de Estatística da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de bacharel em Estatística

Aprovada em 28 de Novembro de 2025

BANCA EXAMINADORA

Professor Doutor Marcel de Toledo Vieira - Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. João Henrique Gonçalves Mazzeu
Universidade Federal de Juiz de Fora

Prof. Dr. Augusto Carvalho Souza
Universidade Federal de Juiz de Fora

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me guardar, fortalecer e capacitar ao longo de toda essa jornada.

À minha mãe, Alessandra, pelo amor, apoio e suporte incondicional, mesmo eu estando a mais de 500 km de distância de casa. As ligações e mensagens diárias foram essenciais para que eu não desistisse, mesmo nos dias mais difíceis. Obrigado mãe!

Ao meu pai, Márcio, por sempre me incentivar a perseguir meus sonhos e por estar ao meu lado em todas as etapas do caminho.

Agradeço à UFJF, ao Departamento de Estatística e a todos os professores, pelo ensino de excelência que recebi.

Expresso minha gratidão especial ao Prof. Marcel Toledo, por todo o aprendizado, auxílio e paciência, acompanhando minha formação desde os primeiros períodos até a conclusão do curso. Sendo um orientador acadêmico e um mentor da vida. Minha profunda admiração! Sem ele, certamente nada disso teria sido possível.

Agradeço também ao Prof. Augusto Souza, meu primeiro orientador de bolsa, que despertou em mim o interesse por programação e por séries temporais;

Ao Prof. Lupércio, agradeço pelos ensinamentos que ultrapassam a sala de aula;

Ao Prof. Tiago Magalhães, registro meu agradecimento por todo período acadêmico, mas em especial pela orientação durante o período final da graduação.

Aos demais mestres, que tive a honra de ter sido aluno; Clécio, Camila, Ronaldo, Ângela e Tufi. Obrigado por todo aprendizado

Aos meus queridos amigos de sala, Pedro, Arthur, Camila, Joysce e Natasha, agradeço por todas as conversas, risadas e momentos compartilhados. Levo comigo cada aprendizado e alegria vivida com vocês.

Aos meus irmãos em Cristo que se tornaram irmãos de vida, Gadu e Patrick, agradeço pelas inúmeras histórias, viagens e risadas. Minha profunda admiração por vocês e pelo cuidado com a família e com todos ao redor. Agradeço a Cristo por nossa amizade

Aos demais irmãos em Cristo que conheci em Juiz de Fora, Bia, João, Gabi, Laís e Anne, agradeço de coração. Foi uma bênção compartilhar meus dias em Juiz de Fora com vocês.

"The Good Lord made all the integers; the rest is man's doing."
(Leopold Kronecker)

RESUMO

Os modelos de séries temporais elaborados por Box e Jenkins em 1970 constituem a principal classe de modelos temporais atuais, os denominados modelos ARMA(Autorregressivo de Médias Móveis). Embora esses modelos tenham se tornado populares em diversas áreas, sua aplicação em séries de contagem, compostas por valores discretos que representam eventos específicos como número de clientes ou casos de doenças, apresentam desafios singulares. Nesse contexto, os modelos que utilizam distribuições de probabilidade de contagem, sendo as principais Poisson e Binomial Negativa, mostram-se como importantes ferramentas para esse tipo de dados. Assim, o trabalho teve como objetivo descrever a implementação dos modelos GLARMA (Autorregressivo de Médias Móveis Linear Generalizado) e INGARCH (Heterocededasticidade Condicional Autorregressivo Generalizado de Valores Inteiros) no pacote do software R fableCount. Além das definições matemáticas e estatísticas de estimação e previsão para cada modelo, os seus aspectos computacionais também foram desenvolvidos, como os métodos numéricos de estimação, construção de intervalos de previsão via bootstrap e métodos de validação cruzada de séries temporais. Outra importante funcionalidade do pacote explicitada no trabalho são os algoritmos para modelagem automatizada, divididos em: seleção automática de distribuição, seleção automática de ordens de parâmetros, e seleção automática de melhor modelo para previsão. Para a aplicação, dados epidemiológicos foram selecionados, onde um considerável ganho de desempenho e interpretação foi visto ao se trabalhar com os modelos temporais de contagem comparados aos modelos ARMA e NNETAR . Ao final, métricas de utilização do pacote e planos futuros são descritos,

Palavras-chave: Séries Temporais; Modelos Generalizados; Dados de Contagem; Programação em R.

ABSTRACT

The time series models developed by Box and Jenkins in 1970 constitute the main class of current temporal models, the so-called ARMA (Autoregressive Moving Average) models. Although these models are popular in many areas, their application to counting series, composed of discrete values that represent specific events such as number of customers or disease cases, presents unique challenges. In this context, models that use counting probability distributions, the main ones being Poisson and Negative Binomial, prove to be important tools for this type of data. Thus, the aim of the work was to describe the implementation of the GLARMA (Generalized Linear Autoregressive Moving Average) and INGARCH (Integer-valued Generalized Autoregressive Conditional Heteroscedasticity) models in the R software package fableCount. In addition to the mathematical and statistical definitions of estimation and forecasting for each model, computational aspects were also developed, including numerical estimation methods, construction of prediction intervals via bootstrap, and time series cross-validation procedures. Another important feature of the package explained in the work are the automated modeling algorithms, divided into: automatic distribution selection, automatic parameter order selection and automatic model selection for forecasting. For the application, epidemiological data were used, where a significant improvement in both forecasting performance and interpretability was observed when using count time series models compared to ARMA and NNETAR models. At the end, package usage metrics and future development plans are described

Keywords: Time Series; Generalized Models; Counting Data; R Programing.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

MLG	Modelo Linear Generalizado
ARMA	Autoregressive Moving Average (Autorregressivo de Médias Móveis)
ARIMA	Autoregressive Integrated Moving Average (Autorregressivo Integrado de Médias Móveis)
GLARMA	Generalized Linear Autoregressive Moving Average (Modelo Linear Generalizado Autorregressivo de Médias Móveis)
NNETAR	Neural Network Autoregression (Rede Neural Autorregressiva)
ARCH	Autoregressive Conditional Heteroskedasticity (Autorregressivo de Heterocedasticidade Condicional)
GARCH	Generalized Autoregressive Conditional Heteroskedasticity (Autorregressivo Generalizado com Heterocedasticidade Condicional)
INGARCH	Integer-valued Generalized Autoregressive Conditional Heteroskedasticity (Modelo de Contagem Autorregressivo Generalizado de Heterocedasticidade Condicional)
FAC	Função de Autocorrelação
FACP	Função de Autocorrelação Parcial
AIC	Akaike Information Criteria (Critério de Informação de Akaike)
BIC	Bayesian Information Criteria (Critério de Informação Bayesiano)
RMSE	Root Mean Squared Error (Raiz Quadrada do Erro Quadrático Médio)
MASE	Mean Absolute Scaled Error (Erro Absoluto Médio Escalado)
MAPE	Mean Absolute Percentage Error (Erro Percentual Absoluto Médio)
OOS	Out-of-Sample (Fora da Amostra)
SRAG	Síndrome Respiratória Aguda Grave
TSCV	Time Series Cross-Validation (Validação Cruzada para Séries Temporais)

SUMÁRIO

1 INTRODUÇÃO

1.1 SÉRIES TEMPORAIS

Dentro do campo da teoria das probabilidades, um processo estocástico descreve a evolução aleatória de um sistema ao longo do tempo, sendo representado por uma família de variáveis aleatórias. Isso significa que, em um processo estocástico, a evolução futura do sistema não é completamente previsível, mesmo quando a condição inicial é conhecida. Existem múltiplas direções possíveis para a evolução do sistema, algumas vezes até mesmo infinitas, devido à presença da aleatoriedade. Em contraste, um processo determinístico é aquele em que a evolução futura do sistema é completamente determinada pela sua condição inicial. Não há incerteza envolvida; dado o estado inicial do sistema, é possível prever exatamente como ele evoluirá ao longo do tempo. Essa diferença fundamental torna os processos estocásticos ferramentas valiosas para modelar uma ampla gama de fenômenos da vida real, onde a aleatoriedade desempenha um papel significativo, (Parzen 1961)

Dentre a classe de processos estocásticos, destacam-se as séries temporais, amplamente utilizadas para modelar diversos problemas reais. Alguns exemplos incluem:

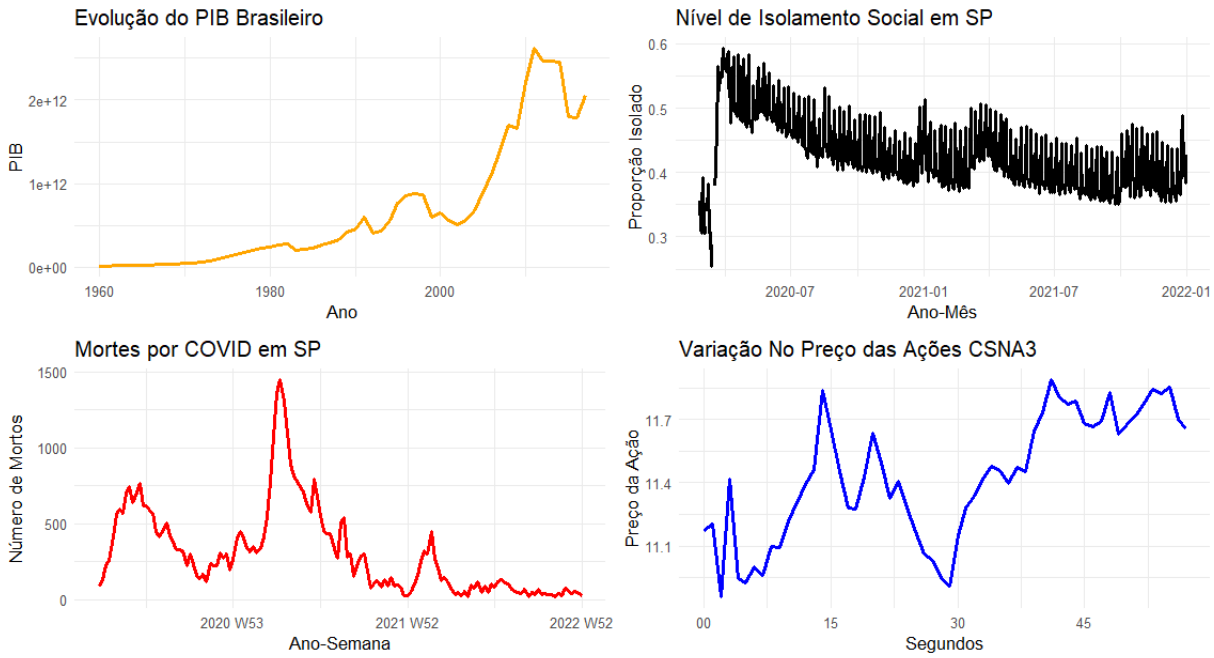
- Evolução do PIB: O Produto Interno Bruto (PIB) de um país é calculado anualmente e apresenta variações que dependem de inúmeros fatores econômicos e sociais. Essas variações não são completamente previsíveis, caracterizando o PIB como um processo estocástico.

- Movimentação do preço de uma ação: O preço de uma ação na bolsa de valores flutua constantemente ao longo do dia. Essas flutuações dependem de uma combinação de fatores, como notícias econômicas, desempenho da empresa, e comportamento dos investidores, tornando a previsão dos preços futuros um processo estocástico.

- Grau de isolamento social durante a pandemia: Durante a pandemia de COVID-19, o grau de isolamento social variou mês a mês, influenciado por medidas governamentais, conscientização da população, e taxa de infecção. Essas variações apresentam um comportamento estocástico, dado que não são completamente previsíveis.

- Número de mortos por COVID-19 a cada semana epidemiológica: O número de mortos por COVID-19 também variou semanalmente, sendo influenciado por diversos fatores como a taxa de contágio, a eficácia das medidas de controle, e o estado do sistema de saúde. Essa variação semanal é um exemplo de processo estocástico, já que não pode ser prevista com precisão total.

4 exemplos distintos de Séries Temporais



Elaborado pelo autor (2025)

De maneira teórica, uma série temporal é um conjunto de observações de uma determinada variável feitas em períodos sucessivos de tempo, ao longo de um determinado intervalo. Essas séries podem ser tratadas como amostras aleatórias ordenadas no tempo; a ordem em que são feitas as medições é fundamental e não pode ser esquecida. A sequência temporal das observações é crucial porque cada valor depende, em maior ou menor grau, dos valores anteriores. Existem diferenças importantes entre séries temporais e dados transversais (cross-section). A análise de séries temporais considera a dependência temporal entre as observações. Isso significa que os valores em diferentes instantes de tempo estão correlacionados, e a ordem das observações é essencial para a análise. Por exemplo, o preço de uma ação hoje pode ser influenciado pelo preço de ontem e pela expectativa para o preço de amanhã. Em contraste, os dados transversais são coletados em um único ponto no tempo, mas em diferentes locais ou sujeitos. Esses dados possuem apenas dependência espacial e não linear; um valor alto de um dado não necessariamente indica algo sobre o valor de outro dado. Por exemplo, um estudo que coleta a renda de diferentes famílias em um país em um determinado ano trata de dados transversais. Aqui, a ordem das observações não é importante, e cada observação é independente das outras. Em resumo, a principal diferença reside na dependência temporal presente nas séries temporais, onde a ordem das observações é essencial, enquanto nos dados transversais, cada observação é independente das demais e a ordem das observações não é relevante.(Miranda 2014)

Dentro os exemplos de séries temporais citados, diferentes modelos seriam sugeri-

dos para cada um deles, como Nowcasting para bolsa de valores e métodos sazonais para isolameto e mortes. Apesar da variedade de modelos disponíveis para a análise e modelagem desse tipo de dados, todos esses compartilham uma característica fundamental: a correlação entre as observações ao longo do tempo.

Dentre os modelos e métodos de previsões mais conhecidos, estão aqueles iniciados por Box e Jenkins em 1970, conhecidos como modelos Autorregressivos Integrados de Médias Móveis, ou apenas ARIMA. Assim como uma grande parcela de outros modelos temporais, tal classe é escrita acompanhada por seus parâmetros, onde no caso de um ARIMA, tem-se os parâmetros Autorregressivos, Diferenciação (Integrado) e de Médias Móveis, formando assim um ARIMA(p, d, q). Esses indicam a ordem das 3 diferentes parcelas do modelo:

(i) **Autorregressiva**, que busca capturar a relação linear entre uma observação e um número fixo de observações passadas, refletindo a influência das próprias observações passadas na atual;

(ii) **Integrado**, representa o número de diferenciações necessárias para tornar a série estacionária;

(iii) **Médias Móveis**, que busca modelar o erro a cada defasagem, permitindo capturar padrões não capturados pela parcela autorregressiva.

Juntas, essas parcelas fornecem uma estrutura flexível para modelar uma ampla gama de séries temporais. Durante o trabalho, os valores que esses parâmetros podem assumir serão chamados de **ordem**. Assim como será detalhado nas seções busca automática de ordens de parâmetros

Apesar de tal modelo não ser o foco do trabalho, ele se destaca por ser a base para outras classes de modelos temporais, onde essas que abordadas serão posteriormente.

1.2 JFSALVANDOTODOS E PREVISÕES EPIDEMIOLÓGICAS

Conforme mencionado na seção anterior, um dos principais nichos da modelagem de séries temporais está voltado para a previsão epidemiológica. Questões como a disponibilidade de leitos hospitalares, o aumento no número de bombas de oxigênio ou a aquisição de medicamentos exigem um planejamento logístico que depende não apenas da observação das necessidades atuais, mas também de projeções confiáveis para o futuro. Nesse contexto, um dos grandes motivadores para o trabalho e o tema estudado foram as dificuldades encontradas na modelagem preditiva de dados epidemiológicos para a plataforma de análise epidemiológica **JF SalvandoTodos**.

A Plataforma JF SalvandoTodos, fundada em março de 2020, surgiu com o objetivo de disponibilizar dados sobre a evolução da pandemia de COVID-19 e difundir informação de qualidade, consolidando-se como difusora científica, já que no início da pandemia pouco

se sabia sobre o novo coronavírus. (Prado 2022)

A plataforma permite a visualização de dados sobre a evolução da pandemia da COVID-19 de forma gratuita, segura, simples e amigável para todos os municípios do Brasil, regiões de saúde, regiões do IBGE, Unidades da Federação, para a Região Integrada de Desenvolvimento do Distrito Federal e para o país como um todo. Ela foi idealizada assim que a pandemia da COVID-19 foi decretada pela Organização Mundial da Saúde e foi fundada em 29/03/2020 pelo professor Marcel de Toledo Vieira do Departamento de Estatística da UFJF e pelo Estatístico Pedro Pacheco, então aluno do Curso de Estatística e atualmente formado pela UFJF que foi o responsável pelo desenvolvimento e programação.

A plataforma é uma aplicação web construída em RShiny, que disponibiliza os dados epidemiológicos através de mapas e gráficos interativos

Após mais de 4 anos de seu início, além dos dados da COVID-19 a plataforma conta hoje com dados sobre as SRAGs (Síndromes Respiratórias Agudas Graves), trazendo número de pessoas internadas por conta de COVID, Influenza e outras vírus que atacam o sistema respiratório.

Buscando inovações ao trazer informações relevantes a gestores de saúde, em Outubro de 2023 a plataforma inaugurou sua seção de previsão epidemiológica para os dados de SRAGs, possuindo previsão para unidade federal, grandes regiões, unidades federativas, mesorregiões e microrregiões

A previsão na plataforma busca prever casos confirmados para um intervalo de 1 mês, possuindo estimativas pontuais e intervalares para cada uma das semanas do mês estudado. Inicialmente, 2 modelos foram utilizados para as previsões, sendo eles o ARIMA (Autorregressivo Integrado de Médias Móveis) e o NNETAR (Autorregressivo de Redes Neurais). Apesar dos modelos apresentarem bom desempenho para o Brasil, regiões e suas unidades federativas, esses apresentaram diversos problemas para previsões para macro, microrregiões e municípios (em fase de teste)

Tais problemas compõem a seguinte lista

- **Desempenho Computacional** - Atualmente, o Brasil é dividido em 5 grandes regiões, 137 mesorregiões, 558 microrregiões e 5.568 municípios. Para contemplar todas essas hierarquias geográficas, implementamos mensalmente modelos em cada uma delas, totalizando 6.263 unidades geográficas. Como são considerados dois modelos por unidade (ARIMA e NNETAR), isso resulta na implementação de mais de 12.500 modelos por mês na plataforma. Para a estimação desse número elevado de modelos, a avaliação de funções de autocovariância e autocovariância parcial de maneira manual para a escolha das ordens de cada parâmetro é inviável, e para isso algoritmos para seleção automática de parâmetros são utilizados. Tais fatores impli-

cam em um tempo de execução do processo de modelagem extremamente elevado. O modelo ARIMA apresenta um tempo de execução computacional relativamente eficiente, completando um ciclo de estimação e previsão definido como a execução dos modelos para as 6.263 unidades geográficas em aproximadamente 5 horas. Em contrapartida, o modelo NNETAR demanda um tempo substancialmente maior, podendo ultrapassar 12 horas por ciclo, a depender da complexidade e do número de parâmetros envolvidos na rede neural. Esse valor representa um aumento de cerca de 140% no tempo de execução em relação ao modelo ARIMA, o que pode impactar significativamente a escalabilidade e a atualização frequente das previsões na plataforma.

- **Erros para Previsões Intervalares**

- Esse problema manifesta-se principalmente em modelos do tipo ARIMA . Tais modelos pressupõem que os erros seguem uma distribuição de probabilidade Gaussiana, a qual, por definição, atribui probabilidade a todo o conjunto dos números reais, inclusive a valores negativos. Em contextos onde os números de casos confirmados são baixos por exemplo, localidades cuja média semanal de casos permanece inferior a 10 , a construção de intervalos de confiança baseada nessa premissa pode resultar em limites inferiores negativos. Em situações práticas, isso leva a interpretações absurdas, como afirmar que um município possui 95% de chance de registrar entre -10 e 10 casos em uma determinada semana do próximo mês, o que é incoerente, dado que o número de casos não pode ser negativo.

O livro Fpp3 - Forecasting: Principles and Practice (3rd ed) (Hyndman e Athanasopoulos 2021) menciona que problemas em que as previsões precisam ser restritas a um intervalo específico [a,b] podem ser atenuados mediante o uso de transformações de variáveis. Uma das transformações sugeridas é a função loggit escalonada, definida por:

$$log(\frac{x-\alpha}{\beta-x}), \tag{1.1}$$

em que α e β são parâmetros que delimitam o intervalo permitido, ambos assumindo valores inteiros. Essa transformação é de fácil implementação e impõe um custo computacional relativamente baixo. No entanto, apesar de suas vantagens operacionais, ela acarreta uma série de efeitos colaterais importantes.

Primeiramente, a aplicação dessa transformação induz assimetria nos intervalos de confiança obtidos, dificultando a interpretação direta dos resultados. Além disso, compromete-se a interpretabilidade dos parâmetros autoregressivos e de médias móveis do modelo, uma vez que as transformações alteram a estrutura original das relações temporais. Mais criticamente, na modelagem de dados epidemiológicos como séries de casos confirmados de doenças , a utilização dessa técnica pode resultar na perda das características sazonais intrínsecas ao fenômeno estudado. A

sazonalidade, que é fundamental para capturar padrões periódicos (por exemplo, oscilações anuais em infecções respiratórias), pode ser distorcida ou suprimida pela transformação, prejudicando significativamente a qualidade e a utilidade prática das previsões.

A análise dos itens destacados evidencia as principais vantagens e limitações associadas a cada modelo considerado. O modelo ARIMA, embora apresente um tempo de execução reduzido tanto para a estimação quanto para a geração de previsões, mostra-se inadequado em contextos de baixas contagens, pois produz erros intervalares que incluem valores negativos. Por outro lado, modelos baseados em redes neurais demonstram excelente desempenho preditivo, com erros significativamente reduzidos mesmo em regiões próximas de zero; contudo, seu custo computacional é substancialmente elevado, tanto no processo de treinamento quanto na etapa de previsão.

Diante desse cenário, tornou-se evidente a necessidade da adoção de modelos que conciliem duas características essenciais: **rápida estimação/previsão e robusto desempenho em séries com valores próximos a zero**. Considerando essas exigências, foram realizados diversos estudos com o objetivo de identificar alternativas mais adequadas para a plataforma em desenvolvimento. A partir dessa análise criteriosa, concluiu-se que os denominados **modelos séries temporais de contagem** se mostravam particularmente promissores, beneficiando-se de uma sólida base teórica já consolidada na literatura estatística.

O tópico seguinte apresenta a introdução às classes de modelos de contagem selecionadas, destacando suas principais propriedades e justificativas para a sua escolha.

1.3 DADOS E MODELOS TEMPORAIS DE CONTAGEM

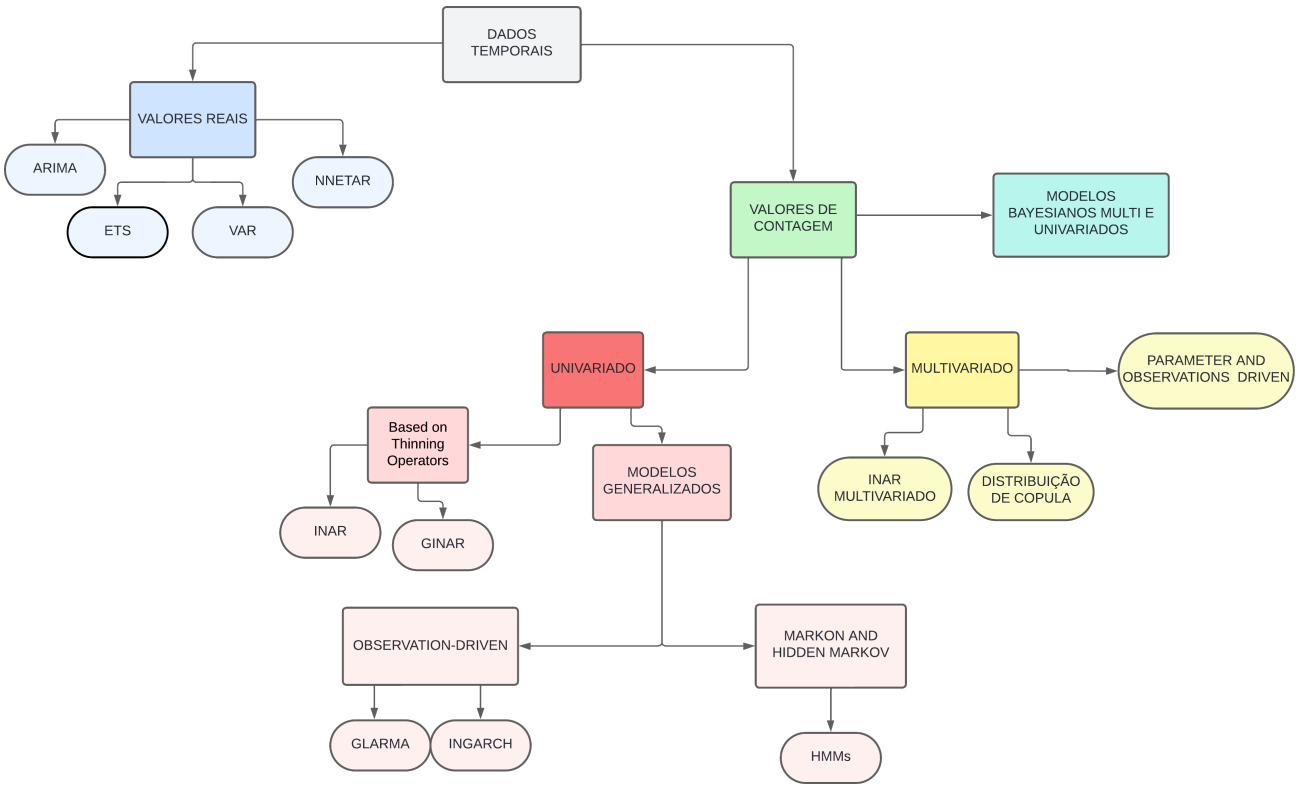
Dados de contagem são valores inteiros, ou seja são limitados em $(0, 1, 2, ..., \infty)$. Seus exemplos estão presentes em diversas áreas, desde o número de casos de certa doença, chegadas de clientes de um estabelecimento e a quantidade de bactérias de uma placa de petri. Problemas de contagem surgiram na área de regressão, onde pequisadores ao utilizarem os modelos de regressão linear simples, que assumiam que a variável dependente, mesmo sendo uma contagem, deveria seguir uma distribuição normal, observavam diversas quebras de suposições em relação aos resíduos desse modelo. Essa abordagem mostrou-se inadequada, já que a distribuição normal não é apropriada para modelar variáveis de contagem, que tendem a ter características diferentes, como assimetria e excesso de zeros. Nesse contexto, surgiu a necessidade de desenvolver modelos estatísticos que se adequassem melhor a esses tipos de dados. Como será descrito posteriormente, (Nelder e Wedderburn 1972) deram início a teoria dos Modelos Lineares Generalizados, os chamados MLGs, teoria essa que descreve a utilização de diferentes distribuições de pro-

babilidade para a variável resposta. Para dados de contagem, as distribuições comumente empregadas são a Poisson e Binomial Negativa, essas 2 distribuições serão detalhadamente estudadas ao longo do trabalho por conta de suas aplicações nos modelos temporais de contagem

As técnicas clássicas de séries temporais, tanto nos domínios de frequência quanto de tempo, são tipicamente baseadas em segunda ordem. Ou seja, a modelagem não vai além dos primeiros (média) e segundos (covariância) momentos. Como resultado, modelos Gaussianos, que são completamente caracterizados por seus dois primeiros momentos, se tornaram populares. Eventualmente, os pesquisadores buscaram mais, percebendo que os modelos Gaussianos frequentemente descreviam mal séries de contagem e outros valores discretos. A modelagem de séries temporais de contagem começou seriamente no final da década de 1970, quando extensões dos MLGs para séries temporais começaram a ser desenvolvidos

Após aproximadamente quatro décadas de pesquisa, os dados temporais de contagem têm sido abordados através de diversos conjuntos de classes e categorias de modelos. A vasta gama de modelos e suas respectivas classes podem ocasionar complexidade na análise. Portanto, o presente fluxograma foi elaborado com o propósito de categorizar os principais modelos de acordo com suas respectivas classes de estudo.

Classes de modelos de séries temporais



Elaborado pelo autor (2025)

Nesse trabalho, 2 modelos foram estudados detalhademente, o primeiro é o GLARMA, abreviação de ***Generalized Linear Autoregressive Moving Average*** ou em português **Modelo Autorregressivo de Médias Móveis Linear Generalizado**. Já o segundo é o INGARCH. abreviação de ***Integer-valued Generalized Autoregressive Conditional Heteroscedasticity*** ou em português **Modelo de Heterocedasticidade Condicional Autorregressivo Generalizado de Valores Inteiros**.

Tal destaque se dá pela sua implementação no pacote desenvolvido, tópico que será abordado posteriormente. Para evitar confusões de siglas, deve-se citar a existência do modelo IGARCH (Heterocedasticidade Condicional Autorregressivo Integrado), modelo esse que possui um parâmetro de integração indicando o número de diferenciações aplicada aos dados. Apesar de sua semelhança na nomenclatura e forma estatística com o modelo INGARCH, tal modelo não foi abordado nesse trabalho

Os modelos GLARMA e INGARCH são classificados como 'Orientados a Observações' dentre o grupo de modelos Generalizados. Fokianos define esse grupo como tendo uma das estruturas mais comuns e flexíveis para análise de séries temporais. Por exemplo, modelos ARMA e ARIMA podem ser representados como um modelo de espaço de estados linear, para o qual algoritmos de filtragem de Kalman, suavização e previsão podem ser implementados para calcular verossimilhanças Gaussianas, previsão de 1 passo, erros quadráticos médios de previsão e valores suavizados dos estados. Para séries temporais de contagens, o modelo de espaço de estados linear não é diretamente aplicável, pois a variável resposta assume valores discretos; uma forma mais geral do modelo de estado generalizado padrão é necessária. Por exemplo, um modelo de regressão Poisson assume

$$X_t|\alpha_t \sim Poisson(\exp^{\alpha_t}),$$

Onde $Poisson(\lambda)$ representa uma função densidade de probabilidade Poisson de média λ. Além da Poisson, outras distribuições de contagem podem ser utilizadas nesses modelos. Para isso, a distribuição deve pertencer a família exponencial de 1 parâmetro, tendo sua f.d.p fatorada em

$$\mathbb{P}\left(X_t = x_t \mid \alpha_t\right) = \exp\left\{\varphi\left(x_t\right) + \alpha_t x_t - A\left(\alpha_t\right)\right\}, \quad x_t = 0, 1, 2, \ldots, \tag{1.2}$$

onde $\phi(.)$ é uma função e $A(\alpha)$ é uma constante normalizadora que garante que a f.d.p resulte na unidade: $A(\alpha) = \log\left(\sum_{j=0}^{\infty} \exp\{\varphi(j) + \alpha j\}\right)$

Assim, outras distribuições comumente utilizadas são a Binomial Negativa e a Poisson Generalizada

Como descrito em (Davis et al. 2021), simples generalizações dos modelos autorregressivos são os modelos lineares e log-lineares. O modelo Poisson Autorregressivo Linear segue a forma

$$X_t \mid \mathcal{F}_{t-1} \sim \text{Poisson}\left(\lambda_t\right), \quad \lambda_t = d + \sum_{j=1}^{p} b_j X_{t-j}$$

onde $\mathcal{F}_{t-1} = \sigma(X_t, X_{t-1}, ...)$, d e $[b_j]_{j=1}^{p}$ são não negativos (essa característica garante que a média λ_t do processo seja não negativa)

Já o modelo log-linear tem forma análoga, como

$$v_t = d + \sum_{j=1}^{p} b_j \log\left(X_{t-j} + 1\right),$$

como descrito em (**?**) e (**?**) a função de ligação é dada por $v_t = log(\lambda_t)$, e diferente do modelo linear d e $[b_j]_{j=1}^{p}$ podem ser positivos ou negativos, porém devem satisfazer as condições de estacionariedade

Apesar da fácil generalização, Fokianos detalha que tais modelos possuem funções de autocovariâncias semelhantes ao modelos AR(p) e ARCH(p), (Autorregressivo e Heterocedasticidade Condicional Autorregressivo), onde essas se caracterizam como de "curta memória". Buscando modelos que contornam esse paradigma, (Bollerslev 1986) foi o primeiro a especificar o modelo GARCH (Heterocedastidicade Condicional Autorregressivo Generalizado), onde após sua primeira definição diversos autores ajudaram a desenvolver sua teoria para dados de contagem, como (Rydberg 2000), (Streett 2000), (Heinen 2003), (10), (13) , tendo inicialmente a seguinte estrutura

$$X_t \mid \mathcal{F}_{t-1} \sim \text{Poisson}\left(\lambda_t\right), \quad \lambda_t = d + \sum_{i=1}^{p} a_i \lambda_{t-i} + \sum_{j=1}^{q} b_j X_{t-j}$$

Dado a natureza de valores do conjunto numérico dos inteiros que o modelo possui ao se utilizar a distribuição Poisson, ele também é chamado de INGARCH (Heterocedastidicade Condicional Autorregressivo Generalizado de Valores Inteiros).

Analogamente, o modelo log-linear tem a seguinte estrutura

$$v_t = d + \sum_{i=1}^{p} a_i v_{t-i} + \sum_{j=1}^{q} b_j \log\left(X_{t-j} + 1\right),$$

com propriedades correspondente ao modelo linear. Não há restrições de sinal para os coeficientes e covariáveis podem ser facilmente incluidas, como exposto por (Fokianos 2011)

Demais tópicos, como diferentes funções de ligação são propostas por (Tjøstheim 2012), assim como métodos de estimação via quasi-verossimilhança discutidas por (13) e

(Christou e Fokianos 2014). Tais tópicos estão além do escopo desse trabalho, mas se mostram extremamentes relevantes no contexto estudado

Já para a família de modelos GLARMA, (Davis e Liu 2012) declara essa como uma das mais flexíveis e fáceis de estimação dentre os modelos do grupo 'Orientados

a Observação e Parâmetros'. O modelo base é construído a partir de $\{e_t\}$, onde o erro, assumindo a famiília exponencial de 1 parâmetro para as observações, gera uma sequência martingale de média 0 e variância unitária, definida por

$$e_t = \frac{X_T - B(\alpha_t)}{\sqrt{B\dot{t}(\alpha_t)}}$$

Como uma sequência martingale é não-correlacionada, tem-se que e_t é uma ruído branco de média 0 e variância unitária. A partir desse definição, um modelo GLARMA(p,q) é construído baseado em um modelo ARMA(p,q) via recursão de $\{e_t\}$.

De maneira específica, dado $\{W_t\}$ um modelo ARMA(p,q) inversível, ele segue a seguinte forma

$$W_t = \sum_{i=1}^{p} \phi_i \alpha_{t-i} + \sum_{j=1}^{q} \theta_j e_{t-j} + e_t,$$

onde α_t é chamado como o melhor preditor linear de W_t dado um passado infinito $\{W_s, s < t\}$

1.4 SOFTWARE R E PACOTES ESTATÍSTICOS

Durante o trabalho será descrito como os modelos temporais explicitados no tópico anterior foram implementados em um pacote de funções, onde que para isso o software R foi escolhido.

"O R é um ambiente de *software* livre para computação estatística e gráficos. Compila e roda em uma ampla variedade de plataformas UNIX, Windows e MacOS" (R Project, 2021). Atualmente, porém, o R se tornou muito mais do que um simples ambiente de software estatístico e se estabeleceu como uma das principais linguagens de programação de alto nível com potencial para lidar com problemas de modelagem estatística, contando com uma série de ferramentas modernas que permitem inclusive o desenvolvimento de aplicativos para a internet e manipulação de grandes bases de dados. O uso do R tem se dado principalmente por estatísticos, matemáticos, programadores e cientistas contemporâneos no geral. As próprias universidades do Brasil e do mundo têm se esforçado para ensinar e incentivar seus alunos a trabalharem com a ferramenta e o motivo desta operação tem sido a necessidade computacional no desenvolvimento de estudos científicos. O resultado deste necessário esforço é que cada vez mais novos pacotes capazes de lidarem com problemas cada vez mais complexos estão sendo desenvolvidos em um intervalo de tempo cada vez menor, possibilitando que os demais usuários da comunidade usufruam deste material e apliquem em suas respectivas áreas de interesse (Pacheco 2021)

A criação de um pacote passa por diversas etapas, indo da sua motivação inicial, diferentes maneiras de estruturação de código até a forma de disponibilização do produto final para o público. Assim, para seu desenvolvimento, o pacote construído neste trabalho se baseou fortemente na filosofia exposta por (Wickham e Bryan 2023) no livro R Packages (2e) "Este livro defende nossa filosofia de desenvolvimento de pacotes: tudo que pode ser automatizado, deve ser automatizado. Faça o mínimo possível manualmente. Faça o máximo possível com funções."

A escrita de uma função não é apenas um processo de generalização, é também um processo de automatização, e portanto toda sua estrutura deve ser analisada com cuidado. Seja o número de argumentos que ela deve ter, a ordem no qual esses argumentos estão dispostos, como também seu objeto de retorno, todos os passos desse algoritmo deve ser minuciosamente dissecado buscando facilitar sua utilização por parte do usuário. Para isso a forma de escrita e organizações de funções teve como fundamento o conceito de programação funcional descrito por (Wickham 2014) e por (Pacheco 2021)

Assim este trabalho buscou ir além das definições matemáticas dos modelos implementados, ao também apresentar o pensamento computacional empregado na construção do pacote.

1.5 OBJETIVOS E ORGANIZAÇÃO

O presente trabalho está estruturado de forma a conduzir o leitor desde a fundamentação matemática dos modelos de séries temporais de contagem até a aplicação prática em dados reais, passando por aspectos de implementação computacional, automatização de processos e perspectivas futuras do pacote desenvolvido.

No Capítulo 2, são apresentadas as bases matemáticas e estatísticas dos modelos GLARMA e INGARCH . Este capítulo estabelece os fundamentos teóricos necessários para compreender a estrutura e as propriedades desses modelos, fornecendo a sustentação formal para as etapas posteriores.

O Capítulo 3 discute o papel dos pacotes estatísticos no ambiente R, com destaque para aqueles voltados à modelagem de séries temporais. Nesse contexto, é introduzido o pacote `fableCount`, desenvolvido no âmbito deste trabalho, detalhando sua estrutura interna, a implementação computacional dos modelos GLARMA e INGARCH, bem como sua identidade visual. Esse capítulo conecta a teoria do Capítulo 2 ao processo de tradução prática em código.

No Capítulo 4, são abordados os métodos de modelagem automatizada, divididos em três eixos: a seleção automática de distribuições, a seleção automática de ordens de parâmetros e a busca pelo melhor modelo com foco em previsão. São discutidas, também, as métricas de avaliação de desempenho, e os métodos de avaliação para modelos com

foco em previsão, como o *Time Series Cross Validation* e o *Out-of-Sample*, que orientam a seleção do modelo preditivo mais adequado.

O Capítulo 5 traz a aplicação empírica do pacote desenvolvido em dados epidemiológicos da COVID-19. Nessa etapa, demonstra-se o desempenho dos modelos de contagem frente a alternativas clássicas como ARIMA e NNETAR, evidenciando ganhos relevantes em termos de acurácia e desempenho computacional, se alinhando com as hipóteses desenvolvidas durante o capitulo

Já o Capítulo 6 apresenta métricas de utilização do pacote, número de usuários ativos e demais indicadores de impacto, além de discutir os planos futuros para sua expansão e aprimoramento. Essa seção conecta os resultados obtidos à relevância prática e à sustentabilidade do projeto.

Por fim, o Capítulo 7 sintetiza as principais contribuições do trabalho, destacando avanços, limitações e potenciais desdobramentos para pesquisas futuras no campo das séries temporais de contagem.

2 DEFINIÇÕES DOS MODELO ORIENTADOS A OBSERVAÇÕES

Conforme detalhado na seção 1.3, o pacote desenvolvido concentrou-se exclusivamente na implementação dos modelos classificados como "Orientados a Observações". A implementação desses modelos é subdividida em duas partes distintas: a definição matemática e a implementação computacional. Dessa forma, nesta seção, foram delineados os conceitos fundamentais relacionados às definições matemáticas e estatísticas de cada modelo. Posteriormente, na seção 3, intitulada "Pacote", foram abordadas as questões relacionadas à implementação computacional, fornecendo detalhes sobre como os modelos foram traduzidos em código e integrados ao pacote.

Uma importante consideração deve ser fixada sobre como os nomes dos modelo foram utilizados. Ao nos referirmos aos modelos de séries temporais, estamos abordando uma classe abrangente de modelos. Tomando como exemplo a classe ARMA, é possível identificar suas diversas variações como: modelos sazonais, modelos com parâmetros de diferenciação e aqueles que incluem covariáveis, denominados SARMA, ARIMA e ARMAX, respectivamente. Adicionalmente, é possível combinar diferentes modelos dentro de cada classe, como é o caso do SARIMAX, um modelo Sazonal Autorregressivo Integrado de Médias Móveis com Variáveis Exógenas, que é uma fusão dos três tipos mencionados anteriormente. Com o intuito de simplificar a comunicação e facilitar a compreensão do trabalho, os modelos GLARMA e INGARCH **serão sempre referidos por seus nomes originais, enquanto suas variações serão acompanhadas de sufixos, indicando a presença de parcelas sazonais ou covariáveis.** Essa abordagem visa aprimorar a clareza e a consistência na exposição dos conceitos e resultados apresentados neste estudo.

2.1 MODELOS LINEARES GENERALIZADOS

A classe dos modelos orientados a observações são originados daqueles como classificados como Modelos Lineares Generalizados, onde os modelos de cadeia oculta de Markov formam ou outro subgrupo. Os Modelos Lineares Generalizados (MLG) propostos por (Nelder e Wedderburn 1972) são uma extensão dos modelos lineares clássicos, onde o trabalho proposto pelos autores buscou especificar outras distribuições para a variável resposta

Os modelos lineares usuais assumem que sua variável resposta assume distribuição Normal, com parâmetros de locação e escala constantes, tal suposição se mostra robusta em diversos contextos, onde assumir tal distribuição facilita interpretações do modelo, testes de significância e certas propriedades exclusivas de uma distribuição Gaussiana. Porém, em contextos como dados de contagem, binários, e limitados na reta real, utilizar tal modelo se mostra inviável e errôneo. Assim, os MLGs surgem como ferramenta para esses casos. Nesse contexto, a teoria desenvolvida por (Nelder e Wedderburn 1972)

especificou demais distribuições de probabilidade para a váriavel resposta.

Dado uma amostra com n observações independentes, $\mathbf{X}$ uma matriz com $p+1$ colunas e $\mathbf{y}$ um vetor de observações amostrado de Y, são definidos os 3 componentes de um MLG

- Y tem distribuição probabilística como membro da Família Exponencial de distribuições, com uma função de probabilidade ou função densidade de probabilidade, para variáveis aleatórias discretas e contínuas respectivamente

$$\mathbb{P}\left(X_t = x_t \mid \alpha_t\right) = \exp\left\{\varphi\left(x_t\right) + \alpha_t x_t - A\left(\alpha_t\right)\right\}, \quad x_t = 0, 1, 2, \ldots, \tag{2.1}$$

  onde $\varphi(.)$ é uma função e $A(\alpha)$ é uma constante normalizadora que garante que a f.d.p resulte na unidade: $A(\alpha) = \log\left(\sum_{j=0}^{\infty} \exp\{\varphi(j) + \alpha j\}\right)$

$$(y_i) = \mu_i = b \tag{2.2}$$

$$Var(y_i) = \sigma^2 \tag{2.3}$$

- A matriz $\mathbf{X}$ de covariáveis relacionadas no chamado preditor linear na forma

$$\eta_i = \boldsymbol{X_i^T \beta} \tag{2.4}$$

- Uma Função de Ligação monótona (inversível) e diferenciável $g(.)$, que liga o preditor linear η_i à média de Y onde escrevemos

$$g(\mu_i) = \eta_i \tag{2.5}$$

Para variáveis de contagem, as distribuições Poisson e Binomial Negativa são comulmente utilizadas. A distribuição de Poisson é definida para uma variável aleatória discreta Y que representa o número de eventos ocorrendo em um intervalo fixo. A função de probabilidade é dada por

$$f_Y(y|\lambda) = \frac{\lambda^y \exp^{-\lambda}}{y!} \tag{2.6}$$

onde λ é o parâmetro da taxa que representa o número médio de eventos no intervalo, y é o número de eventos. A distribuição Poisson possui os primeiro e segundo momentos iguais, ou seja

$$E(Y) = Var(Y) = \lambda \tag{2.7}$$

Em sua forma na família exponencial, a distribuição pode ser fatorada em

$$f_Y(y|\lambda) = exp\ (y\ log(\lambda) - \lambda + (-log\ y!)) \tag{2.8}$$

Sua função de verossilhança e log-verossimulhança são dadas por

$$L(\lambda, \tilde{y}) = \sum_{j=1}^{n} \exp(-\lambda)\frac{1}{y_j!}\lambda^{y_j} \tag{2.9}$$

$$l(\lambda, \tilde{y}) = -n\lambda - \sum_{j=1}^{n} ln(y_j!) + ln(\lambda)\sum_{j=1}^{n} y_j \tag{2.10}$$

Já a Binomial Negativa é uma distribuição de probabilidade discreta que pode ser usada para modelar o número de falhas até que um número fixo de sucessos seja alcançado em uma sequência de experimentos de Bernoulli independentes. Esta distribuição é frequentemente utilizada em situações onde os dados apresentam sobredispersão, ou seja, quando a variância é maior do que a média, algo que não pode ser adequadamente modelado pela distribuição de Poisson. Sua função de probabilidade é dada por

$$f_Y(y|r,p) = \binom{r+y-1}{y} p^r(1-p)^y \tag{2.11}$$

onde y representa o número de falhas até que ocorra o r-ésimo sucesso, r representa o número de sucessos desejados e p é a probabilidade de sucesso de cada experimento Bernoulli

Diferentemente da distribuição Poisson, a Binomial Negativa possui esperança e variância distintas

$$E(Y) = \frac{r(1-p)}{p} \tag{2.12}$$

$$Var(Y) = \frac{r(1-p)}{p^2} \tag{2.13}$$

Em sua forma na família exponencial, a distribuição pode ser fatorada em

$$f_Y(y|r,p) = \binom{r+y-1}{y} \exp(y\ln(p) + r\ln(1-p)) \tag{2.14}$$

Sua função de verossimilhança e log-verossimilhança são dadas por

$$L(r,p|y) = \sum_{j=1}^{n} \binom{r+y_i-1}{y_i} p^r(1-p)_i^y \tag{2.15}$$

$$l(r,p|y) = \sum_{j=1}^{n} \ln \binom{r+y_i-1}{y_i} + r\ln(p) + y_i \ln(1-p) \tag{2.16}$$

Normalmente, a função identidade e a função log são utilizadas como função de ligação para ambas as distribuições descritas

Utilizando função identidade, a média é diretamente modelada pelo preditor linear, dado por

$$\mu = \eta = \sum_{j=1}^{d} x_j \beta_d = \boldsymbol{x'}\beta \tag{2.17}$$

Já o modelo com função de ligação log tem-se a seguinte estrutura

$$log(\mu) = \eta = \sum_{j=1}^{d} x_j \beta_d = \boldsymbol{x'}\beta \tag{2.18}$$

Com os MLGs devidamente introduzidos, podemos então nos aprofundar nos 2 modelos que são o foco do trabalho: GLARMA e INGARCH

2.2 GLARMA - AUTORREGRESSIVO DE MÉDIA MÓVEIS LINEAR GENERALIZADO

Dado uma série temporal dada como $\boldsymbol{Y}_t : t \in \mathbb{N}$ e um vetor k-dimensional de de covariáveis dado como $\boldsymbol{X}_t : t \in \mathbb{N}$. Denota-se $\mathcal{F}_t = \{Y_s : s < t, x_s : s \leq t\}$ como a informação em tempo anterior para a variável resposta e a informação em um tempo anterior e presente para as covariáveis. Em geral a distribuição condicional de Y_t dado $\mathcal{F}_t$ é dado através da sua fatoração na forma da família exponencial

$$f\left(y_t \mid W_t\right) = \exp\left\{y_t W_t - a_t b\left(W_t\right) + c_t\right\}, \tag{2.19}$$

onde a_t e c_t são sequências de constantes possivelmente dependendo das observações y_t. A informação de $\mathcal{F}_t$ é resumida na variável W_t

Como descrito por (Benjamin e Stasinopoulos 1998), modelos orientados a observações podem assumir diversas formas. O trabalho teve como foco a utilização para o modelo GLARMA onde a utilização do vetor de estados em (2.1) é forma geral

$$W_t = x_t^T \beta + O_t + Z_t. \tag{2.20}$$

Além dos parâmetros da regressão β, o termo O_t pode ser incluído como *offset* do modelo. O *offset* é um termo utilizado para ajustar o modelo em diferenças conhecidas entre observações, permitindo a modelagem de taxas relativas sem estimar novos parâmetros.

Para a dependência temporal descrito no processo de estado de (2.20) de W_t, o termo Z_t é introduzido no processo de estado como uma recursão de médias móveis autorregressivas, de forma

$$Z_t = \sum_{i=1}^{p} \phi_i (Z_{t-i} + e_{t-i}) + \sum_{i=1}^{q} \theta_i e_{t-i}, \tag{2.21}$$

no qual, seus resíduos preditivos são definidos como

$$e_t = \frac{Y_t - \mu_t}{\nu_t}, \tag{2.22}$$

tendo ν_t como um termo chamado de função normalizadora ou função de escala. Esse termo se refere a uma transformação que ajusta ou redimensiona os resíduos preditivos com base em uma medida de dispersão, como o desvio padrão.

É importante destacar que esses tipos de resíduos são uma sequência martingale, e portanto possuem média 0 e são independentes. Quando ν_t é ajustado ao desvio padrão condicional de Y_t e e_t também possui variância unitária, temos que esses são um ruído branco fracamente estacionários (Dunsmuir e Scott 2015)

Além da forma descrita em (2.21), Z_t pode ser escrito via combinações lineares dos resíduos preditivos e_t descritos em (2.22), isto é,

$$Z_t = \sum_{j=1}^{\infty} \gamma_j e_{t-j}, \tag{2.23}$$

Para a parametrização da parcela de peso de média móveis infinitas γ_j na equação, configurar que sejam representados como os coeficientes em um filtro autorregressivo de médias móveis. Dado por

$$\sum_{j=1}^{\infty} \gamma_j \zeta^j = \theta(\zeta)/\phi(\zeta) - 1 \tag{2.24}$$

onde $\phi(\zeta) = 1 - \phi_1\zeta - ... - \phi_p\zeta^p$ e $\theta(\zeta) = 1 + \theta_1\zeta + ... + \theta_q\zeta^q$ são os respectivos polinomios autorregressivos e de médias móveis do filtro ARMA.

Como descrito em (6), ao se definir $\{Z_t\}$ dessa maneira, pode se enxergar ele como o melhor predito linear de um processo ARMA estacionário e invertível, com ruído dado pela sequência de $\{e_t\}$ de desvios padronizados das respostas de contagem a partir de sua média condicional, dado os valores passados para a variável resposta e valores passados e atuais das covariáveis.

Continuando as definições do modelo a respeito de seus resíduos preditivos, dado por $e_t = \frac{Y_t - \mu_t}{\nu_t}$. Seja $\nu(W_t)$ uma função de escala, é possível construir diversos tipos de resíduos a partir dessa definição

Função de Escala de Pearson

Seja $\nu_t = \nu_{P,t}$, onde

$$\nu_{P,t} = [a_t b(W_T)]^{0.5}$$

Tal função de escala resulta no resíduo do tipo Pearson

Função de Escala Score ype

Se baseia no trabalho de (5), onde se realiza a substituição do desvio padrão condicional pela variância condicional, dado por

$$\nu_{S,t} = a_t b(W_T)$$

Função de Escala Identidade

Se baseia no trabalho de (Wang e Li 2011), que remete as características de um modelo BARMA (ARMA Binário), onde não se considera uma função de escala, dado por

$$\nu_{I,t} = 1$$

Cada tipo de função de escala e consequentemente de resíduos, possuem resultados distintos nos modelos GLARMA Poisson e GLARMA Binomial Negativa, antes de citarmos tais diferenças, tais distribuições para respostas devem ser explicitadas

2.2.1 DISTRIBUIÇÕES PARA A VARIÁVEL RESPOSTA

O pacote desenvolvido oferece ao usuário 2 tipos de função distribuição de probabilidade para o usuário utilizar para a variável resposta, sendo elas distribuição de Poisson e distribuição Binomial Negativa

Para a distribuição de Poisson, tem-se que $\alpha_t = 1$, $b(W_t) = exp(W_t)$, $c_t = -log(y_t)$ e a função de ligação utilizada é a canônica dado por $g(\mu) = ln(\mu)$

Para a distribuição Binomial Negativa, tem-se que $\mu_t = exp(W_t)$, onde a seguinte parametrização é utilizada no pacote

$$f\left(y_t \mid W_t, \alpha\right) = \frac{\Gamma\left(\alpha + y_t\right)}{\Gamma(\alpha)\Gamma\left(y_t + 1\right)} \left[\frac{\alpha}{\alpha + \mu_t}\right]^{\alpha} \left[\frac{\mu_t}{\alpha + \mu_t}\right]^{y_t}. \tag{2.25}$$

É importante destacar que $\mu_t = exp(W_t)$ e que $\sigma_t^2 = \mu + \mu^2/\alpha$. Para $\alpha \sim \infty$ a distribuição Binomial Negativa converge para uma Poisson. Note que, se α é conhecido, tal densidade pode ser fatorada em uma pertencente a família exponencial uni-paramétrica

Tais opções são disponibilizadas a depender do grau de dispersão dos dados trabalhados. A distribuição Poisson é mais simples de se trabalhar por conta de apresentar

apenas 1 parâmetro, dado por $\lambda_t = exp(W_t)$, onde portanto a estimação de apenas 1 parâmetro é necessária. Apesar de tal comodidade, tal distribuição assume igualdade de média e variância, no qual tem-se que $\lambda_t = \mu_t = \sigma_t^2$. Assim para dados que apresentam o fenômeno de sobredispersão, tal distribuição apresenta perda de desempenho. Nesse contexto a utilização da distribuição Binomial Negativa se mostra mais interessante, ao possuir funções distintas para sua esperança e variância.

A escolha da melhor distribuição será novamente abordade no tópico 4 - 'Algoritmos para Modelagem Automatizada'.

2.2.2 ESCOLHA DOS TERMOS AR E MA

A escolha das defasagens adequadas para os componentes AR e MA no modelo GLARMA costuma ser consideravelmente mais difícil do que em séries Gaussianas. Nesses casos, os resíduos obtidos a partir de ajustes por mínimos quadrados podem fornecer informações bastante úteis sobre a estrutura de dependência serial, especialmente por meio das funções de autocorrelação e autocorrelação parcial.

No entanto, ao contrário dos resíduos da regressão por mínimos quadrados que frequentemente ajudam a identificar a estrutura do modelo de dependência serial em respostas contínuas , no caso de respostas com valores discretos, os resíduos do ajuste de um modelo GLM geralmente não fornecem boas orientações para a escolha dos parâmetros p e q necessários para especificar o modelo GLARMA. Isso é ainda mais evidente quando a dependência serial é fraca ou moderada.

Assim como a escolha da melhor distribuição para o modelo, a escolha do número de defasagens adequadas, tanto para o termo AR quanto para o MA, serão discutidos novamente no tópico 4 'Algoritmos para Modelagem Automatizada'.

2.2.3 CÁLCULO DE PREVISÕES

Em comparação com modelos ARMA e ARMAX, os métodos para previsão de valores futuros cálculados a partir de um GLARMA possuem um complexidade adicional dado a estrutura condicional do modelo.

Para previsões um passo a frente *(one-step ahead forecast)* devemos considerar a especificação condicional de Y_t. No entanto, para previsões de múltiplos passos à frente, essa formulação condicional implica que todos os possíveis caminhos futuros da amostra ao longo do horizonte de previsão precisam ser considerados, seja teoricamente ou por meio de simulação.

Para previsões um passo à frente, denotadas por Y_{t+1}, onde n é o índice máximo para as amostras de treinamento do modelo, temos a seguinte estrutura

$$\hat{Z}_{t+1} = \sum_{j=1}^{p} \hat{\phi}_j \left(\hat{Z}_{t+1-j} + \hat{e}_{t+1-j} \right) + \sum_{j=1}^{q} \hat{\theta}_j \hat{e}_{t+1-j}. \tag{2.26}$$

Note que $\hat{Z}_{t+1}$ pode ser estimado usando apenas os valores de $\hat{Z}_t$ e de $\hat{e}_t$

Já para o caso de previsões múltiplos passos à frente, temos a seguinte estrutura

$$f(y_{n+L} \mid \mathcal{F}_n) = \sum_{y_{n+1}} \cdots \sum_{y_{n+L-1}} \prod_{j=1}^{L} f(Y_{n+j} \mid \mathcal{F}_{n+j})) \tag{2.27}$$

Onde definimos $\mathcal{F}_n$ como uma função que possui o conjunto de informações utilizadas no modelo como: valores regressores $x_t, ..., x_{t+j}$ bem como os valores passados da própria série temporal $y_{t+1}, ..., y_{t+j-1}$. Os termos que estão dentro do somatório irão crescer de maneira exponencial dado o número de previsões a frente L

Apesar de apresentar uma complexidade alta para estimação e previsão em um ambiente computacional, o framework **glarma** utilizado no ambiente R, encapsula todas as funções necessárias para estimação, previsões e os métodos computacionais de otimização para ambos os objetivos. Facilitando assim, a implementação desse modelo no novo framework criado **fablecount**

2.3 INGARCH - HETEROCEDEDASTICIDADE CONDICIONAL AUTORREGRESSIVO GENERALIZADO

Dada uma série temporal dada como $\boldsymbol{Y}_t : t \in \mathbb{N}$ e um vetor r-dimensional de covariáveis dado como $\boldsymbol{X}_t : t \in \mathbb{N}$, estamos interessados em modelar a média condicional do processo dado por $E(Y_t|F_{t-1}) = \lambda_t$. Os modelos tem forma geral dado por

$$g(\lambda_t) = \beta_0 + \sum_{k=1}^{p} \beta_k \tilde{g}(Y_{t-i_k}) + \sum_{\ell=1}^{q} \alpha_\ell g(\lambda_{t-j_\ell}) + \boldsymbol{\eta}^\top \boldsymbol{X}_t, \tag{2.28}$$

onde $g : \mathbb{R} \to \mathbb{R}^+$ é uma função de ligação e $\tilde{g} : \mathbb{N}_0 \to \mathbb{R}$ é uma função de transformação. $\boldsymbol{\eta} = (\eta_1, \eta_2, ..., \eta_r)^T$ é o vetor de efeitos das covariáveis $\boldsymbol{X}_t$. Na teoria dos MLGs e sua termologia, chamamos $\nu_t = g(\lambda_t)$ de preditor linear.

Para a escrita de um modelo INGARCH(p, q), defini-se os parametros autorregressivos p e de médias móveis q. A ordem p refere-se ao número de defasagens das observações utilizadas como autorregresivas $Y_{t-1}, Y_{t-2}, Y_{t-p}$, já a ordem q refere-se ao número de defasagens das médias condicionais $\lambda_{t-1}, \lambda_{t-2}, \lambda_{t-q}$. Além disso, um INGARCH com sazonalidade estocástica é escrito como INGARCH$(p, q)(P, Q)_{[m]}$, onde P representa a ordem autorregressiva sazonal, Q a ordem de médias móveis sazonais e m o indice de sazonalidade

Um exemplo pode ser escrito de um modelo INGARCH(p, q) com função de ligação e tranformação identidade, $g(x) = \tilde{g}(x) = x$, e efeito das covariáveis igual a 0, $\boldsymbol{\eta} = \mathbf{0}$

$$\lambda_t = \beta_0 + \sum_{k=1}^{p} \beta_k Y_{t-k} + \sum_{\ell=1}^{q} \alpha_\ell \lambda_{t-\ell} \tag{2.29}$$

Por si só, sem especificação de distribuição para Y_t, o modelo acima é descrito como um GARCH(p, q), na qual a partir da especificação de uma distribuição de contagem como Poisson ou Binomial Negativa, tal modelo passa a se chamar GARCH de Valores Inteiros, ou apenas INGARCH. Em casos de especificação da distribuição Poisson, tal modelo é chamado por alguns autores de *Autorregressive Conditional Poisson*(ACP)

Além da função identidade, a função de ligação log é extremamente utilizada para ajustes de modelos log-lineares. Assim, dado uma função de ligação $g(x) = log(x)$, uma função de transformação $\tilde{g}(x) = log(x + 1)$, temos o seguinte modelo INGARCH(p, q)

$$g(\lambda_t) = \beta_0 + \sum_{k=1}^{p} \beta_k \log(Y_{t-k} + 1) + \sum_{\ell=1}^{q} \alpha_\ell \nu_{t-\ell}. \tag{2.30}$$

Além das escolhas para as funções de ligação e transformação, é necessário a utilização de alguma distribuição de probabilidade de contagem, onde no próximo tópico os modelos com distribuição Poisson(λ_t) e Binomial Negativa(λ_t, θ) foram estudados

2.3.1 DISTRIBUIÇÕES PARA A VARIÁVEL RESPOSTA

Assim como o modelo GLARMA, o modelo INGARCH utiliza como base para sua estimação as distribuições de probabilidade Poisson ou Binomial Negativa

A distribuição de Poisson é comumente usada para modelar a taxa de eventos aleatórios que ocorrem em algum intervalo de tempo fixo. Se assumirmos que λ denota a taxa de chegadas, então a distribuição da variável aleatória Y , que denota o número de chegadas em um intervalo de tempo fixo, segue a distribuição de Poisson com função densidade de probabilidade (Fokianos 2012)

Para modelos paramétricos de contagem como o caso do INGARCH, é necessário a utilização de alguma distribuição de contagem. Assim, nesse contexto, a distribuição mais simples.

O modelo faz a seguinte suposição:

$$\mathrm{P}(Y_t = y \mid \mathcal{F}_{t-1}) = \frac{\lambda_t^y \exp(-\lambda_t)}{y!}, \quad y = 0, 1, \ldots \tag{2.31}$$

Além disso, dado uma variável aleatória $X \sim Poisson(\lambda)$, tem-se que

$$E(X) = Var(X) = \lambda.$$

Esse resultado segue para os casos temporais, onde portanto:

$$E(Y_t|\mathcal{F}_{t-1}) = Var(Y_t|\mathcal{F}_{t-1}) = \lambda_t.$$

Para a estimação do modelo com função de ligação e função de transformação identidade, o espaço paramétrico com covariáveis é dado por

$$\Theta = \left\{ \boldsymbol{\theta} \in \mathbb{R}^{p+q+r+1} : \beta_0 > 0, \beta_1, \ldots, \beta_p, \alpha_1, \ldots, \alpha_q, \eta_1, \ldots, \eta_r \geqslant 0, \sum_{k=1}^{p} \beta_k + \sum_{\ell=1}^{q} \alpha_\ell < 1 \right\} \tag{2.32}$$

Já para modelos log-lineares, o espaço paramétrico com covariáveis é dado por

$$\Theta = \left\{ \boldsymbol{\theta} \in \mathbb{R}^{p+q+r+1} : |\beta_1|, \ldots, |\beta_p|, |\alpha_1|, \ldots, |\alpha_q| < 1, \left| \sum_{k=1}^{p} \beta_k + \sum_{\ell=1}^{q} \alpha_\ell \right| < 1 \right\} \tag{2.33}$$

A utilização dessa distribuição torna a estimação do modelo menos complexo, dado que apenas 1 parâmetro (λ_t) deve ser estimado. Porém em casos no qual a variância é significativamente maior que a média, fenômeno esse chamado de sobredispersão, a utilização da distribuição Binomial Negativa pode trazer ganhos ao modelo

A distribuição Binomial Negativa permite que a variância condicional do modelo seja maior que média λ_t. Como definido por (Christou e Fokianos 2014), o modelo assume que $Y_t \mid \mathcal{F}_{t-1} \sim \text{NegBin}(\lambda_t, \phi)$, onde a distribuição possui 2 parâmetros: o primeiro λ_t é média e o segundo ϕ modela a dispersão. Isso é

$$P(Y_t = y \mid \mathcal{F}_{t-1}) = \frac{\Gamma(\phi + y)}{\Gamma(y+1)\Gamma(\phi)} \left(\frac{\phi}{\phi + \lambda_t} \right)^{\phi} \left(\frac{\lambda_t}{\phi + \lambda_t} \right)^{y}, \quad y = 0, 1, \ldots \tag{2.34}$$

Para esse caso, $Var(Y_t \mid \mathcal{F}_{t-1})$, ou seja, a variância condicional aumenta quadraticamente em relação a λ_t. Vale destacar que a distribuição Poisson é um caso onde $\phi \to \infty$

2.3.2 ESCOLHA DOS TERMOS AR E MA

Como já descrito para o modelo GLARMA, encontrar o valor ótimo para as ordens autorregressivas e de médias móveis não é uma tarefa trivial.

Buscando contornar as dificuldades ligados a esse tema o tópico 4 "Algoritmos para Modelagem Automatizada" apresenta métodos que facilitam a escolha dos valores das ordens autorregressivas e de médias móveis

2.3.3 CÁLCULO DE PREVISÕES

Em relação ao erro quadrático médio, para previsões 1 passo a frente, o melhor estimador $\hat{Y}_{n+1}$ dado $\mathcal{F}_n$ é dado pela média condicional de λ_{n+1}, como descrito em

$$g(\lambda_t) = \beta_0 + \sum_{k=1}^{p} \beta_k \log(Y_{t-k} + 1) + \sum_{\ell=1}^{q} \alpha_\ell \nu_{t-\ell}. \tag{2.35}$$

Já para previsões múltiplos passos a frentes, previsões essas chamadas de "previsões h passos a frente", onde h representa qual é o horizonte de previsão, o melhor estimador $\hat{Y}_{n+h}$ é obtido através da chamada de previsões 1 passo a frente recursivamente, onde os valores desconhecidos $Y_{n+1}, \ldots, Y_{n+h-1}$ são obtidos por suas respectivas previsões 1 passo a frente. Vale destacar que a distribuição de previsões h passos a frente não é conhecida de forma analítica, mas pode pode ser aproximada por métodos numéricos. Nesse trabalho utilizamos um bootstrap paramétrico para isso, permitindo a construção de intervalos de confiança para as estimativas calculadas

O pacote construído encapsula o pacote `tscount` para estimação dos modelos e cálculo de previsões. Dessa forma, reciclamos métodos numéricos de difícil construção que já foram desenvolvidos e focamos em na melhor forma de disponibilizar esses métodos para os usuários.

3 PACOTE

3.1 PACOTES DE SÉRIES TEMPORAIS NO R

Durante as décadas de 1960 e 1970, os principais modelos estatísticos de séries temporais foram desenvolvidos. Diferentemente dos métodos anteriores aos anos 1960, como o método de Delphi, os modelos de Suavização Exponencial e Autorregressivo de Médias Móveis exigem um processo intensivo de avaliação das funções de autocovariância, seleção da ordem dos parâmetros e estimação desses parâmetros por meio de métodos como máxima verossimilhança, mínimos quadrados, entre outros.

Consequentemente, esses métodos apresentam dificuldades de aplicação sem o auxílio de um computador. Os primeiros softwares a oferecerem funcionalidades para análise de séries temporais foram o Autoreg, que lançou suas funções em 1960, o SPSS, que as introduziu a partir de 1968, e o SAS, em 1970.

Embora pioneiros em seus respectivos campos, esses softwares não conseguiram ganhar grande popularidade além do ambiente comercial, principalmente devido à sua disponibilidade exclusiva por meio de assinatura ou compra de licenças. Essa abordagem acabou afastando estudantes autônomos, acadêmicos e aqueles sem acesso a recursos financeiros para investir nessas ferramentas. No entanto, com o avanço da tecnologia e o surgimento da comunidade de código aberto, novas soluções surgiram, oferecendo acesso gratuito a ferramentas poderosas de análise de séries temporais. Linguagens como R, por exemplo, rapidamente se tornaram populares devido à sua gratuidade, vasta gama de pacotes e suporte ativo da comunidade. O R oferece uma ampla variedade de funções para modelagem, análise e visualização de séries temporais, além de permitir maior flexibilidade e customização de análises conforme as necessidades do usuário. Sua comunidade ativa e colaborativa também proporciona suporte e recursos adicionais, tornando-o uma escolha atraente para estudantes, acadêmicos e profissionais em todo o mundo.

Com sua primeira versão lançada em 1993, o R se tornou uma das linguagens de programação mais populares para análise de dados, impulsionado em grande parte por sua natureza open-source. Essa característica permite que qualquer pessoa contribua para o desenvolvimento da linguagem e crie novos pacotes para atender a necessidades específicas. A comunidade de usuários do R é extremamente ativa, o que resulta em uma vasta gama de pacotes disponíveis no Comprehensive R Archive Network (CRAN), repositório oficial de pacotes da linguagem. Abrangendo desde a manipulação de dados e visualização até modelagem estatística avançada, aprendizado de máquina e ciência de dados, sua versatilidade, combinada com a sua natureza open-source, a torna uma ferramenta indispensável para pesquisadores, cientistas de dados e analistas de negócios em todo o mundo.

O R oferece uma vasta gama de pacotes especializados em análise de séries tempo-

rais, cada um com suas próprias características e funcionalidades. Entre os mais populares, destacam-se o `forecast`, conhecido por sua ampla variedade de modelos, o `zoo`, que além de séries temporais, oferece ferramentas para manipulação de objetos indexados no tempo, e o `TSA`, voltado para análises estatísticas mais tradicionais.

O `forecast` se caracteriza como o pacote mais famoso para o nicho de séries temporais, possuindo maior número de downloads dentre os pacotes dessa área, ultrapassando a marca de mais de 20 milhões. Desenvolvido por Rob Hyndman, George Athanasopoulos e outros, o pacote trás modelos como ARIMA, ETS, ARIFMA, BATS, TBATS e outros. Apesar de sua popularidade, o pacote lançado em 2008 não era capaz de suportar métodos para modelos hierárquicos, de regressão dinâmica e métodos de reconciliação. Ao capturar a estrutura hierárquica presente em muitos conjuntos de dados, como vendas por produto e região, os modelos hierárquicos permitem que informações de níveis superiores sejam utilizadas para melhorar as previsões em níveis inferiores. A regressão dinâmica, por sua vez, modela a evolução das relações entre variáveis ao longo do tempo, capturando efeitos de defasagem, cointegração e não estacionariedade. Já os métodos de reconciliação garantem a consistência entre as previsões agregadas e seus componentes, melhorando a precisão e a interpretabilidade dos resultados. Em conjunto, essas técnicas permitem analisar dados mais complexos, obter previsões mais precisas e facilitar a compreensão das relações hierárquicas entre as variáveis.

Além disso, ele utilizava uma forma de escrita de código que, embora fosse eficaz na época de seu lançamento, tornou-se datada em comparação aos padrões modernos de programação. Hoje, um grande número de pacotes em R adota o estilo de programação `tidy`, que prioriza a legibilidade, a consistência e a eficiência do código.

Sendo um movimento iniciado em 2014 a partir do artigo Tidy Data de Hadley Wickham, o estilo `tidy` é caracterizado por uma abordagem mais organizada e intuitiva na manipulação e análise de dados. Ele utiliza funções que operam de maneira declarativa, permitindo que o código seja lido quase como uma sequência de instruções em linguagem natural. Um dos aspectos centrais do estilo `tidy` é o uso do operador pipe (%>% ou |>), que facilita o encadeamento de funções, tornando o fluxo de trabalho mais linear e fácil de seguir.

Ainda, o `tidy` promove a ideia de "tidy data", onde os dados são organizados de forma padronizada: cada variável em uma coluna, cada observação em uma linha, e cada tipo de unidade observacional em uma tabela separada. Esse formato simplifica tanto a análise quanto a comunicação dos resultados, tornando o código mais compreensível e menos propenso a erros.

O estilo `tidy` se tornou o padrão de fato para muitos usuários de R, sendo amplamente adotado por pacotes populares no ecossistema `tidyverse`, como `dplyr`, `tidyr` e `ggplot2`. Esse movimento reflete uma evolução na prática de programação, onde a cla-

reza, a reprodutibilidade e a eficiência são priorizadas, facilitando o trabalho colaborativo e a manutenção do código ao longo do tempo.

O termo `tidyverse` foi criado em 2014, quando essa maneira de se organizar dados e criar funções iniciou seu processo de consolidação. Desde de lá, inúmeros pacotes foram desenvolvidos, atigindo outros nichos além da área de organização de dados, como os campos de amostragem e modelagem. Enquanto os pacotes base do `tidyverse` têm como objetivo a organização e manipulação de dados estruturados, duas outras bibliotecas se destacam

O pacote `srvyr` no R é uma extensão do pacote `dplyr`, projetada especificamente para facilitar a manipulação e análise de dados de pesquisas (survey data) com planos amostrais complexos. Ele permite que os usuários realizem análises de pesquisas usando a sintaxe familiar do `tidyverse`, o que torna mais intuitivo e eficiente trabalhar com dados de pesquisas que envolvem pesos amostrais, estratificação e conglomerados.

Já o pacote `tidymodels` é uma coleção de pacotes que fornece ferramentas para a modelagem estatística e aprendizado de máquina de uma maneira coesa, utilizando o estilo de programação `tidyverse`. Ele oferece uma abordagem unificada e consistente para todo o ciclo de vida de modelagem, desde a pré-processamento de dados, criação e ajuste de modelos, tunagem de hiperparâmetros, até a avaliação e seleção de modelos.

Apesar dessa ampla cobertura, ainda havia uma lacuna significativa no universo de pacotes `tidy`: a organização e modelagem de dados de séries temporais. Dados temporais apresentam características específicas que exigem abordagens e ferramentas especializadas para sua análise adequada, algo que não estava plenamente contemplado pelos pacotes existentes.

A seguinte frase descrita por Wickham é citada por Hyndman como a motivação inicial para a descontinuidade do `Forecast` em vista do desenvolvimento de um novo conjunto de pacotes de organização e modelagem de séries temporais que seriam construídos utilizando a metodologia `tidy`, *"Tidy datasets are all alike, but every messy dataset is messy in its own way."* (Wickham 2014). Hyndman cita a especificidade ao se trabalhar com dados temporais

Dados estruturados não temporais possuem uma estrutura padrão em que cada linha é uma observação, que pode ser um indivíduo, país, lugar e assim por diante, e cada coluna representa uma variável dessa observação estudada. Por sua vez, dados temporais trazem um desafio único. Em dados temporais, cada linha representa uma unidade temporal (dia, mês, ano, etc.), e cada coluna pode representar uma variável. Esse tipo de dados requerem técnicas especiais para lidar com dependências temporais, tendências e sazonalidades que não estão presentes em dados estruturados convencionais. A manipulação e análise de dados temporais exigem ferramentas que possam manejar essas características intrínsecas do tempo, como a autocorrelação e a variabilidade ao

longo do tempo.

A manipulação de dados temporais apresenta desafios únicos que vão além das operações básicas de limpeza e transformação de dados. A conversão de formatos de data (por exemplo, de ano/mês/dia para dia/mês/ano), a imputação de valores faltantes em datas específicas e a unificação de dados de diferentes fontes com frequências distintas exigem um conjunto característico de ferramentas. Enquanto o `tidyverse` oferece uma gama robusta de funções para manipulação de dados em geral, a natureza sequencial e temporal dos dados exige soluções mais especializadas. A complexidade inerente a esses problemas torna os pacotes do tidyverse insuficientes para lidar com as nuances da análise de séries temporais

A transformação de datas, por exemplo, pode parecer trivial, mas possui desafios únicos. Diferentes países e regiões utilizam formatos de data distintos (dia/mês/ano, mês/dia/ano, ano/mês/dia). A não conversão para um formato padrão pode levar a interpretações incorretas. Além disso ao trabalhar com dados de diferentes localidades, é crucial considerar as diferenças de fuso horário. A não conversão para um fuso horário comum pode gerar inconsistências nos dados, assim como a mudança para o horário de verão em algumas regiões pode introduzir descontinuidades nos dados se não for devidamente considerada. Ainda, a inclusão ou exclusão de um dia extra a cada quatro anos, característica do ano bissexto, impacta diretamente a precisão de cálculos envolvendo datas. A falta de ajuste para o ano bissexto pode gerar descontinuidades e dificultar a identificação de padrões sazonais.

Assim como acontece com dados não temporais, séries temporais também podem apresentar valores faltantes. No entanto, enquanto técnicas de imputação relativamente simples, como a substituição por média ou mediana, podem ser adequadas para dados não temporais, as séries temporais exigem abordagens mais sofisticadas devido à sua natureza sequencial e dependência temporal. Valores faltantes em séries temporais representam um desafio maior porque as imputações imprecisas podem distorcer padrões temporais fundamentais, como tendências, ciclos e sazonalidade. Por exemplo, substituir um valor faltante em uma série temporal com a média geral pode ignorar variações sazonais importantes, levando a ajustes sazonais errôneos e, consequentemente, a previsões ou análises imprecisas.

Além disso, a imputação inadequada pode introduzir viés nos modelos preditivos, afetando negativamente a capacidade do modelo de capturar a verdadeira estrutura dos dados. Técnicas mais complexas, como interpolação linear, suavização exponencial, ou até mesmo modelos preditivos específicos para séries temporais, como ARIMA ou modelos de estado espaço, são frequentemente necessárias para preservar a integridade dos padrões temporais.

Para preencher essa lacuna, Rob Hyndman, George Athanasopoulos e outros co-

laboradores desenvolveram, no final de 2020, o conjunto de pacotes do `tidyverts`. Este novo conjunto de ferramentas foi projetado para integrar o estilo de programação do tidyverse com os modelos avançados e robustos do pacote `forecast`. O `tidyverts` proporciona uma solução moderna e eficiente para a análise de séries temporais, alinhando-se às práticas e padrões do `tidyverse` e promovendo um fluxo de trabalho mais coeso e intuitivo para os usuários.

A biblioteca tem como principais pacotes.

- **tsibble**: Para limpeza a manipulação de bases de dados de séries temporais.
- **fable**: Para modelagem de séries, oferecendo os modelos ARIMA e suas variações sazonais e com covariáveis, ETS, NNETAR, método de Croston, VAR, método Theta.
- **fabletools**: Oferece ferramentas para construção de modelos para o `fable`
- **feasts**: Para extração de informações e estatísticas dos modelos ajustados e de previsões.

Cada pacote citado foi construído para o desenvolvimento de um *pipeline* de modelagem e sua fácil escalabilidade

Dado a facilidade de construção de um processo de modelagem, com ótimo desempenho computacional, tal biblioteca de pacotes foi utilizada na plataforma de análises estatísticas JFSalvandoTodos para a seção de previsão de casos confirmados das SRAGs. Apesar da facilidade da construção de um ciclo de limpeza, modelagem e previsão de valores, notou-se rapidamente que o pacote de modelagem `fable` apresentava uma lacuna ao se trabalhar com dados de contagem. Como já citado anteriormente, a utilização do modelo ARIMA e NNETAR apresentaram diferentes complexidades e obstáculos, indo desde tempo de execução computacional de modelagem extremamente alto, até previsões de valores negativos

Dado a necessidade de se trabalhar com modelos específicos para dados de contagem, iniciou-se a ideia da criação de um pacote para esse fim, denominado posteriormente de `fableCount`

Antes de destacar a criação do pacote, devemos introduzir o conceito de pipeline de dados e como o `tidyverts` e mais especificamente os pacotes `fable` e `fableCount` se localizam em um processo de modelagem

3.2 PIPELINE DE DADOS

Os pipelines de dados têm suas raízes em conceitos de engenharia de software, onde um *"pipeline"* refere-se a uma série de etapas interconectadas que processam dados de

entrada para produzir uma saída desejada. Esses conceitos foram adaptados para o campo da ciência de dados à medida que os pesquisadores começaram a lidar com conjuntos de dados cada vez maiores e mais complexos, exigindo abordagens sistemáticas e escaláveis para processamento e análise de dados.

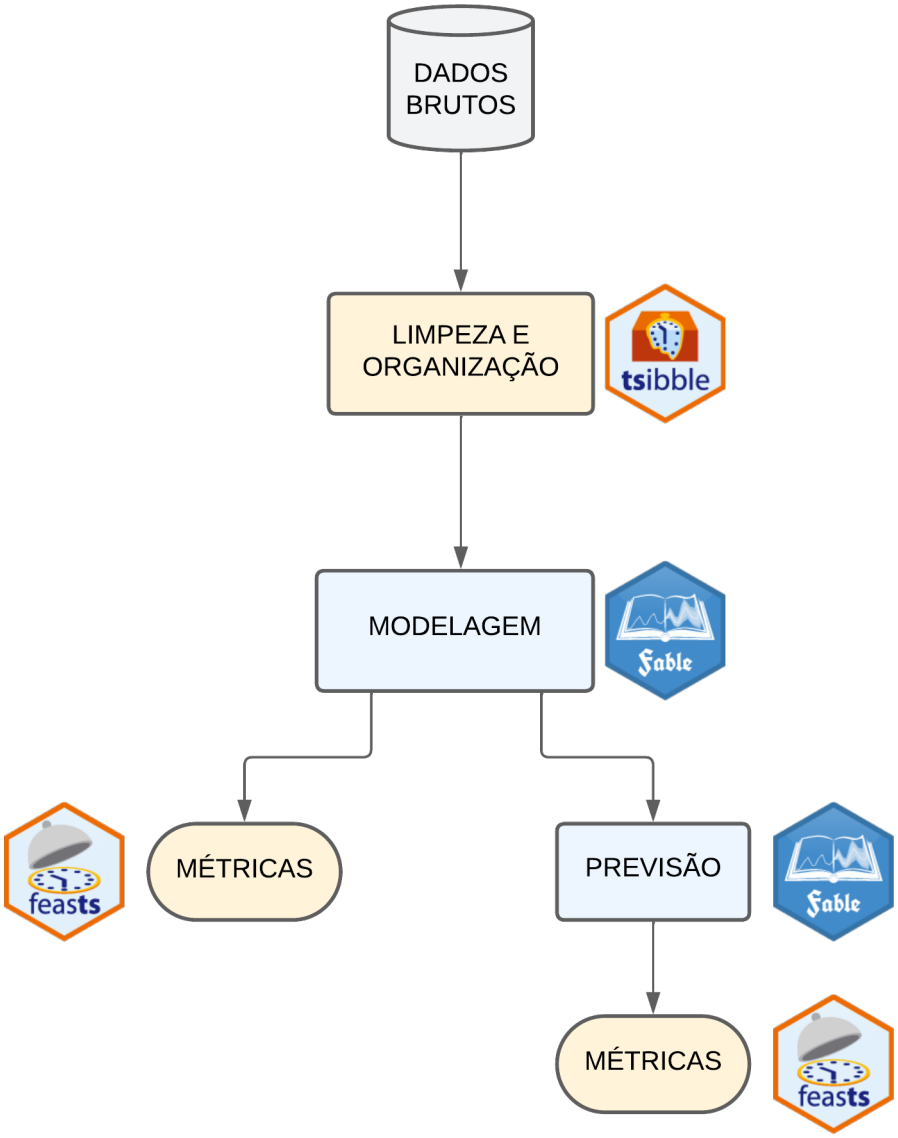
Dentre o contexto de séries temporais, um *pipeline* pode ser divido em três grandes processos

(i) **Limpeza e Organização de dados:** Esta etapa envolve o preparo dos dados para análise. Isso pode incluir a detecção e tratamento de valores ausentes, a remoção de outliers, a correção de erros de digitação e a padronização de formatos de dados. Para séries temporais, também é importante garantir que os dados estejam organizados corretamente no tempo, ou seja, em ordem cronológica, para que a análise subsequente possa ser realizada adequadamente.

(ii) **Análise e Engenharia de Características:** Nessa fase, são extraídas e criadas características (features) relevantes a partir dos dados brutos para melhorar a capacidade preditiva dos modelos. Em séries temporais, isso pode envolver a criação de novas variáveis, como médias móveis, diferenças entre valores consecutivos, tendências, sazonalidades e outras transformações que possam capturar padrões temporais importantes. A análise exploratória de dados também é crucial nesta etapa para compreender melhor as relações e características dos dados.

(i) **Modelagem e Previsão:** A última etapa consiste na construção e ajuste de modelos para inferência ou previsão que serão utilizados para interpretação de parâmetros ou previsão de valores futuros. Isso pode incluir a aplicação de modelos estatísticos tradicionais, como ARIMA e ETS, ou técnicas mais modernas de aprendizado de máquina, como redes neurais feedforward (FFNN), tendo como exemplo o modelo NNETAR, redes neurais recorrentes (RNNs) e as Long Short-Term Memory (LSTM). A avaliação dos modelos é realizada através de métricas de desempenho, como os critérios de informação para modelos focados em inferência e métricos baseadas no erro como RMSE, MAE, MAPE para modelos com foco em previsão.

O seguinte fluxograma foi elaborado para exemplificar um pipeline de modelagem utilizando o framework do `tidyverts`



O pacote construído e descrito nesse trabalho, `fableCount`, se localiza na etapa ocupada pelo pacote `fable`, tendo como foco a disponibilização de modelos temporais de contagem e algoritmos para a automatização de tal processo. Ele surge como uma extensão ao `fable`, onde sua versão base não apresenta modelos para dados de contagem

Além do `fableCount`, outros pacotes foram desenvolvidos como extensão ao `fable` base e que valem serem mencionados.

O primeiro foi o `fable.prophet`, pacote que tem como objetivo disponibilizar o modelo prophet para usuários do tidyverts. O prophet foi um modelo de séries temporais criado pelo Facebook (agora META), e tem como principal característica a capacidade de modelar sazonalidade complexa em conjunto de dados de alta dimensionalidade ao mesmo tempo que se mostra robusto a valores faltantes.

O segundo foi o fable.binary que trás modelos para séries temporais binárias. Ele

possui 2 modelos principais: logístico e NNET. O modelo logístico temporal se assemelha ao modelo logístico padrão, vindo da teoria de modelos lineares generalizados, sendo uma extensão a dados temporais. Já o NNET é um modelo baseado em redes neurais, mais especificamente em uma rede feedforward com apenas uma camada oculta, chamada de Single Layer Perceptron (SLP)

Assim o pacote construído e detalhado nesse trabalho é a terceira extensão ao fable. Apesar dos pacotes descritos serem distintos entre si, onde cada um possui sua própria gama de modelos e nicho específico, a construção de cada um é feita com base nas especificações do `tidyverts`, utilizando a biblioteca `fabletools` para construção de pacotes de extensão ao `fable`. Assim, todos eles possuem a mesma estrutura de utilização em código R e podem ser implementados em um mesmo pipeline, facilitando comparação entre modelos, como a busca por aquele de melhor qualidade de ajuste, melhores valores previstos e melhores tempos de execução computacional

3.3 ESTRUTURA DO PACOTE

Um pacote em R segue uma estrutura base, que inclui espaços distintos dedicados ao armazenamento de funções, conjuntos de dados e informações específicas do pacote.

Dentro do escopo das funções de um pacote, existem dois grupos principais: as funções externas, que são disponibilizadas para uso direto pelo usuário, e as funções internas, que são utilizadas exclusivamente dentro do pacote e não estão acessíveis ao usuário.

Como será detalhado nos próximos tópicos, a principal funcionalidade do pacote é a disponibilização de métodos automatizados para a seleção de modelos. As funções responsáveis pela execução desses métodos automatizados são funções internas, não exportadas diretamente para o usuário. Em vez disso, o acesso a esses métodos é feito através de uma função principal de modelagem, que contém gatilhos para acionar as funções automatizadas conforme necessário.

Essa explicação é crucial, pois o número de funções descritas neste trabalho é significativamente menor do que o total de funções disponíveis internamente no pacote. O foco está nas funções acessíveis ao usuário, embora uma complexa infraestrutura de funções internas suporte e automatize os processos de modelagem.

As seções "Implementação INGARCH" e "Implementação GLARMA" descrevem detalhadamente as funções exportadas ao usuário que estão associadas a cada um desses modelos. No capítulo subsequente, "Algoritmos para Automatização de Modelagem", é apresentada a construção das funções responsáveis pela seleção automática de distribuições, ordens de parâmetros, e modelos de previsão. Por serem de uso interno, essas funções não são exportadas para o usuário e, portanto, não podem ser acessadas diretamente em

um arquivo ".R".

Além disso, para uma melhor compreensão das próximas seções e capítulos, é necessário explicar certos tipos de objetos do R que são específicos para as etapas de organização e modelagem de séries temporais. Esses objetos, devido ao seu nicho de aplicação, não são comumente utilizados por usuários gerais do R, mas são essenciais para o trabalho com séries temporais.

- **tibble** - Se autodenomina como a versão aperfeiçoada de uma *data.frame*, e é o objeto tabular padrão do Tidyverse
- **tsibble** - É um *Time Series Tibble*, ou um Tibble para Séries Temporais. Possui a estrutura padrão de um *tibble*, trazendo maiores funcionalidades para manipulação de datas
- **mable** - É um *Model Table*, ou uma Tabela de Modelos. Esse termo refere-se a uma estrutura que armazena diferentes modelos ajustados em um formato de tabela, facilitando a comparação e a análise dos modelos em um único quadro de dados. Representa o objeto de retorno ao se utilizar a função *INGARCH()* e *GLARMA()*

3.4 IMPLEMENTAÇÃO INGARCH

Para implementação da estimação e produção de previsões, o pacote `tscount` é utilizado. O pacote construído oferece 6 funções aos usuários relacionadas ao modelo INGARCH

São elas:

- **INGARCH()** - Função base para estimação do modelo INGARCH
- **fitted()** - Extrai os valores estimados de um modelo construído
- **forecast()** - Estima valores para um intervalo de tempo futuro, ou seja, previsão de novos valores
- **glance()** - Retorna as métricas erro-padrão, log-verossimilhança, AIC e BIC do modelo construído
- **residuals()** - Extrai os resíduos de um modelo construído
- **tidy()** - Retorna os coeficientes do modelo, assim como suas métricas de variância, e intervalo de confiança

Cada função possui um objeto de entrada específico e um objeto de retorno correspondente. Além disso, algumas funções incluem argumentos adicionais que precisam

ser explicados em detalhes para garantir um uso adequado do pacote. Para oferecer um entendimento mais aprofundado, cada função foi detalhadamente descrita nos tópicos a seguir.

3.4.1 INGARCH()

É a função base para estimação de modelo INGARCH.

Em um fluxo de modelagem, a função é chamada da seguinte maneira

Exemplo chamada modelo INGARCH

```
# Exemplo chamada modelo INGARCH base ------------------------------------
dados |>
  fabletools::model(
    nome_modelo = fableCount::INGARCH(variavel_reposta ~ pq(ordem_AR, ordem_MA))
    )

```

Elaborado pelo autor (2025)

Possui os seguintes argumentos.

- **formula** - Argumento que define as ordens autorregressiva e de médias móveis do modelo. Esse argumento possui 3 parcelas distintas.

  pq - Define os termos autorregressivas e de médias móveis não sazonais, pode ser definido pelo usuário, ou se for omitido, o algoritmo de seleção automática de parâmetros é acionado. O algoritmo de seleção automática de parâmetros ajustará o melhor modelo com base no critério de informação

  PQ - Define os termos autoregressivos e médias móveis sazonais, pode ser definido pelo usuário, ou se for omitido, o algoritmo de seleção automática de parâmetros é acionado (gatilho padrão aciona o algoritmo de selação `ARMA-Based`). Tal algoritmo será detalhado no próximo capítulo

  xreg - Define variáveis exógenas para utilização no modelo.

- **ic** - Representa o critério de informação que deve ser utilizado se o algoritmo de seleção automática de parâmetros for acionado, possuindo as opções ”AIC” ou ”BIC”

- **link** - Função de ligação que deve ser utilizada para o modelo generalizado, possuindo as opções ”identity” ou ”log”

- **distr** - Função de densidade de probabilidade que deve ser utilizada para o modelo generalizado, possuindo as opções ”poisson” ou ”nbinom” (binomial negativa). Se esse argumento for omitido, o algoritmo de seleção automática de distribuição é acionado

- **algorithm** - Define qual o algoritmo de seleção automática de ordem de parâmetros vai ser utilizado no caso das ordens ”pq” serem omitidas. Possui 3 opções, ”naive_search”, ”arma_based” e ”LASSO”.

Cada método de seleção automática de distribuição e seleção automática de ordens de parâmetros será aprofundado no capítulo ”Algoritmos para Automatização de Modelagem”

A função tem como objeto de retorno um ”mabble”

Um exemplo considerando um modelo INGARCH(2,1) é dado na seguinte imagem. O modelo construído foi chamado de ***exem_model_ingarch*** e foi utilizado novamente para exemplificações nas demais funções. Os dados utilizados para sua estimação foram simulados a partir de um modelo INGARCH(2,1) com distribuição Poisson com tamanho amostral igual a 100

Exemplo utilização INGARCH

```
43
44 exem_model_ingarch = serie_simulada |>
45   fabletools::model(
46     ing = fableCount::INGARCH(var_resposta ~ pq(2, 1),
47                               distr = 'poisson',
48                               link = 'identity')
49   )
50
```

Elaborado pelo autor (2025)

O objeto de retorno é um *mable* tendo a seguinte estrutura

Exemplo objeto de retorno mable INGARCH

```
# A mable: 1 x 1
                ing
             <model>
1 <INGARCH(2, 1)>
>
```

Elaborado pelo autor (2025)

3.4.2 fitted()

Função que tem como objetivo extrair os valores estimados de um modelo construído e possui os seguintes argumentos

Em um fluxo de modelagem, a função é chamada da seguinte maneira

Exemplo função fitted INGARCH

```
modelo_ingarch = dados |>
  fabletools::model(
    nome_modelo = fableCount::INGARCH(variavel_reposta ~ pq(ordem_AR, ordem_MA))
  )


modelo_ingarch |>
  fitted()
```

Elaborado pelo autor (2025)

- **object** - Este é o argumento principal da função e refere-se ao modelo de série temporal que foi previamente ajustado.
- **...** - Argumentos adicionais que podem ser passados para métodos específicos de subclasses. Geralmente, esses argumentos não são necessários para o uso básico da função.

Utilizando novamente o modelo construido, ***exem_model_ingarch***, que dado um número de observações igual a 100, teve um intervalo temporal de estudo definido como 1901 a 2000 (100 anos), a função fitted apresenta a seguinte estrutura de retorno

Exemplo utilização fitted INGARCH

```

exem_model_ingarch |>
  fitted()

```

Elaborado pelo autor (2025)

Exemplo retorno fitted INGARCH

```
> exem_model_ingarch |>
+   fitted()
# A tsibble: 100 x 3 [1Y]
# Key:       .model [1]
   .model  date .fitted
   <chr>  <dbl>   <dbl>
 1 ing     1901    13.8
 2 ing     1902    15.9
 3 ing     1903    17.0
 4 ing     1904    16.5
 5 ing     1905    13.8
 6 ing     1906    16.0
 7 ing     1907    17.7
 8 ing     1908    19.3
 9 ing     1909    21.6
10 ing     1910    16.4
# i 90 more rows
# i Use `print(n = ...)` to see more rows
>
```

Elaborado pelo autor (2025)

3.4.3 forecast()

Função para o cálculo de previsões. Para previsões de 1 passo à frente, ele retorna uma previsão paramétrica, baseada na distribuição especificada pelo parâmetro 'distr'. Para previsões de múltiplos passos à frente, a distribuição não é conhecida analiticamente, então é utilizado um bootstrap paramétrico.

Em um fluxo de modelagem, a função é chamada da seguinte maneira

Exemplo função forecast INGARCH

```
19
20  modelo_ingarch |>
21    forecast(h = horizonte_de_previsão)
22
```

Elaborado pelo autor (2025)

Possui os seguintes argumentos.

- **object** - Este é o argumento principal da função e refere-se ao modelo de série temporal que foi previamente ajustado.
- **h** - Controla o horizonte de previsão. É um valor numérico que especifica o número de passos à frente para os quais as previsões serão geradas
- **new_data** - Se o modelo construído possuir variáveis exógenas, essas devem tem seus valores futuros passados através desse argumento .
- **...** - Argumentos adicionais que podem ser passados para métodos específicos de subclasses. Geralmente, esses argumentos não são necessários para o uso básico da função.

A função tem como objeto de retorno um *tsibble*, contando com 4 colunas: a primeira é o nome do modelo, a segunda é a data que do *forecast*, a terceira é a distribuição da previsão para aquele momento do tempo (a distribuição é utilizada para construção de intervalos de confiança) e a quarta coluna representa a média da previsão

Utilizando novamente o modelo construido, ***exem_model_ingarch***.

É importante que a função ***forecast*** pode ter 2 tipos de retorno dependendo do horizonte de previsão escolhido para o modelo INGARCH. Para previsão de 1 passo a frente, a distribuição escolhida para o modelo é utilizada.

Já para previsões possuindo um horizonte de previsão maior que 1, a distribuição analítica não é conhecida e portanto um bootstrap paramétrico é utilizado para a previsão de valores

Para um horizonte de previsão de 1 passo a frente, a função possui o seguinte comportamento

Exemplo utilização forecast INGARCH (h = 1)

```
67
68  exem_model_ingarch |>
69    fabletools::forecast(h = 1)
70
```

Elaborado pelo autor (2025)

Exemplo retorno forecast INGARCH (h = 1)

```
> exem_model_ingarch |>
+   fabletools::forecast(h = 1)
# A fable: 1 x 4 [1Y]
# Key:     .model [1]
  .model  date        x .mean
  <chr>   <dbl>   <dist> <dbl>
1 ing     2001 Pois(15)  14.7
>
```

Elaborado pelo autor (2025)

Já para um horizonte igual a 10 (sendo portanto maior que 1), a função apresenta a seguinte estrutura

Exemplo utilização forecast INGARCH (h > 1)

```
71
72  exem_model_ingarch |>
73    fabletools::forecast(h = 10)
74
```

Elaborado pelo autor (2025)

Exemplo retorno forecast INGARCH (h > 1)

```
> exem_model_ingarch |>
+   fabletools::forecast(h = 10)
# A fable: 10 x 4 [1Y]
# Key:     .model [1]
   .model  date            x .mean
   <chr>   <dbl>       <dist> <dbl>
 1 ing     2001 sample[1000]  14.6
 2 ing     2002 sample[1000]  14.5
 3 ing     2003 sample[1000]  14.4
 4 ing     2004 sample[1000]  14.5
 5 ing     2005 sample[1000]  14.5
 6 ing     2006 sample[1000]  14.4
 7 ing     2007 sample[1000]  14.1
 8 ing     2008 sample[1000]  14.0
 9 ing     2009 sample[1000]  13.9
10 ing     2010 sample[1000]  13.9
>
```

Elaborado pelo autor (2025)

É possivel ver as diferenças entre os objetos de retorno. Enquanto o primeiro possui no coluna ”x”, a distribuição utilizada para estimação do modelo, onde no exemplo utilizado é uma Poisson, a segunda previsão apresenta na coluna ”x” uma distribuição baseada em um bootstrap paramétrico de 1000 valores

3.4.4 glance()

Função que retorna uma tabela com as métricas erro-padrão, log-verossimilhança, AIC e BIC do modelo construído.

Em um fluxo de modelagem, a função é chamada da seguinte maneira

Exemplo função glance INGARCH

```
25
26  modelo_ingarch |>
27    glance()
28
```

Elaborado pelo autor (2025)

Possuindo os seguintes argumentos

- **x** - Este é o argumento principal da função e refere-se ao modelo de série temporal que foi previamente ajustado.
- **...** - Argumentos adicionais que podem ser passados para métodos específicos de subclasses. Geralmente, esses argumentos não são necessários para o uso básico da função.

O erro-padrão é calculado a partir do estimador não viciado dado por $\frac{\sum_{t=0}^{T}(\hat{e}_t)^2}{n-k+1}$

Utilizando novamente o modelo construido, ***exem_model_ingarch***, a função apresenta a seguinte estrutura de retorno

Exemplo utilização glance INGARCH

```
93
94  exem_model_ingarch |>
95    fabletools::glance()
96
```

Elaborado pelo autor (2025)

Exemplo retorno glance INGARCH

```
> exem_model_ingarch |>
+   fabletools::glance()
# A tibble: 1 × 5
  .model sigma2 log_lik   AIC   BIC
  <chr>   <dbl>   <dbl> <dbl> <dbl>
1 ing      12.3   -261.  530.  540.
>
```

Elaborado pelo autor (2025)

Exemplo função residuals INGARCH

```
30
31  modelo_ingarch |>
32    residuals()
33
```

Elaborado pelo autor (2025)

3.4.5 residuals()

Função que extrai os resíduos do modelo construído.

Em um fluxo de modelagem, a função é chamada da seguinte maneira

Possui apenas um argumento

- **object** - Este é o argumento principal da função e refere-se ao modelo de série temporal que foi previamente ajustado.

Utilizando novamente o modelo construido, ***exem_model_ingarch***, a função apresenta a seguinte estrutura de retorno

Exemplo utilização residuals INGARCH

```
101
102  exem_model_ingarch |>
103    residuals()
104
```

Elaborado pelo autor (2025)

Exemplo retorno residuals INGARCH

```
> exem_model_ingarch |>
+   residuals()
# A tsibble: 100 x 3 [1Y]
# Key:       .model [1]
   .model  date .resid
   <chr>  <dbl>  <dbl>
 1 ing     1901  8.84
 2 ing     1902 -4.27
 3 ing     1903 -3.01
 4 ing     1904  2.75
 5 ing     1905  1.19
 6 ing     1906 -0.673
 7 ing     1907  5.64
 8 ing     1908 -3.39
 9 ing     1909 -0.658
10 ing     1910  6.06
# i 90 more rows
# i Use `print(n = ...)` to see more rows
>
```

Elaborado pelo autor (2025)

3.4.6 tidy()

Função para extrair métricas sobre os coeficientes. A função retorna a estimativa pontual, desvio-padrão e intervalo de confiança. O intervalo de confiança pode ser calculado através da aproximação via distribuição Normal ou via Bootstrap

Por padrão, os erros padrão e os intervalos de confiança são baseados em uma aproximação normal do estimador de máxima verossimilhança. Os erros padrão são as raízes quadradas dos elementos diagonais da inversa da matriz de informação. Já para o erro padrão para o coeficiente de sobredispersão ***"sigmasq"*** quando a distribuição Binomial Negativa é utilizada, como não há uma aproximação analítica, seu erro padrão e seu intervalo de confiança são definidos como NA.

Ela possui os seguintes argumentos

- **object** - Este é o argumento principal da função e refere-se ao modelo de série temporal que foi previamente ajustado.
- **type** - Refere-se ao a forma na qual o intervalo e confiança deve ser calculado. "normalaproxx" se refere a uma aproximação via distribuição Normal e "boot" se refere a método de bootstrap paramétrico
- **...** - Argumentos adicionais que podem ser passados para métodos específicos de subclasses. Geralmente, esses argumentos não são necessários para o uso básico da função.

Em um fluxo de modelagem, a função é chamada da seguinte maneira

Exemplo função tidy INGARCH

```
34
35  modelo_ingarch |>
36    tidy()
37
```

Elaborado pelo autor (2025)

Utilizando novamente o modelo construido, ***exem_model_ingarch***, a função apresenta a seguinte estrutura de retorno

Exemplo utilização tidy INGARCH

```
110
111  exem_model_ingarch |>
112    fabletools::tidy()
113
```

Elaborado pelo autor (2025)

Exemplo retorno tidy INGARCH

```
>
> exem_model_ingarch |>
+   fabletools::tidy()
# A tibble: 4 × 6
  .model term        estimate std.error `CI(lower)` `CI(upper)`
  <chr>  <chr>          <dbl>     <dbl>       <dbl>       <dbl>
1 ing    (constant)     1.85      1.06       -0.218        3.92
2 ing    ar_1           0.238     0.0992      0.0435       0.432
3 ing    ar_2           0.313     0.143       0.0333       0.593
4 ing    ma_1           0.283     0.202      -0.113        0.679
>
```

Elaborado pelo autor (2025)

3.5 IMPLEMENTAÇÃO GLARMA

Assim como as função do modelo INGARCH, todas a funções do modelo GLARMA foram explicitadas. A estrutura de funções para a utilização por parte do usuário é a mesma do INGARCH, porém alguns parâmetros das funções são distintos, dado a forma de estimação de cada modelo

Temos as seguintes funções elas:

- **GLARMA()** - Função base para estimação do modelo GLARMA
- **fitted()** - Extrai os valores estimados de um modelo construído
- **forecast()** - Estima valores para um intervalo de tempo futuro, ou seja, previsão de novos valores
- **glance()** - Retorna as métricas erro-padrão, log-verossimilhança, AIC e BIC do modelo construído
- **residuals()** - Extrai os resíduos de um modelo construído
- **tidy()** - Retorna os coeficientes do modelo, assim como suas métricas de variância, e intervalo de confiança

3.5.1 GLARMA()

É a função base para estimação de modelo GLARMA.

Em um fluxo de modelagem, a função é chamada da seguinte maneira

Exemplo chamada modelo GLARMA

```r
# Exemplo chamada modelo GLARMA base ----------------------------------
dados |>
  fabletools::model(
    nome_modelo = fableCount::GLARMA(variavel_reposta ~ pq(ordem_AR, ordem_MA))
  )

```

Elaborado pelo autor (2025)

Possui os seguintes argumentos.

- **formula** - Argumento que define as ordens autorregressiva e de médias móveis do modelo. Esse argumento possui 3 parcelas distintas.

  pq - Define os termos autorregressivas e de médias móveis não sazonais, pode ser definido pelo usuário, ou se for omitido, o algoritmo de seleção automática de parâmetros é acionado. O algoritmo de seleção automática de parâmetros ajustará o melhor modelo com base no critério de informação

  PQ - Define os termos autorregressivos e médias móveis sazonais, pode ser definido pelo usuário, ou se for omitido, o algoritmo de seleção automática de parâmetros é acionado (somente para o algoritmo 'arma_to_glarma'). O algoritmo de seleção automática de parâmetros vai se ajustar ao melhor modelo baseado no critério de informação

  xreg - Define variáveis exógenas para utilização no modelo.

- **ic** - Representa o critério de informação que deve ser utilizado se o algoritmo de seleção automática de parâmetros for acionado, possuindo as opções "AIC" ou "BIC"

- **distr** - Função de densidade de probabilidade que deve ser utilizada para o modelo generalizado, possuindo as opções "poisson" ou "nbinom"(binomial negativa). Se esse argumento for omitido, o algoritmo de seleção automática de distribuição é utilizado

- **method** - Método iterativo que deve ser utilizado para a estimação do modelo. Possui as opções "FS" (Fisher scoring) e "NR" (Newton-Raphson)

- **algorithm** - Define qual o algoritmo de seleção automática de ordem de parâmetros vai ser utilizado no caso das ordens pq serem omitidas. Possui 2 opções, "naive_search", "arma_based".

- **residuals** - Tipo de resíduo para ser utilizado (como definido na seção de definição matemática do modelo). Possui as opções "Pearson" e "Score"

Cada método de seleção automática de distribuição e seleção automática de ordens de parâmetros será aprofundado no capítulo "Algoritmos para Automatização de Modelagem"

A função tem como objeto de retorno um "mabble"

Um exemplo considerando um modelo GLARMA(1,0) é dado na seguinte imagem. O modelo construído foi chamado de ***exem_model_glarma*** e foi utilizado novamente para exemplificações nas demais funções. Os dados utilizados para sua estimação foram simulados a partir de um modelo GLARMA(1,0) com tamanho amostral igual a 100

Exemplo utilização GLARMA

```
115
116 exem_model_glarma = serie_simulada |>
117   fabletools::model(
118     gla = GLARMA(var_resposta ~ pq(1, 0),
119                  method = "FS",
120                  distr = "poisson")
121   )
122
```

Elaborado pelo autor (2025)

Exemplo objeto de retorno mable GLARMA

```
> exem_model_glarma
# A mable: 1 x 1
                gla
             <model>
1 <GLARMA(1, 0)>
>
```

Elaborado pelo autor (2025)

3.5.2 fitted()

Função que tem como objetivo extrair os valores estimados de um modelo construído e possui os seguintes argumentos

Em um fluxo de modelagem, a função é chamada da seguinte maneira

Exemplo função fitted GLARMA

```
123
124 modelo_glarma = dados |>
125   fabletools::model(
126     nome_modelo = fableCount::GLARMA(variavel_reposta ~ pq(ordem_AR, ordem_MA))
127   )
128
129
130 modelo_glarma |>
131   fitted()
132
```

Elaborado pelo autor (2025)

- **object** - Este é o argumento principal da função e refere-se ao modelo de série temporal que foi previamente ajustado.
- **...** - Argumentos adicionais que podem ser passados para métodos específicos de subclasses. Geralmente, esses argumentos não são necessários para o uso básico da função.

Utilizando novamente o modelo construido, ***exem_model_glarma***, que dado um número de observações igual a 100, teve um intervalo temporal de estudo definido como 1901 a 2000 (100 anos), a função fitted apresenta a seguinte estrutura de retorno

Exemplo utilização fitted GLARMA

```
135
136 exem_model_glarma |>
137   fitted()
138
```

Elaborado pelo autor (2025)

Exemplo retorno fitted GLARMA

```
>
> exem_model_glarma |>
+     fitted()
# A tsibble: 100 x 3 [1Y]
# Key:       .model [1]
   .model  date .fitted
   <chr>  <db1>   <db1>
 1 gla     1901   10.9
 2 gla     1902   14.1
 3 gla     1903    9.57
 4 gla     1904   11.3
 5 gla     1905   11.7
 6 gla     1906   11.3
 7 gla     1907   11.1
 8 gla     1908   13.2
 9 gla     1909   10.0
10 gla     1910   11.9
# i 90 more rows
# i Use `print(n = ...)` to see more rows
>
```

Elaborado pelo autor (2025)

3.5.3 forecast()

Função para o cálculo de previsões. A função retorna uma previsão paramétrica, baseada na distribuição especificada pelo parâmetro ”distr”, independentemente do horizonte de previsão solicitado. Diferente do método de *forecast* do INGARCH, que retorna uma previsão baseado em um bootstrap paramétrico para horizontes de previsão maiores que 1

Em um fluxo de modelagem, a função é chamada da seguinte maneira

Exemplo função forecast GLARMA

```
131
132 modelo_glarma |>
133   forecast(h = horizonte_de_previsão)
134
```

Elaborado pelo autor (2025)

Possui os seguintes argumentos.

- **object** - Este é o argumento principal da função e refere-se ao modelo de série temporal que foi previamente ajustado.
- **h** - Controla o horizonte de previsão. É um valor numérico que especifica o número de passos à frente para os quais as previsões serão geradas

- **...** - Argumentos adicionais que podem ser passados para métodos específicos de subclasses. Geralmente, esses argumentos não são necessários para o uso básico da função.

A função tem como objeto de retorno um *tsibble*, contando os 4 colunas: a primeira é o nome do modelo, a segunda é a data que o valor foi predito, a terceira é a distribuição da previsão para aquele momento do tempo (a distribuição é utilizada para construção de intervalos de confiança) e a quarta coluna representa a média da previsão

Como já destacado, diferente da função construída para o modelo glarma, que utiliza um bootstrap paramétrico para previsões maiores que 1 passo a frente, a função ***forecast*** para o modelo GLARMA utiliza a distribuição de probabilidade especificada independentemente do tamanho do horizonte de previsão. Ou seja as previsões são geradas de forma direta a partir da distribuição assumida para o processo

Utilizando novamente o modelo construido, ***exem_model_glarma***.

Exemplo utilização forecast GLARMA (h = 1)

```
141
142 exem_model_glarma |>
143   fabletools::forecast(h = 1)
144
```

Elaborado pelo autor (2025)

Exemplo retorno forecast GLARMA (h = 1)

```
>
> exem_model_glarma |>
+   fabletools::forecast(h = 1)
# A fable: 1 x 4 [1Y]
# Key:     .model [1]
  .model  date var_resposta .mean
  <chr>  <dbl>       <dist> <dbl>
1 gla     2001     Pois(10)    10
>
```

Elaborado pelo autor (2025)

Já para um horizonte igual a 10 (sendo portanto maior que 1), vemos que a função apresenta a mesma distribuição de quando calculamos uma previsão 1 passo a frente

Exemplo utilização forecast GLARMA (h > 1)

```
147
148 exem_model_glarma |>
149   fabletools::forecast(h = 10)
150
```

Elaborado pelo autor (2025)

$\neq$

Exemplo retorno forecast GLARMA (h > 1)

```
>
> exem_model_glarma |>
+   fabletools::forecast(h = 10)
# A fable: 10 x 4 [1Y]
# Key:     .model [1]
   .model  date var_resposta .mean
   <chr>  <dbl>       <dist> <dbl>
 1 gla     2001     Pois(10)    10
 2 gla     2002     Pois(12)    12
 3 gla     2003     Pois(11)    11
 4 gla     2004     Pois(11)    11
 5 gla     2005     Pois(17)    17
 6 gla     2006     Pois(18)    18
 7 gla     2007      Pois(6)     6
 8 gla     2008     Pois(10)    10
 9 gla     2009     Pois(14)    14
10 gla     2010     Pois(13)    13
>
```

Elaborado pelo autor (2025)

3.5.4 glance()

Função que retorna uma tabela com as métricas erro-padrão, log-verossimilhança, AIC e BIC do modelo construído.

Em um fluxo de modelagem, a função é chamada da seguinte maneira

Exemplo função glance GLARMA

```
140
141 modelo_glarma |>
142   glance()
143
```

Elaborado pelo autor (2025)

Possuindo os seguintes argumentos

- **x** - Este é o argumento principal da função e refere-se ao modelo de série temporal que foi previamente ajustado.
- **...** - Argumentos adicionais que podem ser passados para métodos específicos de subclasses. Geralmente, esses argumentos não são necessários para o uso básico da função.

Utilizando novamente o modelo construido, ***exem_model_glarma***, a função apresenta a seguinte estrutura de retorno

Exemplo utilização glance GLARMA

```
158
159 exem_model_glarma |>
160   glance()
161
```

Elaborado pelo autor (2025)

Exemplo retorno glance GLARMA

```
>
> exem_model_glarma |>
+   glance()
# A tibble: 1 × 4
  .model sigma2 log_lik   AIC
  <chr>   <dbl>   <dbl> <dbl>
1 gla      17.0   -282.  568.
>
```

Elaborado pelo autor (2025)

3.5.5 residuals()

Função que extrai os resíduos do modelo construído.

Em um fluxo de modelagem, a função é chamada da seguinte maneira

Exemplo função residuals GLARMA

```
145
146 modelo_glarma |>
147   residuals()
148
```

Elaborado pelo autor (2025)

Possui apenas um argumento

- **object** - Este é o argumento principal da função e refere-se ao modelo de série temporal que foi previamente ajustado.

Utilizando novamente o modelo construido, ***exem_model_glarma***, a função apresenta a seguinte estrutura de retorno

Exemplo utilização residuals GLARMA

```
170
171 exem_model_glarma |>
172   residuals()
173
```

Elaborado pelo autor (2025)

Exemplo retorno residuals GLARMA

```
>
> exem_model_glarma |>
+   residuals()
# A tsibble: 100 x 3 [1Y]
# Key:       .model [1]
   .model  date .resid
   <chr>  <dbl>  <dbl>
 1 gla     1901  9.14
 2 gla     1902 -5.09
 3 gla     1903  1.43
 4 gla     1904  2.65
 5 gla     1905  1.30
 6 gla     1906  0.740
 7 gla     1907  6.91
 8 gla     1908 -3.20
 9 gla     1909  3.00
10 gla     1910  6.12
# i 90 more rows
# i Use `print(n = ...)` to see more rows
>
```

Elaborado pelo autor (2025)

3.5.6 tidy()

Função para extrair métricas sobre os coeficientes. A função retorna a estimativa pontual, desvio-padrão e z-ratio e o p-valor do parâmetro estudado

Ela possui os seguintes argumentos

- **object** - Este é o argumento principal da função e refere-se ao modelo de série temporal que foi previamente ajustado.
- **type** - Refere-se ao a forma na qual o intervalo e confiança deve ser calculado. ”normalaproxx” se refere a uma aproximação via distribuição Normal e ”boot” se refere a método de bootstrap paramétrico
- **...** - Argumentos adicionais que podem ser passados para métodos específicos de subclasses. Geralmente, esses argumentos não são necessários para o uso básico da função.

Em um fluxo de modelagem, a função é chamada da seguinte maneira

Exemplo função tidy GLARMA

```
150
151 modelo_glarma |>
152   tidy()
153
```

Elaborado pelo autor (2025)

Utilizando novamente o modelo construido, ***exem_model_glarma***, a função apresenta a seguinte estrutura de retorno

Exemplo utilização tidy GLARMA

```
181
182 exem_model_glarma |>
183   tidy()
184
```

Elaborado pelo autor (2025)

Exemplo retorno tidy GLARMA

```
>
> exem_model_glarma |>
+   tidy()
# A tibble: 4 × 4
  .model statistic intercept      ar_1
  <chr>  <chr>         <dbl>     <dbl>
1 gla    estimate     2.39    0.0939
2 gla    std_error    0.0397  0.0230
3 gla    z_ratio     60.1     4.08
4 gla    p_value      0       0.0000445
>
```

Elaborado pelo autor (2025)

3.6 DISPONIBILIZAÇÃO E IDENTIDADE VISUAL

Como já citado anteriormente, umas das mais famosas características do R, senão a mais famosa, é o suporte da comunidade através do desenvolvimento de pacotes gratuitos. E buscando facilitar o acesso dos usuários a esses pacotes desenvolvidos pelas comunidade, o time R (R team) como é conhecida a equipe de pesquisadores do R, disponibiliza 2 opções para publicação de pacotes.

A primeira e mais simples é pelo Github. O Github é uma plataforma virtual gratuita para armazenamento de repositórios utilizando o sistema Git. Essa opção é utilizada para armazenamento do pacote em sua fase inicial, ou fase de desenvolvimento. É uma opção mais simples pois o pacote desenvolvido não passa por nenhum tipo de teste antes da publicação e sua manutenção se dá exclusivamente pelo desenvolvedor

Já a segunda opção e que se mostra como o padrão ouro para todo pacote bem estruturado é via CRAN. O CRAN (Comprehensive R Archive Network) é o repositório oficial de pacotes do R e para um pacote ser aceito e publicado nesse repositório é necessário a aprovação em diversos testes automatizados de software e testes não automatizados realizados pelo time R. Dado a robustez de testes, pacotes publicados no CRAN possuem um maior grau de acabamento e detalhes, possuindo menos erros e com uma documentação maior e mais detalhada quando comparados aos pacotes disponibilizados apenas no Github

O pacote estudado nesse trabalho, `fableCount`, possui um repositório oficial no Github para versões de desenvolvimento e também possui sua versão oficial já aceita e publicada no CRAN.

Além dessas 2 formas de baixar o pacote, uma página na web foi criada no intuito de disponibilizar de maneira simplicada a documentação oficial do pacote, facilitando o entendimento de cada modelo e função por parte dos usuários

Portanto os meios oficias de comunicação e disponibilização do pacotes são:

- **Github**: Repositório onde versões em desenvolvimento são armazenadas. Nele, os métodos e modelos mais recentes são disponibilizados, embora ainda não tenham passado por um processo intensivo de testes e documentação. Esses recursos estão em fase experimental e podem sofrer alterações antes de serem integrados às versões estáveis. Pode ser acessado via: https://github.com/Gustavo039/fableCount

- **CRAN**: Repositório onde a versão oficial do pacote é armazenada. A versão disponível passou por um rigoroso processo de testes e documentação, tanto pela equipe de desenvolvimento do pacote quanto pelo time do R. Como resultado, a probabilidade de erros nas funções disponibilizadas é extremamente baixa, garantindo maior

estabilidade e confiabilidade para os usuários. Pode ser acessado via: https://cran.r-project.org/web/packages/fableCount/

- **Página Web Oficial**: Página que oferece a documentação oficial do pacote, incluindo exemplos detalhados de uso e aplicação de cada função. Além disso, apresenta uma linha do tempo que destaca as novidades introduzidas em cada versão e suas respectivas datas de lançamento. A página também conta com uma seção dedicada aos autores e às fundações de fomento que contribuíram para o desenvolvimento do pacote. Pode ser acessado via: https://gustavo039.github.io/fableCount/

Um dos pontos de sucesso do tidyverse é sua identidade visual, onde cada pacote conta com cores, fontes, logo e estilização própria. Essa estratégia auxilia na classificação de cada pacote pelos usuários.

Durante a criação da página web, identificou-se a necessidade de desenvolver uma identidade visual para o pacote, considerando o crescimento de pacotes que funcionam como extensão do fable. Tornou-se essencial estabelecer um nome e uma identidade visual distintos para garantir uma diferenciação clara em relação aos demais. Buscando atender tais critérios,a identidade visual do pacote foi criada, incluindo conjunto de cores, tipografia, logotipo oficial e sua descrição

A identidade visual do pacote precisava seguir o padrão do `fable` base, reforçando sua natureza como uma extensão ao pacote original, mas também foi desenvolvida estrategicamente para melhorar a diferenciação em relação aos demais pacotes fable, facilitando o reconhecimento pelos usuários.

Considerando fatores como: cor, fonte, arte central e nome explicitado, o pacote teve o seguinte logo construído

Logo FableCount

Elaborado pelo autor (2025)

Vemos que em comparação com a logo do pacote fable, ele se diferencia em relação a fonte utilizada, assim como na estilização arte central. Apesar disso, o tom azulado e

e o gráfico temporal no centro da imagem continuam presentes. Dessa forma, o pacote criado possui a sua própria identidade, reforçando seus aspecto únicos, mas sem esquecer totalmente do pacote origial para o seu desenvolvimento

Comparação fableCount e fable

Elaborado pelo autor (2025)

4 ALGORITMOS PARA AUTOMATIZAÇÃO DE MODELAGEM

O seguinte capítulo tem como objetivo apresentar cada parte do fluxo de modelagem automatizadas disponibilizada no pacote.

É importante destacar o nome dado para tal processo: ”Algoritmos para Automatização”. No contexto de generalização e mecanização de processos, 3 palavras similares mas que possuem significados distintos são comumente utilizadas fora de seus nichos específicos, sendo elas ”automático”, ”automação” e ”automatização”

A palavra ”automático” buscar descrever algo que opera por si só, sem a necessidade de intervenção humana contínua, e portanto é um adjetivo que descreve algo que funciona por conta própria. Onde seu desenvolvimento foi idealizado para que não houvesse intervenção humana em nenhum momento do processo.

A ”automação” está diretamente ligada a processos industriais. Refere-se a um mecanismo ou sistema que opera praticamente sem a intervenção humana. Ele está diretamente ligado ao conceito de Indústria 4.0 ou das chamadas ”fábricas inteligentes”, integrando diferentes tecnologias para melhorar a gestão e aumentar a produtividade. Além de executar a tarefa que foi programada, o sistema de automação industrial pode aprender a melhor forma de desempenhá-la, acionar outros sistemas e fazer escolhas sem necessitar de ajuda humana. Ou seja, ele é capaz de analisar o próprio trabalho e tomar decisões sozinho.

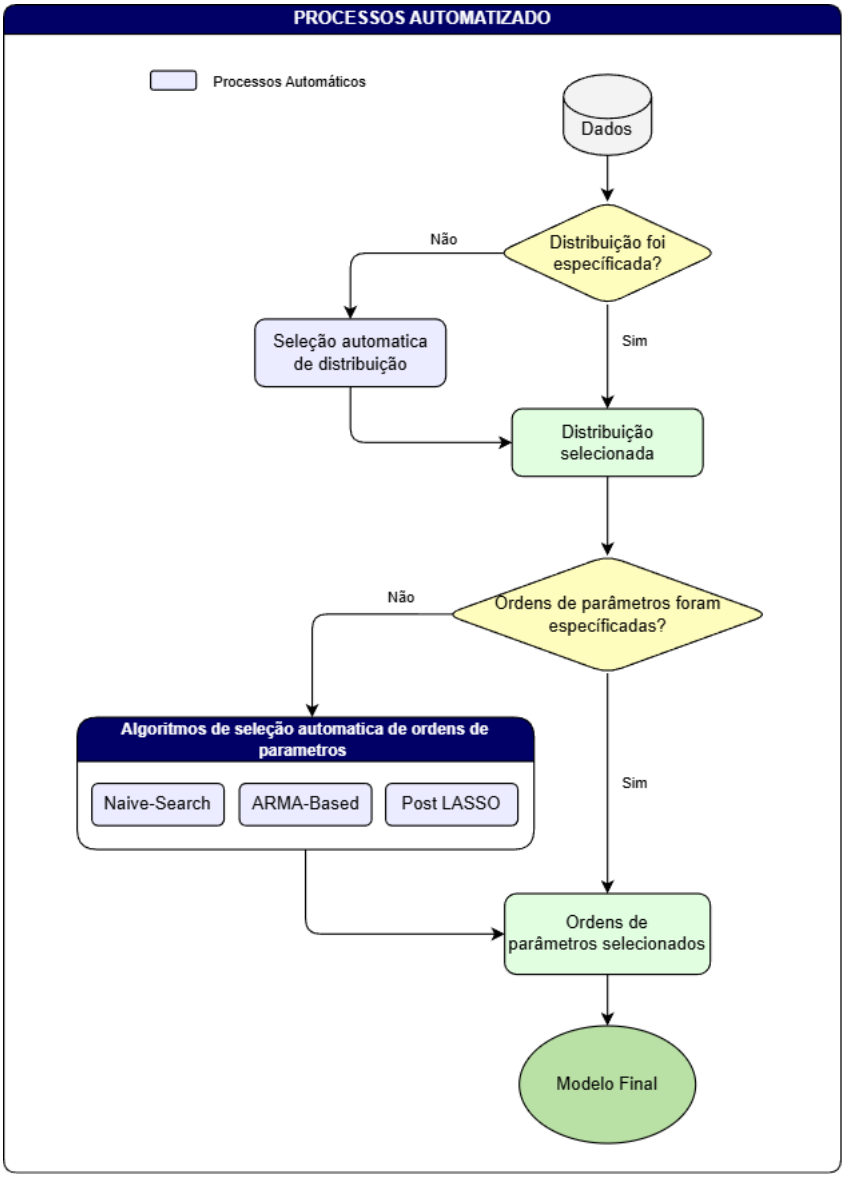
Já um mecanismo automatizado, refere-se a algo que foi configurado ou adaptado para operar de maneira automática. Este termo geralmente implica que um processo, tarefa ou sistema que anteriormente requeria intervenção manual foi modificado para funcionar de maneira autônoma. Ou seja, tal mecanismo ainda pode possuir intervenção humana em seu processo.

O pacote desenvolvido possui algoritmos para seleção automática de distribuição, para seleção automática de parâmetros e para busca de melhor modelo de previsão, porém a chamada desses não é obrigatória, permitindo ao pesquisador escolher métodos mais tradicionais de modelagem. Além disso, cada algoritmo possui diferentes métodos de serem abordados, necessitando de uma especificação inicial por parte do usuário. Assim, a palavra que melhor se encaixa no contexto estudado é a de ”automatização de processos”.

Apesar do método como um todo ser melhor descrito a partir da palavra ”automatizado”, cada passo de processo não possui a intervenção humana e desde sua concepção ele foi idealizado de tal forma. Assim, os nomes para cada parte do processo serão acompanhados da palavra ”automática”.

O seguinte fluxograma foi elaborado para facilitar o entendimento de tais definições, onde essas poderiam ser consideradas supérfluas ou delongadas, mas nas quais são necessárias para a plena construção de um tópico tão importante para o trabalho.

Fluxo Automizado fableCount



Elaborado pelo autor (2025)

Tal automatização se divide em 3 passos.

(i) **Seleção da Distribuição**

- Determina a distribuição de probabilidade a ser utilizada no modelo

(ii) **Seleção das Ordens Paramétricas**

- Determina as ordens autorregressivas e de médias móveis (p, q) do modelo utilizado. Certos métodos possuem a capacidade de determinação da ordem sazonal (P, Q)

(iii) **Seleção do Melhor Modelo de Previsão**

- Seleciona o modelo que minimiza alguma função de custo como EQM, MAE, MASE e outros

No contexto de automatização, a construção de suas funções deve ser realizada de uma maneira cautelosa, na qual o *trade-off* de desempenho e tempo de execução computacional deve ser realizado com equilíbrio

Algoritmos extremamente precisos exigem um alto desempenho computacional, contrariando uma das propostas iniciais do pacote em relação a rápida estimação de modelos. Além disso, certos limites computacionais são observados devido à busca exacerbada por precisão, onde a tentativa de uma ligeira melhoria na modelagem pode resultar em várias horas adicionais de execução computacional.

Já no outro extremo, algoritmos de execução instantânea muitas vezes apresentam baixo desempenho estatístico, pulando ou deixando de lado inúmeros passos fundamentais na busca pelo melhor modelo. Tal fato tornaria inútil tais algoritmos. Todo pesquisador gostaria de estimar rapidamente modelos para as 5568 cidades do Brasil, mas tal busca se torna absurda e infundada ao saber que os modelos estimados são imprecivos e errôneos

A partir do contexto apresentado, os métodos implementados no pacote buscam satisfazer ambos os lados, apresentando algoritmos otimizados que buscam um ótimo tempo de execução computacional, mas que para isso, não deixem de lado a precisão necessária para boas análises e previsões. Além disso, as seguintes seções buscaram descrever que o passo a passo desse processo automatizado pode ser personalizado, assim métodos mais precisos e mais longos ou menos precisos e mais rápidos, podem ser selecionados pelo usuário

4.1 ALGORITMO PARA SELEÇÃO AUTOMÁTICA DE DISTRIBUIÇÃO

Ambos os modelos implementados no pacote possuem como ponto de partida a seleção de uma distribuição de probabilidade adequada para os dados trabalhados. Diferentemente de um modelo ARMA, que supõe normalidade, os modelos GLARMA de Valores Inteiros e INGARCH podem utilizar as distribuições Poisson e Binomial Negativa como já citado no capítulo 2.

A idéia inicial dos algoritmos desenvolvidos para esse passo era a de testar se os dados trabalhados apresentavam o fenômeno de sobredispersão, onde a variância é significativamente maior que a média. A partir do teste aplicado, tería-se 2 possibilidades

1. Se **não** houve sobredispersão nos dados, a distribuição Poisson seria utilizada
2. Se houve sobredispersão nos dados, a distribuição Binomial Negativa seria utilizada

Apesar de estatisticamente o fenômeno de sobredispersão significar que para uma variável aleatória X, $Var(X) > E(X)$, sendo relativamente simples testar no contexto de testes para paramêtros, no contexto de uma regressão Poisson, que é o caso dos modelos GLARMA e INGARCH tem-se que a distribuição condicional dos dados segue certa distri-

buição de densidade de probabilidade, não sua distribuição marginal. Consequentemente, a variância será igual à média dentro de cada conjunto específico de valores das variáveis, mas não em todas elas. Em geral, a variância marginal será maior que a média marginal, mesmo quando os pressupostos da regressão de Poisson estiverem exatamente corretos (Cameron e Trivedi 1990). Ou seja, um simples teste de $E(X) = Var(x)$ não deve ser utilizado diretamente nos dados, e sim aplicados em modelos INGARCHs e GLARMAs já estimados.

A partir do ponto apresentado, o desenvolvimento de tal passo apresentou suas barreiras iniciais: Para testar sobredispersão, modelos iniciais devem ser estimados, mas a especificação de tais modelos no contexto de séries temporais ainda é desconhecido, onde a determinação dos parâmetros (p, q) se tornam necessários antes da definição da distribuição a ser utilizada no modelo. Tal fato implica em um alto custo computacional para a determinação de um passo que deveria ser o mais simples e rápido dentro os restantes.

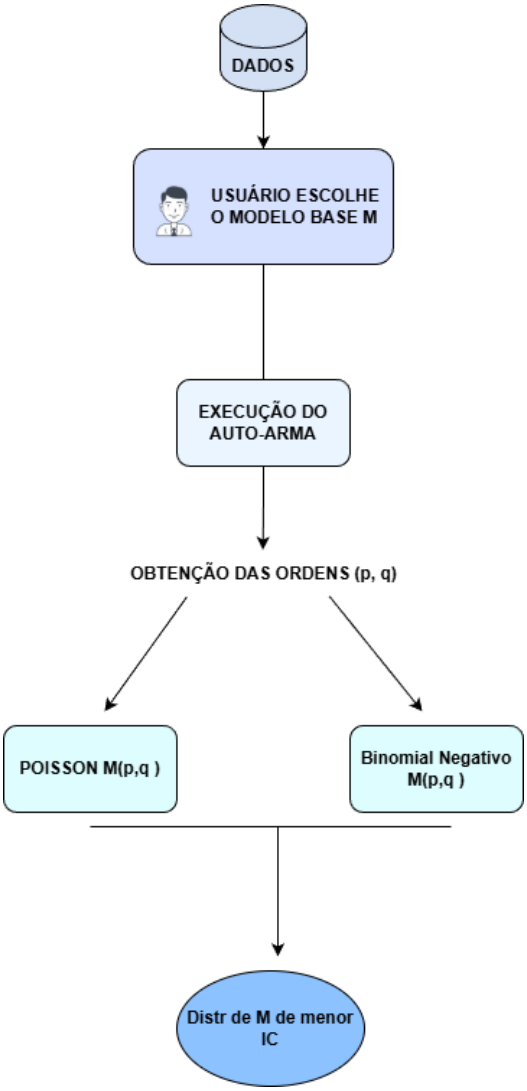
(Weiß e Schweer 2015) introduz a ideia de detecção de sobredispersão para modelos INARCH(1) (Heterocededasticidade Condicional Autorregressivo), que se desenvolve entorno da construção da distribuição assintótica do modelo para construção de um teste de hipótese. Apesar do teste apresentar um bom poder, sua aplicação nos modelos INGARCH e GLARMA é utópico no contexto desse trabalho, onde distribuições assintóticas para ambos os modelos teriam que ser construídas

Dado a impossibilidade do desenvolvimento de um teste apropriado de sobredispersão no contexto dos modelos estudados, (Weiß e Schweer 2015) descreve a boa precisão dos critérios de informação AIC e BIC selecionarem o modelo correto em casos assintóticos. Além disso, o mesmo autor cita a estrutura similar de autocorrelação entre o modelo ARMA e os modelos GLARMA e INGARCH. Assim, o algoritmo desenvolvido nesse passo se baseia na estimação de modelos Poisson GLARMA(p, q) e Binomial Negativo GLARMA(p, q) ou Poisson INGARCH(p, q) e Binomial Negativo INGARCH(p, q), onde as ordens (p, q) são obtidas via o algoritmo do **auto-arma**, selecionando o modelo que miniza certo critério de informação. Dessa forma, o problema de estimação de diversos modelos GLARMA ou INGARCH é eliminado através do algoritmo otimizado do **auto-arma**

O seguinte fluxograma foi elaborado para ilustrar esse passo (o objeto "m" sinalizado no fluxograma, vem de "modelo" e pode assumir os valores: INGARCH ou GLARMA)

O algoritmo **auto-arma** será abordado em detalhes no próximo tópico. De forma resumida, trata-se de um método para a seleção automática das ordens autorregressiva (AR, representada por p) e de média móvel (MA, representada por q) em modelos ARMA. Para isso, utiliza-se o algoritmo de **Hyndman-Khandakar**, que automatiza a escolha

Algoritmo de Seleção de Distribuição



Elaborado pelo autor (2025)

dessas ordens. As ordens selecionadas são então replicadas no ajuste de modelos de contagem (GLARMA e INGARCH)

Com a distribuição a ser utilizada no modelo já selecionada, o segunda passo da modelagem automatizada é disparada, onde as ordens dos parâmetros são definidas

4.2 ALGORITMOS PARA BUSCA AUTOMÁTICA DE ORDEM DE PARÂMETROS

Um obstáculo comum para muitas pessoas no uso dos modelos ARMAs para previsão é que o processo de seleção de ordens de parâmetros geralmente é considerado subjetivo e difícil de aplicar. (Hyndman e Khandakar 2008)

No contexto convencional de análise de séries temporais, a determinação das ordens dos parâmetros autorregressivos e de médias móveis envolve um processo exaustivo de investigação das funções de autocovariância (FAC) e autocorrelação parcial (FACP).

Este processo é muitas vezes caracterizado por sua natureza manual, onde o pesquisador deve avaliar minusiosamente lags significativos e não significativos na FAC e FACP. Em muitos casos, esse processo também é subjetivo, pois os sinais de truncamento e decaimento exponencial desejáveis nas funções não são facilmente identificáveis, tornando a interpretação dessas características uma tarefa passíveis de erros. Além disso, tal procedimento se mostra inviável no contexto de grandes bases de dados, onde por exemplo deseja-se estimar um modelo para cada cidade no Brasil. Assim, a motivação para o desenvolvimento de algoritmos de busca automática de parâmetros em modelos de séries temporais é evidente diante dos desafios apresentados pelo processo convencional de determinação das ordens dos parâmetros autorregressivos e de médias móveis

O pacote `fable` tem como uma de suas principais características possuir algoritmos para seleção automática de ordens de parâmetros em todos modelos disponibilizados. Cada modelo possui seu próprio algoritmo, tendo métodos de buscas e critérios de paradas distintos. Os mais importantes e que serviram de base para a elaboração dos algoritmos que serão descritos para os modelos de contagem são aqueles disponibilizados para a classe dos ARMAs (ARIMAs, SARIMAs, e ARMAX).

Nesse âmbito, o pacote desenvolvido nesse trabalho também disponibiliza formas automáticas de seleção de parâmetros. Tal tópico ainda se mostra inédito no contexto de modelos temporais de contagem, e portanto espera-se que o trabalho desenvolvido seja utilizado como ponto de partida para demais pesquisadores

O pacote `fableCount` implementa três algoritmos distintos para a seleção automática das ordens (p, q) nos modelos de séries temporais de contagem. Cada um desses algoritmos apresenta vantagens e limitações específicas, sendo concebidos para atender diferentes perfis de usuários e contextos analíticos. Os métodos disponíveis são:

- **Naive-Search**
- **ARMA-Based**
- **Post-Lasso**

Esses algoritmos podem ser agrupados em duas categorias principais, conforme proposto por (Hyndman e Khandakar 2008) e (Tran e Reed 2004):

1. **Algoritmos Atravessadores de Espaço de Parâmetros** (*Parameter Space Traversers Algorithms*)
2. **Algoritmos de Regularização** (*Regularization Algorithms*)

A distinção entre essas classes se dá pela natureza do processo de busca do modelo ideal:

- **Algoritmos de atravessamento de espaço de parâmetros** como o *Naive-Search* e o *ARMA-Based* consistem na estimação de múltiplos modelos, cada um com diferentes combinações de ordens (p, q). O critério de informação (AIC ou BIC) é utilizado para identificar o modelo mais adequado entre os candidatos estimados.

- **Algoritmos de regularização**, por outro lado, baseiam-se na aplicação de penalizações do tipo L_1 (como no método LASSO), promovendo a seleção automática das defasagens mais relevantes dentro de um modelo único de ordem máxima pré-definida. A estrutura final do modelo é então derivada a partir dos coeficientes distintos de zero, seguido de uma reestimação não penalizada (Post-LASSO).

Os tópicos subsequentes descrevem em detalhe o funcionamento e as particularidades de cada um dos métodos apresentados.

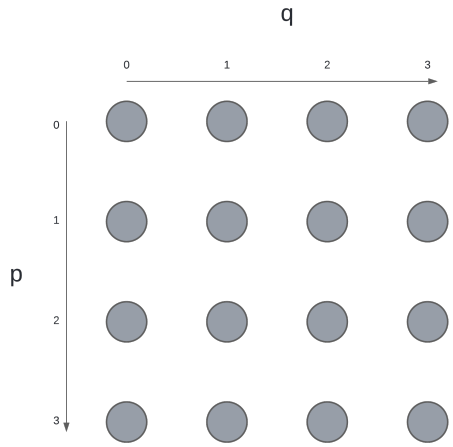
4.2.1 MÉTODO NAIVE-SEARCH

O método *Naive-Search* (em português, Busca Ingênua) considera uma matriz de busca 4x4, onde as linhas dessa matriz representam a ordem autorregresiva e as colunas representam a ordem de médias móveis. Todos os modelos de combinações (p, q) indo de 0 a 3 são estimados, e seus valores de AIC, ou BIC são armazenados nessa matriz. Ao final do algoritmo, a busca-se os indices dessa matriz que apresentam o menor valor do critério de informação utilizado.

De maneira detalhada, primeiramente o algoritmo define uma matriz 4x4, chamada de matriz de busca. O primeiro objetivo dessa matriz é indicar os indices dos modelos que devem ser estimados, onde suas linhas representam a ordem autorregresiva e as colunas representam a ordem de médias móveis, indo de 0 a 3 tanto em linhas tanto em colunas. Dessa forma, todas as combinações de modelos (p, q) para 0 a 3 são estimadas. Tal limite superior para as ordens dos parâmetros foi definido por conta das condições de estacionariedade e invertibilidade dos modelos, onde essas são necessária para a produção de previsões. A seguinte imagem foi criada exemplifica o primeiro passo do algoritmo, onde a matriz de busca é inicializada

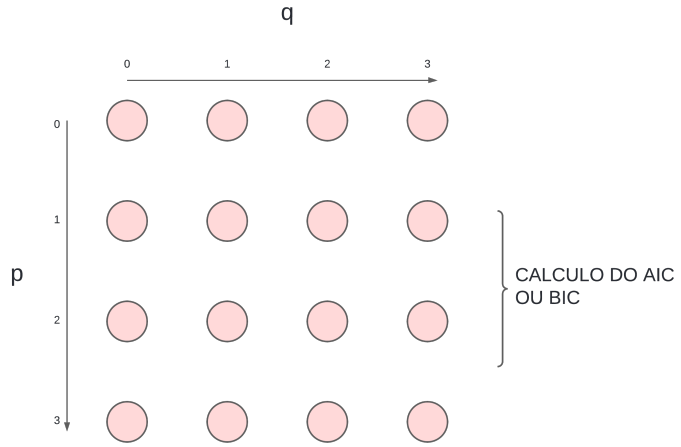
No segundo passo desse algoritmo, cada elemento dessa matriz passa a conter um modelo armazenado, onde por exemplo o elemento 1x1 armazenará um modelo INGARCH(0,0). O critério de informação definido pelo usuário, AIC OU BIC, são calculados para todos os modelos, onde esses valores são armazenados no lugar do modelo estimado, dessa forma a matriz de busca não possui mais o modelo em si, mas só o valor de critério informação. Essa decisão foi tomada por desempenho computacional, onde manipular uma matriz com valores numéricos é mais simples do que manipular uma matriz contendo objetos-modelos

Matriz de Busca no Passo 1 no Método Naive-Search



Elaborado pelo autor (2025)

Matriz de Busca no Passo 2 no Método Naive-Search



Elaborado pelo autor (2025)

No terceiro passo do algoritmo, os indices que apresentam o menor valor de critério de informação são obtidos e assim o modelo com menor AIC ou BIC é retornado

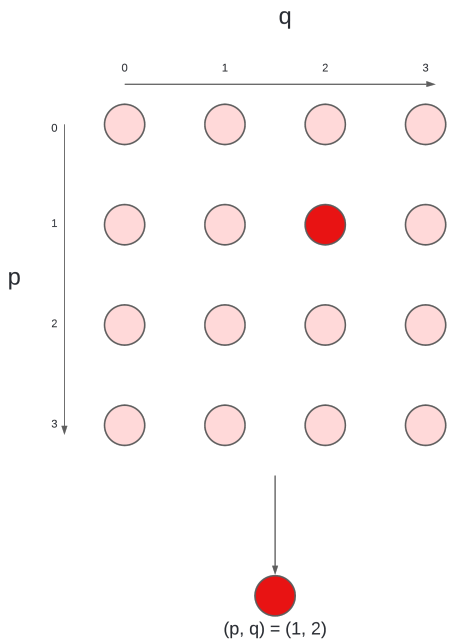
Na imagem a seguir, definiu-se um exemplo onde um modelo com (p, q) igual $(1, 2)$ foi aquele a apresentar menor critério de informação

Resumidamente, o método é dividido nos seguintes passos

1. Estimam-se todos os modelos correspondentes às combinações de ordens (p, q) com $p, q \in \{0, 1, 2, 3\}$ e tais modelos são armazenados na matriz de busca

2. Calcula-se, para cada modelo estimado, o critério de informação especificado pelo usuário (AIC ou BIC)

Matriz de Busca no Passo 3 no Método Naive-Search



Elaborado pelo autor (2025)

3. Retorna-se o modelo associado ao menor valor do critério de informação adotado, sendo este considerado o mais adequado dentre os avaliados.

O algoritmo possui esse nome por estimar todos os modelos do espaço paramétrico definido, não utilizando um caminho ótimo de busca como o método Stepwise. Apesar de ser um método com maior tempo de execução, ele sempre irá retornar o modelo de menor critério de informação. Esse método ainda não possui capacidade para busca de ordens sazonais. Tais características são as principais diferenças em relação ao próximo algoritmo apresentado

4.2.2 MÉTODO ARMA-BASED

O algoritmo *ARMA-Based* baseia-se no algoritmo de busca proposto por (Hyndman e Khandal para seleção automática de parâmetros para modelos ARMA. Diferente do algoritmo *Naive-Search*, Hyndman-Khandakar utilizam um método *Stepwise* para percorrer o espaço paramétrico, sendo chamado de *"A Stepwise procedure for traversing the model space"*. É importante destacar que tal método tem suporte para seleção de ordem de parâmetros sazonais. Ele possui os seguintes passos

- **Passo 0**: Um teste de Canova-Hansen é aplicado para verificar a presença de sazonalidade estável na série temporal.
- **Bifurcação do processo**: Com base no resultado do teste:

  - Se **não houver sazonalidade**, o processo segue a seleção de modelos não sazonais.
  - Se **houver sazonalidade**, adota-se a estratégia de seleção de modelos sazonais.

Para séries não sazonais:

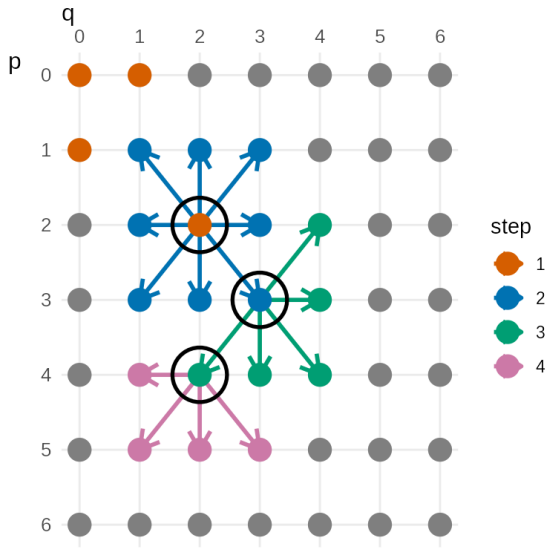
- **Passo 1**: Define-se um conjunto inicial de cinco modelos candidatos:
  - ARMA(0,0) com e sem constante
  - ARMA(1,0) com constante
  - ARMA(0,1) com constante
  - ARMA(2,2) com constante
- **Passo 2**: Calcula-se o critério de informação (AIC ou BIC) para os modelos definidos. O modelo com menor valor é armazenado.
- **Passo 3**: Realizam-se variações incrementais e decrementais de ± 1 nas ordens (p, q) do modelo selecionado, além da inclusão ou remoção da constante. O novo modelo é comparado com o anterior. Os passos 2 e 3 são repetidos até que não haja mais redução no critério de informação.

Para séries sazonais:

- A lógica é a mesma da seleção não sazonal, mas considera-se a estrutura SARMA. Os modelos iniciais incluem:
  - SARMA(0,0)$(0,0)_m$
  - SARMA(1,0)$(0,0)_m$
  - SARMA(0,1)$(0,0)_m$
  - ARMA(2,2) com constante
  - ARMA(0,0) sem constante
- A partir do melhor modelo inicial, aplicam-se novamente variações nas ordens e constantes, com repetição iterativa dos passos até que novo modelo não apresente um critério de informação inferior ao modelo atualmente em consideração.

A seguinte imagem foi elaborada por Hyndman para ilustrar o processo

Método Stepwise de Hyndman-Khandakar



Elaborado pelo autor (2025)

No escopo dos modelos de contagem e da implementação de estratégias automatizadas para seleção de parâmetros, adota-se o algoritmo de Hyndman e Khandakar como referencial para a determinação das ordens ótimas (p, q). Tal escolha é respaldada por (Weiß e Schweer 2015), o qual demonstra que modelos de contagem como GLARMA e INGARCH compartilham estruturas de autocorrelação similares àquelas observadas em modelos ARMA. Nesse sentido, a utilização das ordens identificadas a partir de um modelo ARMA convencional como ponto inicial para a modelagem de séries de contagem apresenta altas chances de obter um bom desempenho

4.2.3 MÉTODO VIA Post-LASSO

O método LASSO (Least Absolute Shrinkage and Selection Operator) foi proposto por Robert Tibshirani em 1996, com o objetivo de melhorar a interpretação e o desempenho de modelos de regressão quando há um grande número de variáveis explicativas. O LASSO se destaca por sua capacidade de realizar seleção de variáveis e regularização simultaneamente, tornando-se especialmente útil em cenários com alta dimensionalidade ou quando se deseja evitar o sobreajuste do modelo.

Em modelos lineares clássicos, como a regressão linear múltipla, a estimação dos coeficientes é feita por mínimos quadrados ordinários (OLS). Embora eficiente sob certas condições, o OLS tende a apresentar instabilidade quando o número de variáveis é grande ou quando existe multicolinearidade entre os regressores. O LASSO surge como uma alternativa que introduz uma penalização na função de custo, incentivando soluções mais parcimoniosas, ou seja, selecionando apenas variáveis relevantes

Formulação Estatística

Considere um modelo de regressão linear com vetor de resposta $\mathbf{y} \in \mathbb{R}^n$, e o vetor de coeficientes $\beta \in \mathbb{R}^p$. A estimação pelo método LASSO é dado por meio do seguinte problema de otimização

$$\hat{\beta} = \arg\min_{\beta} \left\{ \frac{1}{2n} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda \|\beta\|_1 \right\}, \tag{4.1}$$

onde $\|\cdot\|_2$ denota a norma Euclidiana e $\|\cdot\|_1$ denota a norma l_1 definida como $\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$. O parâmetro de regularização $\lambda \geq 0$ controla o grau de penalização: quanto maior λ, maior o encolhimento dos coeficientes, podendo forçar alguns deles a zero o que efetivamente realiza a seleção de variáveis.

Embora o LASSO tenha sido inicialmente desenvolvido para modelos de regressão linear, sua aplicação foi posteriormente estendida para contextos mais complexos, incluindo modelos autoregressivos e séries temporais de contagem. Nesses casos, o LASSO pode ser utilizado para selecionar automaticamente os defasagens mais relevantes dos regressores e da própria série, o que é particularmente útil quando se trabalha com modelos como INGARCH ou GLARMA, onde a estrutura de defasagens pode ser extensa.

Além do LASSO tradicional, outras variantes foram desenvolvidas para aprimorar ainda mais a seleção de variáveis e lidar com limitações do método original. O Adaptive LASSO (AdaLASSO), proposto por (Zou 2006), introduz pesos adaptativos na penalização l_1, permitindo que variáveis com maior relevância inicial recebam penalizações menores. Essa abordagem melhora a consistência na seleção de variáveis, sendo útil em contextos onde se deseja recuperar a estrutura verdadeira do modelo, especialmente em séries temporais com defasagens longas. Mais recentemente, o Weighted Lag Adaptive LASSO (WLadaLASSO) foi proposto para cenários específicos de séries temporais, incorporando pesos distintos para diferentes defasagens e adotando funções perda robustas como a LAD (Least Absolute Deviations). Essa metodologia mostrou-se particularmente eficaz na presença de sobredispersão e heterocedasticidade.

Contudo, neste trabalho optamos por adotar o enfoque do **Post-LASSO**, dada a natureza do problema em questão: nosso principal objetivo é selecionar automaticamente as defasagens mais relevantes dos termos autoregressivos (AR) e de média móvel (MA), sem a necessidade de reestimar os coeficientes a partir da estrutura penalizada do LASSO. Assim, utilizamos o LASSO como ferramenta de triagem de variáveis e, uma vez selecionadas as defasagens, reestimamos os parâmetros por métodos tradicionais, como o estimador de máximo verossimilhança, respeitando a estrutura probabilística do modelo de contagem escolhido. Essa abordagem se mostrou adequada para o nosso objetivo de análise estrutural e interpretabilidade do modelo, mantendo o viés de estimação sob controle ao empregar estimadores não penalizados na etapa final.

A abordagem **Post-LASSO** consiste em um procedimento de duas etapas: seleção de defasagens relevantes por meio da penalização LASSO e reestimação de um modelo

final com apenas as variáveis selecionadas. A seguir, descrevem-se as etapas da estratégia:

1. **Inicialização** - Inicialmente, é necessário definir um conjunto inicial de defasagens autorregressivas e de médias móveis. Para isso, ajusta-se um modelo INGARCH ou GLARMA preliminar, com ordens máximas (p_0, q_0), sendo:

- Defasagens autorregressivas: $y_{t-1}, y_{t-2}, ..., y_{t-p_0}$
- Defasagens de médias móveis: $e_{t-1}, e_{t-2}, ..., e_{t-q_0}$

  Essas defasagens serão utilizadas como variáveis explicativas no modelo LASSO

2. **Estimação via LASSO** - Nesta etapa, ajusta-se um modelo de regressão penalizada LASSO, onde a variável resposta é a própria série temporal y_t, e as variáveis explicativas são as defasagens selecionadas anteriormente. O modelo ajustado tem a seguinte forma:

$$y_t = \beta_0 + \sum_{i=1}^{p_0} \beta_i y_{t-i} + \sum_{j=1}^{q_0} \gamma_j e_{t-j} + \varepsilon_t \tag{4.2}$$

com a penalização l_1 aplicada aos coeficientes β_i e γ_i, que promove a seleção automática de variáveis relevantes, defasagens com coeficientes iguais a zero são descartadas

3. **Seleção final e reestimação (Post-LASSO)** - Após a etapa de penalização, extrai-se o último coeficiente não nulo estimado pelo LASSO para os termos AR e MA. Assim, definem-se as ordens finais do modelo como:

- $\hat{p} = \max\{i : \hat{\beta}_i \neq 0\}$
- $\hat{q} = \max\{i : \hat{\gamma}_i \neq 0\}$

Com os valores de $\hat{p}$ e $\hat{q}$ selecionados, reestima-se o modelo de séries temporais (por exemplo, INGARCH ou GLARMA), utilizando o método usual de estimação para cada um dos modelos, utilizando apenas as defasagens retidas pelo LASSO.

4.2.4 COMENTÁRIOS SOBRE OS MÉTODOS

Diferentemente dos métodos *Naive-Search* e *ARMA-based*, que se baseiam na estimação de diversos modelos e na seleção daquele que minimiza um critério de informação, como o AIC ou o BIC, o método baseado em LASSO estima apenas um modelo final. Apesar dessa vantagem de simplicidade, trata-se de uma abordagem que pode apresentar maior instabilidade computacional, uma vez que depende de métodos numéricos sujeitos à não convergência e da escolha adequada do parâmetro de penalização λ

Cada método possui seus respectivos pontos fortes e limitações, sendo a escolha do mais adequado determinada pelas características dos dados e pelos objetivos do estudo.

A Tabela 1 apresenta uma comparação entre os métodos implementados neste trabalho e pode servir como um guia de referência rápida para a escolha da abordagem mais apropriada.

Tabela 1 – Comparação entre os métodos de seleção de ordens (p, q)

Critério	**Naive-Search**	**ARMA-Based**	**Post-LASSO**
Tipo de busca	Exaustiva	Stepwise guiada	Regularização (L1)
Espaço explorado	Completo (ex.: $p, q \leq 3$)	Parcial, guiado por heurística	Parcial, definido por penalização
Custo computacional	Alto	Baixo	Moderado
Estabilidade da estimação	Alta	Alta	Moderada
Suporte à sazonalidade	Não	Sim	Não
Reestimativa final	Não necessária	Não necessária	Sim (pós-seleção)
Objetivo principal	Precisão na escolha global	Eficiência com boa precisão	Seleção parcimoniosa e adaptativa
Dependência de tuning	Não	Parcial (critérios)	Alta (valor de λ)
Recomendado para	Séries curtas	Séries sazonais e modelagem exploratória	Séries longas ou alta dimensionalidade

4.3 ALGORITMO PARA BUSCA DE MELHOR PREVISÃO

Uma questão recorrente na área de estatística e aprendizado de máquina é a diferenciação de modelos com foco em inferência daqueles com foco em predição/previsão. Enquanto a primeira classe busca a construção de modelos parsimoniosos, buscando a interpretação das variáveis explicativas utilizadas, onde os critérios de informação são ferramentos importantíssimas nesse contexto, a segunda classe de modelos busca exaustivalmente o modelo de menor erro de predição/previsão, deixando em segundo plano questões como um elevado número de covariávies e baixo poder de interpretação. Deve-se destacar que no contexto de séries temporais, é mais adequado a utilização do termo"previsão" em relação ao termo "predição", onde esse tem como significado: a antecipação de algo, a construção daquilo que ainda não aconteceu

Como ja destacado, os critérios AIC e BIC são essenciais para a primeira classe, pois esses se baseiam em uma função da verossimilhança penalizada pelo número de parâmetros. Tais critérios já foram definidos e utilizados em seções anteriores. No outro extremo, para modelos com foco em previsão, outras métricas são utilizadas, onde essas utilizam como ponto inicial o **erro** do modelo estimado.

Um erro de previsão é a diferença entre o valor observado e o valor previsto, e pode ser escrito como

$$e_{t+h} = y_{t+h} - \hat{y}_{t+h|t} \tag{4.3}$$

A análise em cima do erro de previsão por si só não fornece informações suficientes sobre a distribuição e a natureza dos erros. Ao utilizar diferentes métricas, é possível obter uma visão mais abrangente e detalhada do desempenho do modelo, identificando áreas específicas que necessitam de melhorias e permitindo a escolha da métrica mais adequada para o contexto específico da aplicação. É nesse cenário que diversos tipos métricas são utilizadas, onde essas estão disponíveis no pacote, tais métricas também são chamadas de "métricas de avaliação" *(evaluation metrics)*

4.3.1 MÉTRICAS DE AVALIAÇÃO

Diferentes métricas são utilizadas em diferentes cenários. Desde métricas livres de escala, a métricas que buscam penalizar maiores erros, o pacote desenvolvido nesse trabalho buscou disponilizar a maior gama possível para que o usuário pudesse personalizar a escolha do melhor modelo da maneira que o agradasse.

A primeira classe de métricas são as mais simples, e chamadas de **Erros Dependentes de Escala**. Essas métricas utilizam a mesma escala que os dados trabalhados, e portanto não podem ser utilizadas para comparar séries que possuam diferentes escalas

É importante destacar que no pacote tais métricas são chamadas em ingles.

$$\text{MAE - Mean Absolute Error} = \text{média}(|e_t|) \tag{4.4}$$

$$\text{RMSE - Root Mean Squared Error} = \sqrt{\text{média}(e_t^2)} \tag{4.5}$$

Enquanto o MAE possui uma interpretação mais direta e simples, o RMSE busca penalizar modelos que possuem erros de maiores magnitudes. Ambas métricas são popularmente utilizadas e possuem pouca diferença em tempo de execução computacional

A segunda classe de métricas não dependem da escala dos dados e são chamados de **Erros percentuais**. Usualmente, se baseiam no valor de p_t, onde $p_t = \frac{100e_t}{y_t}$, e são utilizados para comparação de séries com diferentes unidades

A métrica mais usual é dada por

$$\text{MAPE - Mean Absolute Percentage Error} = \text{média}(|p_t|) \tag{4.6}$$

Por sua definição, o MAPE possui a desvantagem de atribuir um peso maior a erros negativos do que positivos, tornando-se uma métrica assimétrica. Buscando contornar tal característica, (Genetski 1978) propôs a métrica MAPE Simétrico, dado por

$$\text{sMAPE(Symmetric MAPE)} = \text{média}(\frac{200|y_t - \hat{y}_t|}{(y_t + \hat{y}_t)}) \tag{4.7}$$

Apesar de ambas as métricas apresentarem vantagens para comparações de séries de diferentes unidades, ambas apresentam valores indefinidos ou tendendo ao infinito para y_t próximos a 0, caso comum para dados de contagem. Além disso, o sMAPE pode ter valores negativos, e portanto em certos casos não pode ser considerado uma verdadeira métrica de **Erros percentuais absolutos**

A terceira e última classe de métricas disponibilizadas são os chamados **Erros Dimensionados**. Sendo propostos por (Hyndman e Koehler 2006) tal classe é uma alternativa ao Erros Percentuais, podendo ser utilizada para comparar a precisão de modelos em séries com diferentes unidades. O intuito geral do método é dimensionar o erro baseado no MAE de treinamento do modelo.

Para séries não sazonais, o erro dimensionado é definido por

$$q_j = \frac{e_j}{\frac{1}{T-1}\sum_{t=2}^{T}|y_t - y_{t-1}|} \tag{4.8}$$

Já para séries sazonais, tem-se que

$$q_j = \frac{e_j}{\frac{1}{T-m}\sum_{t=m+1}^{T}|y_t - y_{t-m}|} \tag{4.9}$$

onde m indica o indice sazonal da série

A partir da definição de q_j, as métricas da classe **Erros Dependentes de Escala** são construidas novamente

$$\text{MASE - Mean Absolute Scaled Error} = \text{média}(|q_j|) \tag{4.10}$$

$$\text{RMSSE - Root Mean Squared Scaled Error} = \sqrt{\text{média}(q_j^2)} \tag{4.11}$$

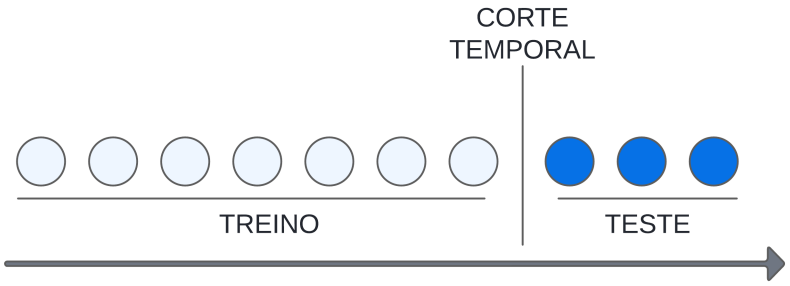
4.3.2 MÉTODOS DE AVALIAÇÃO DE DESEMPENHO PREDITIVO DE SÉRIES TEMPORAIS

Para a avaliação do erro de um modelo, é necessário que esse seja calculado a partir de um conjunto de dados que não tenha sido utilizado em sua estimação. Nesse contexto, a técnica mais conhecida é de divisão do conjunto de dados disponível em 2 partições: **treino e teste**. Enquanto a primeira é utilizada para estimação dos parâmetros do modelo, a segunda tem o objetivo testar sua precisão e acurácia. Na área de regressão e aprendizado de máquina, a técnica mais simples se baseia na divisão de 75% dos dados para a base de treino e 25% para a base teste, onde uma amostra aleatória simples é realizada para essa partição.

Para séries temporais, uma simples amostra aleatória simples não pode ser utilizada por conta da correlação temporal dos dados. Para contornar esse problema, utiliza-se

a técnica chamada Fora da Amostra (Out-of-Sample, OOS). Nessa abordagem, define-se uma data de corte na série temporal; todos os valores anteriores a essa data são utilizados para o treinamento do modelo, enquanto todos os valores posteriores são usados para teste. Dessa forma, preserva-se a estrutura temporal dos dados, garantindo uma avaliação mais realista do desempenho do modelo em situações futuras.

Método OOS para Avaliação de Modelos Temporais



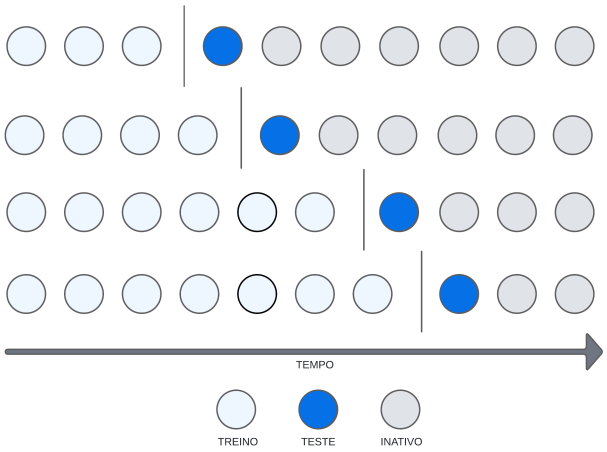
Elaborado pelo autor (2025)

Essa abordagem tem os benefícios de possuir um rápido tempo de execução computacional e apresentar um bom desempenho de como o modelo construído é capaz de generalizar em dados futuros. Apesar disso, tal técnica tem como principal desvantagem a perda de observações para o treinamento do modelo, no qual uma vez que os dados são divididos, a parte de teste se torna oculta na seção de treinamento.

Buscando contornar esse problema, uma abordagem cada vez mais utilizada em aprendizado de máquina é o chamado **K-folds Cross Validation** onde o conjunto de dados é divido em k subconjuntos, onde executa-se k iterações, na qual ao final da execução do algoritmo, todos os dados são utilizados para o treino e para o teste do modelo.

Novamente para séries temporais, a aplicação de tal abordagem se mostra um pouco diferente por conta da correlação temporal entre os dados. Assim um método mais robusto quando comparado ao OSS é o chamado **Time Series Cross Validation** ou **Evaluation on a Rolling Forecasting Origin**. Essa técnica se baseia na criação sucessivos OOS, onde cada novo OOS, há um aumento dos dados de treino

Método Time Series Cross Validation



Elaborado pelo autor (2025)

Tais definições são importantíssimas para o entendimento do tópico, onde tais técnicas são utilizadas para o cálculo das métricas de erro dos modelos construídos

4.3.3 BUSCANDO E AVALIANDO O MELHOR MODELO

Diferentemente das etapas de busca de melhor distribuição e de melhores ordens de parâmetros que utilizam algum critério de informação para retornar o modelo final, a definição do melhor modelo de previsão se define em sua maior parte na busca do modelo a apresentar menores métricas de avaliação

Para facilitar a escrita e tornar a leitura do tópico mais fluida, utilizaremos a abreviação **ABMMP - (Algoritmo de Busca de Melhor Modelo de Previsão)**.

O algoritmo desenvolvido buscou ponderar da melhor forma o *trade-off* de precisão e desempenho computacional. É importante destacar que os métodos que calculam as métricas de avaliação, seja um simples OSS ou um Time Series Cross Validation, demandam um maior tempo de execução computacional que um cálculo de AIC ou BIC. Assim, a idéia base do algoritmo é filtrar alguns dos principais modelo a partir de algum critério de informação fornecido pelo usuário, calcular suas métricas de avaliação e retornar o modelo que minimiza tais valores.

O ABMMP é externo aos demais passos ja descritos, ou seja, os passos de busca de distribuição e ordem de parâmetros conversam entre si obrigatoriamente (dado que o usuário não tenha especificado nenhum desses 2 pontos), porém o passo descrito no tópico atual utiliza uma estratégia diferente das demais

Dado que os algoritmos descritos anteriormente sempre retornavam um modelo ao final de sua execução, se o ciclo continuasse nesse último passo, ele não realizaria uma busca verdadeira, mas apenas calcularia as métricas de avaliação do modelo retornado no

passo 2. Assim, o ABMMP utiliza algoritmos semelhantes aos vistos nos passos 1 e 2, porém com modificações específicas desenvolvidas para satisfazer a parte de busca.

Para chamar o ABMMP, o usuário deve especificar 3 argumentos:

(i) Método de Busca de Ordem de Parâmetros. - Deve-se escolher entre *Naive-Search-Forecast* ou *Tri-EVAL*. Ambos os métodos serão detalhados na próxima seção

(ii) Método de Cálculo da Métrica de Avaliação - Deve-se escolher entre os métodos OSS ou Time Series Cross Validation

(iii) Métrica de Avaliação que deve ser Minimizada - Deve-se escolher apenas 1 métrica dentre as definidas na subseção 4.3.1.

4.3.3.1 BUSCA DE ORDEM DE PARÂMETROS

Os métodos aqui apresentados são apenas modificações daqueles já descritos na seção 4.2 (Algoritmos para busca automática de ordem de parâmetros). As modificações encontram-se principalmente na parte final do algoritmo.

É fundamental destacar que os modelos que apresentam os menores critérios de informação, como AIC ou BIC, não são necessariamente aqueles com as melhores métricas de avaliação de previsão. No entanto, esses modelos geralmente estão próximos dos modelos ótimos para previsão. Em outras palavras, embora os modelos que minimizam o AIC ou BIC não sejam sempre os melhores para previsão, a probabilidade de se encontrar o modelo ótimo é maior na vizinhança desses modelos. Portanto, a base dos algoritmos descritos nesta seção é utilizar algum método de seleção automática de ordens de parâmetros, conforme apresentado na seção 4.2, para selecionar um modelo inicial. Em seguida, testamos os modelos vizinhos para identificar aqueles que apresentam melhores métricas de avaliação.

4.3.3.1.1 NAIVE-SEARCH-FORECAST

O método *Naive-Search-Forecast* possui como idéia principal utilizar o algoritmo *Naive-Search* para buscar o modelo que apresenta uma melhor métrica de avaliação, diferente do método usual, que busca o modelo com o menor valor de AIC e BIC

A ideia central consiste em ranquear os modelos com base em um critério de informação, de forma que o modelo melhor posicionado seja considerado como ponto de partida. A partir desse ranking, realiza-se uma comparação sequencial: o modelo na posição i é comparado com o modelo na posição $i+1$. O processo é interrompido quando o modelo na posição i apresenta uma métrica de avaliação inferior (melhor desempenho) em relação ao modelo $i+1$, adotando-se esse ponto como critério de parada.

O algoritmo pode ser dividido nos seguintes passos

1. Execução do Naive-Search (até o passo 2) - O algoritmo inicia com a execução do método Naive-Search, limitado até a etapa de cálculo dos critérios de informação (AIC ou BIC) para cada modelo gerado na matriz de busca. Nesse estágio, não há ainda avaliação preditiva, apenas a estimativa dos modelos e a quantificação de sua complexidade e ajuste via critérios clássicos.

2. Rankeamento dos modelos - Com os valores de AIC ou BIC calculados, os modelos são ordenados de forma crescente ou seja, do menor para o maior valor do critério de informação. Essa ordenação define a prioridade de comparação entre os modelos, sendo o modelo com menor AIC/BIC considerado o mais promissor inicialmente.

3. Comparação sequencial com base em métrica preditiva - Inicia-se a avaliação sequencial dos modelos rankeados. Compara-se a métrica de desempenho preditivo (por exemplo, RMSE ou MAPE) entre o primeiro e o segundo modelo do ranking. Caso o primeiro modelo apresente desempenho superior, o algoritmo é encerrado, adotando-o como modelo final. Caso contrário, a comparação prossegue entre o segundo e o terceiro modelo, repetindo o processo até que um modelo apresente desempenho melhor que seu sucessor direto. Esse ponto define o critério de parada do algoritmo.

4.3.3.1.2 Tri-EVAL

Executa os 3 métodos de busca automática de ordem de parâmetros Naive-Search, ARMA-Based e Post-LASSO e retorno aquele modelo que possui a combinação de ordens P e Q que apresenta a menor métrica de avaliação (melhor desempenho)

O algoritmo pode ser dividido nos seguintes passos

1. Execução dos métodos de busca automática de ordem de parâmetros - Executa os 3 métodos (Naive-Search, ARMA-Based e Post-LASSO), onde cada método retorna um modelo final

2. Comparação entre modelos - Compara-se a métrica de desempenho preditivo (por exemplo, RMSE ou MAPE) entre os 3 modelos obtidos no passo 1. O modelo que apresentar o melhor desempenho é aquele retornado pela função

Esse passo ainda possui certas limitações como.

(i) O modelo não pode conter covariáveis, mesmo que essas possuirem valor em um tempo futuro, tornando teoricamente previsões possiveis

(ii) A construção de previsões maiores que 1 passo a frente para o modelo INGARCH utiliza o método de bootstrap paramétrico, técnica essa que introduz certo nível de viés e variância quando o método de Times Series Cross Validation é utilizado para cálculo das métricas de avaliação

5 APLICAÇÃO E RESULTADOS

Esta seção tem por finalidade aplicar os modelos de contagem desenvolvidos a um conjunto de dados reais, de modo a avaliar sua capacidade preditiva e computacional em comparação ao modelo ARIMA e NNETAR. Para essa análise, foram utilizados dados da pandemia de COVID-19 no Brasil, disponibilizados pelo DATASUS, consistindo em séries temporais semanais do número de casos e óbitos confirmados.

Além da comparação entre classes de modelos, foi analisado a efetividade dos procedimentos automáticos de seleção da ordem autorregresiva e de médias móveis dos modelos de contagem, com o objetivo de verificar se tais estratégias produzem resultados competitivos ou mesmo superiores aos obtidos pelo ARIMA tradicional e pelo NNETAR.

Como enfatizado na introdução, os principais desafios desse tipo de modelagem residem na adequada representação de séries de baixa magnitude (com valores frequentemente próximos de zero) e na eficiência computacional durante a estimação e previsão.

5.1 DADOS UTILIZADOS

Para a presente aplicação, foram utilizados dados sobre a COVID-19 disponibilizados pelo DATASUS, portal oficial do governo federal.

A base contém informações de todos os municípios brasileiros, abrangendo o período de 2020 a 2025, com atualização diária. Considerando a necessidade de maior controle sobre o escopo da análise e buscando adequar o estudo ao contexto geográfico e temporal de interesse, optou-se por aplicar um filtro, selecionando apenas municípios do estado de Minas Gerais, referentes ao ano de 2024.

O estudo teve como objetivo modelar tanto os casos confirmados de COVID-19 quanto os óbitos registrados. Para isso, foram selecionados 39 municípios mineiros para compor o conjunto de dados de casos confirmados e 35 municípios para o conjunto de dados de óbitos.

Para a seleção desses municípios, realizamos uma amostragem estratificada a partir dos 853 municípios de Minas Gerias, com base no número total de casos confirmados e óbitos ao longo do ano. Foram definidos três estratos para cada variável.

Os estratos foram definidos a partir de uma análise exploratória dos dados originais, resultando na seguinte classificação:

Tabela 2 – Estratificação das variáveis de casos confirmados e óbitos

Tipo de variável	Categoria do estrato	Valor mínimo	Valor máximo
Casos confirmados	Baixo	10	100
Casos confirmados	Médio	100	250
Casos confirmados	Alto	250	—
Óbitos	Baixíssimo	1	4
Óbitos	Baixo	5	9
Óbitos	Médio	10	20
Óbitos	Alto	20	—

Dessa forma, foi possível identificar em quais contextos os modelos apresentaram melhor desempenho, possibilitando uma comparação mais equilibrada entre eles.

A base final utilizada para o treinamento passou por um ajuste no índice temporal das séries. Na base original, as observações eram registradas diariamente; contudo, considerando que os dados diários podem apresentar inconsistências e incoerências devido a atrasos ou atualizações no sistema por parte do governo, optou-se por agregá-los por semana epidemiológica. Com isso, cada município passou a ter 52 observações anuais no conjunto de dados empregado para a modelagem.

A etapa de treinamento consistiu na definição e estimação dos modelos a serem comparados. Foram consideradas duas classes principais: (1) os modelos atualmente empregados na plataforma, representados por **ARMA** e **NNETAR**; e (2) os modelos de contagem desenvolvidos neste trabalho, representados por **GLARMA** e **INGARCH**.

O objetivo dessa etapa foi garantir uma base de comparação justa entre os modelos clássicos e as novas implementações voltadas para séries temporais de contagem, permitindo avaliar tanto o desempenho preditivo quanto os aspectos computacionais de cada abordagem.

O modelo **ARMA** foi ajustado por meio do procedimento stepwise de seleção automática de ordens, conhecido como método de HyndmanKhandakar (Hyndman e Khandakar 2008).

O modelo **NNETAR** foi estimado com base nas rotinas automatizadas de seleção de arquitetura e ordens disponíveis na função `nnetar()` do pacote `fable`. Essa função implementa uma rede neural autoregressiva. O processo inclui a escolha automática do número de defasagens e neurônios na camada oculta, conforme a metodologia proposta por **(author?)** (Hyndman e Athanasopoulos 2021).

Para o modelo **GLARMA**, foram definidos os métodos *Naive-Search* e *ARMA-Based*

De forma análoga, o modelo **INGARCH** foi estimado em três variações, *Naive-Search*, *ARMA-Based* e *Naive-Search* e *Post-LASSO*

A combinação desses diferentes métodos permitiu comparar não apenas o desem-

penho preditivo dos modelos de contagem, mas também a eficiência e robustez de cada estratégia de seleção de parâmetros, em um contexto de ajuste em larga escala.

Tanto para conjunto de dados de casos confirmados e óbitos, os modelos foram treinados e avaliados de duas maneiras distintas, com o objetivo de comparar desempenho em diferentes horizontes de previsão:

1. **Validação temporal (time-series cross-validation)** este procedimento de validação com origem móvel (rolling-origin) foi utilizado para avaliar a acurácia dos modelos em previsões de *um passo à frente* (horizonte mínimo). Essa abordagem permite medir o desempenho em condições de predição curta e verificar estabilidade das previsões ao longo do tempo.

2. **Avaliação out-of-sample (OOS)** aqui os modelos foram treinados em um conjunto de treino fixo e avaliados em um conjunto fora da amostra para previsões de *quatro passos à frente*. No contexto deste estudo, em que a unidade temporal são semanas epidemiológicas, um horizonte de quatro passos corresponde a um mês completo, sendo considerado um horizonte de previsão relativamente longo.

Ambas as estratégias foram adotadas para investigar se a acurácia relativa dos modelos depende do horizonte de previsão. A métrica principal utilizada para comparação é o **RMSE**

5.2 HIPÓTESES

Para orientar a análise empírica e garantir que os resultados obtidos não se dispersassem em observações fragmentadas ou interpretações meramente descritivas, optou-se por estabelecer previamente um conjunto de hipóteses a serem investigadas. A definição dessas hipóteses fornece um caminho estruturado para a análise, permitindo que a investigação se desenvolva de forma organizada e consistente, em vez de ficar à mercê de achados isolados ou desconexos. Com isso, busca-se otimizar a investigação, tornando-a mais clara, eficiente e replicável, além de possibilitar uma interpretação dos resultados dentro de um quadro analítico coerente.

O presente trabalho adota uma abordagem exploratória e comparativa para avaliação das hipóteses, sem recorrer a testes estatísticos formais de significância. As análises foram conduzidas de forma descritiva, tendo o RMSE como métrica principal para mensurar a acurácia dos modelos. A verificação das hipóteses foi realizada por meio da comparação direta dos resultados obtidos, buscando identificar padrões de desempenho nos diferentes estratos de séries temporais, no tempo de execução e nas características específicas das previsões, com ênfase na interpretação prática e no comportamento relativo entre as abordagens.

Com base nos dados e modelos definidos, foram formuladas hipóteses fundamentadas tanto na experiência empírica de utilização dos modelos ARIMA e NNETAR quanto no objetivo de desenvolver e avaliar modelos de contagem. Cada hipótese descreve um cenário esperado de desempenho ou característica dos modelos, sendo avaliada a partir dos valores de RMSE obtidos nas previsões e de observações qualitativas sobre os resultados.

A seguir, temos as hipóteses desenvolvidas, assim como a técnica utilizada para analisar essas hipóteses.

1. **Modelos de contagem apresentam um erro menor para estratos "Baixos" e "Baixíssimo"** Como descrito durante boa parte do trabalho, a construção dos modelos de contagem foram motivados pela necessidade da melhoria do desempenho em dados com valores próximos a 0 e assim é necessário analisar esse comportamento. A avaliação é dada calculando-se o RMSE médio dos modelos para todas as séries pertencentes ao estrato de baixa contagem. Os valores serão comparados entre modelos de contagem e modelos clássicos (ARIMA e NNETAR) para verificar qual grupo apresenta melhor desempenho nesse cenário.

2. **Modelos de contagem exigem menor tempo de treinamento do que NNETAR (por série), tornando-os mais escaláveis para muitos municípios.** O tempo de execução para ajuste de cada modelo será registrado individualmente por série. A comparação será feita analisando-se o tempo médio por estrato de município, analisando de maneira apartada os tempos de treinamento e previsão de cada modelo. Esperamos que os modelos de contagem tenham um tempo total de execução menor que o modelo NNETAR

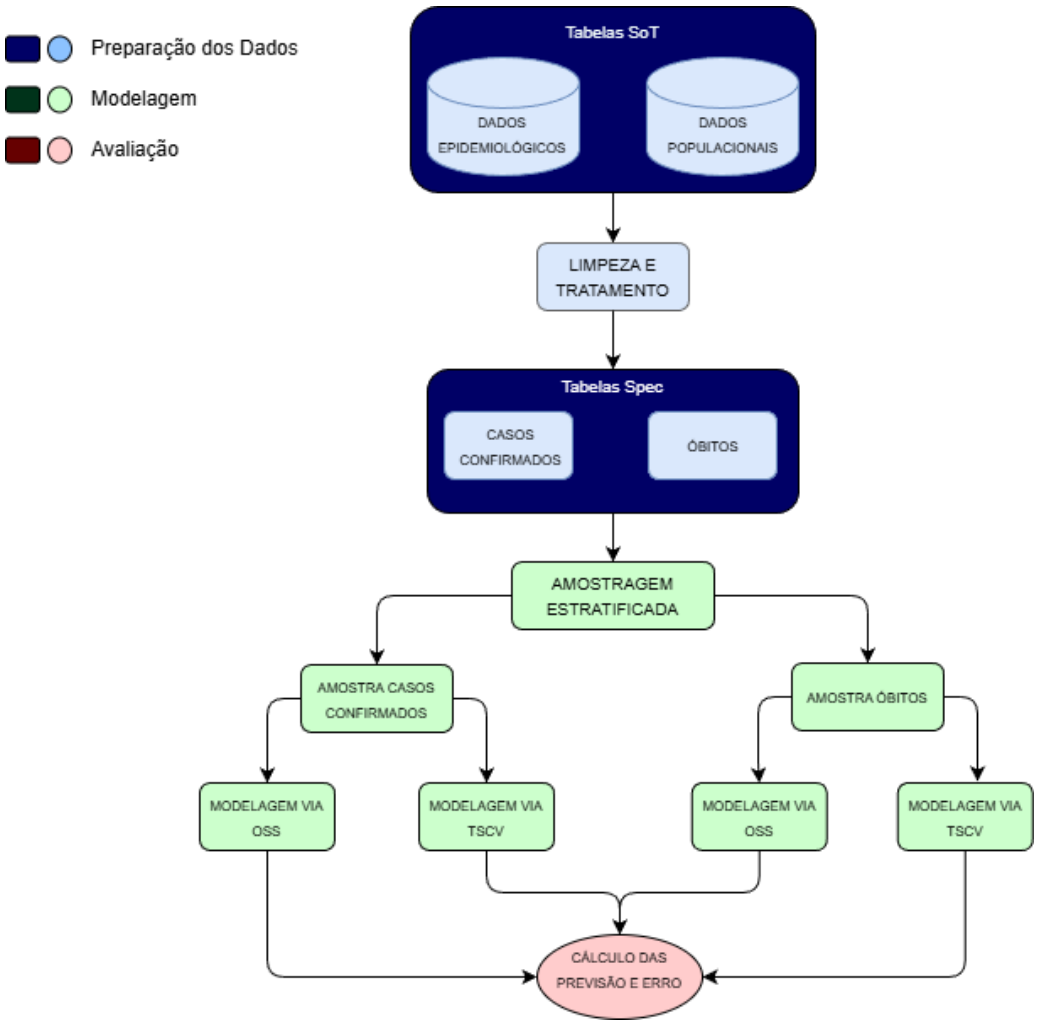
As hipóteses embasam a premissa do trabalho, encontrar modelos que sejam tão rapidos de execução quanto um ARMA, mas que tenham a precisão e interpretabilidade dos resultados de um NNETAR

5.3 RESULTADOS

Após treinarmos os modelos utilizando o método de *cross validation* e via out of sample, foi possível analisar os desempenhos e analisar as hipóteses formuladas considerando a variável de casos confirmados

O *pipeline* utilizado para modelagem pode ser observado através do seguinte fluxograma

Pipeline da construção de modelos e obtenção de métricas



Fonte: Elaborado pelo autor (2025)

5.3.1 CASOS CONFIRMADOS

Após treinarmos os modelos utilizando o método de *cross validation* e via out of sample, foi possível analisar os desempenhos e analisar as hipóteses formuladas considerando a variável de casos confirmados

Para melhor organização, as tabelas com os resultados serão mostradas em 2 blocos distintos. Primeiro, para os modelos com previsão 1 passo a frente, onde o método TSCV foi utilizado, e depois para os modelos com previsão 4 passos a frente, onde o método OOS foi utilizado

É importante destacar que células que não apresentam valores de RMSE, são por conta de modelos que não foram possíveis de serem estimados. Esse erro ocorreu apenas para os modelos GLARMA, e se dá por conta não convergência dos métodos numéricos utilizados para a estimação dos parâmetros do modelo

- AVALIAÇÃO EM UM HORIZONTE CURTO DE PREVISÃO VIA

TIME SERIES CROSS-VALIDATION

Tabela 3 – RMSE via TSCV por município estrato de casos baixos

Município	ARMA	NNETAR	GLA ARMA	GLA Naive	ING ARMA	ING LASSO	ING Naive
Berizal	1.232	1.275	-	-	**0.752**	0.757	0.767
Caiana	0.568	0.444	-	-	**0.439**	0.492	0.491
Cascalho Rico	11.015	10.804	11.113	11.000	10.823	**10.800**	10.823
Cristiano Otoni	0.903	1.019	**0.707**	1.000	0.903	0.917	0.903
Divinésia	0.780	**0.428**	-	-	0.687	0.676	0.676
Dom Viçoso	**0.078**	0.229	-	-	0.198	0.202	0.202
Franciscópolis	0.541	10.499	2.500	**1.225**	1.772	1.408	2.222
Glaucilândia	1.252	3.396	-	-	**0.905**	1.138	1.141
Guidoval	0.344	0.335	0.866	0.758	**0.314**	0.327	0.322
Ipuiúna	0.891	**0.514**	1.414	1.257	0.801	0.649	0.618
Piracema	0.991	0.943	1.225	**0.866**	0.991	1.006	1.006
Ponto Chique	0.437	0.500	0.500	0.707	**0.422**	0.479	0.495
Vazante	1.193	**0.900**	-	-	1.270	1.270	1.221

Dos 13 municípios classificados com nível baixo de casos confirmados, o modelo ARMA apresentou o melhor desempenho em apenas 1 localidade. O modelo NNETAR destacou-se em 3 municípios, enquanto o GLARMA obteve o menor RMSE em 2 casos. Já o modelo INGARCH foi o mais eficiente em 6 municípios, destacando-se como aquele que apresentou o melhor desempenho no maior número de ocorrências.

Cabe ressaltar que a primeira hipótese formulada foi confirmada para este estudo de caso: os modelos de contagem superaram os modelos tradicionais, apresentando melhor desempenho em 8 municípios

Além disso, quando comparamos os métodos para seleção automática de ordem de parâmetros, vemos que o método ARMA-based foi o melhor 6 ocasiões, enquanto o Naive-Search foi o melhor em 2 ocasiões (todas em modelos GLARMA) e o método via LASSO foi o melhor em apenas 1 ocasião

Tabela 4 – RMSE via TSCV por município estrato de casos médio

Município	ARMA	NNETAR	GLA ARMA	GLA Naive	ING ARMA	ING LASSO	ING Naive
Alpinópolis	0.633	0.654	1.155	**0.577**	2.467	0.801	0.729
Araçuaí	7.105	7.884	8.930	5.766	**5.257**	6.776	6.776
Caxambu	**6.144**	6.156	-	-	6.439	6.487	6.487
Conselheiro Pena	2.292	1.456	-	-	1.877	**1.205**	1.905
Coronel Fabriciano	11.558	11.811	18.815	**11.522**	11.972	11.972	11.994
Cruzília	4.401	**1.681**	-	-	3.773	2.429	2.334
João Pinheiro	2.037	**1.927**	2.872	2.550	2.576	2.807	2.166
Leopoldina	6.185	6.739	7.246	7.714	**6.005**	6.428	7.019
Miraí	**0.689**	1.542	-	-	1.243	1.232	1.232
Ouro Branco	4.793	6.045	-	-	4.202	5.075	**4.200**
Santana do Jacaré	3.774	4.063	-	-	**2.972**	3.555	3.433
São José da Lapa	2.758	6.731	4.387	**2.646**	2.758	2.703	3.161
Veríssimo	0.207	0.081	1.936	2.062	0.316	2.169	**0.060**

Analisando os municípios com nível mediano de casos confirmados, vemos que os modelos ARMA e NNETAR apresentaram o menor RMSE em 2 municípios cada. Já o modelo GLARMA teve o melhor desempenho em 3 ocasiões, e o modelo INGARCH foi o melhor em 5 cidades

Novamente, vemos que os modelos de contagem apresentaram um desempenho superior aos modelos tradicionais, se mostrando como a melhor alternativa em 9 dos 13 municípios analisados. Além disso, vemos que o modelo INGARCH foi aquele a apresentar o melhor desempenho, se mostrando o melhor modelo tanto para o estrato de casos baixo como casos medianos

Analisando os métodos para seleção automático de ordem de parâmetros, vemos que o método Naive-Search foi aquele a apresentar o menor RMSE em mais ocasiões, tendo o melhor desempenho em 5 municípios. Já o método via LASSO foi o melhor em apenas 1 ocasião, sendo aquele com menor destaque dentre os 3 métodos

Tabela 5 – RMSE via TSCV por município estrato de casos alto

Município	ARMA	NNETAR	GLA ARMA	GLA Naive	ING ARMA	ING LASSO	ING Naive
Andradas	15.033	17.246	-	-	15.729	15.729	**12.941**
Arcos	213.855	244.020	214.76	221.588	**199.418**	236.637	213.383
Capelinha	3.554	7.690	-	-	**3.377**	4.134	3.916
Congonhas	23.990	23.338	28.00	22.000	**17.337**	20.723	19.404
Curvelo	21.677	31.214	-	-	20.471	**20.168**	20.382
Governador Valadares	17.997	**15.410**	-	-	16.161	20.042	20.042
Itapecerica	20.568	27.670	**9.22**	12.649	20.908	21.183	19.838
Itaú de Minas	10.720	9.920	**6.18**	7.000	8.552	9.880	9.916
Ituiutaba	4.683	5.481	-	-	28.680	5.935	**3.016**
João Monlevade	14.406	12.236	12.50	13.601	12.296	13.422	**11.877**
Lagoa da Prata	4.595	**2.608**	-	-	4.956	4.101	3.613
Mariana	**41.345**	51.652	-	-	45.337	45.337	44.332
São João del Rei	15.688	11.568	3.00	**1.450**	15.816	15.536	14.702

Passando para o cenário dos municípios com a maior volumetria de casos confirmados, vemos que os modelos de contagem apresentaram o melhor desempenho em 10 dos 13 municípios.

Novamente, vemos que o modelos INGARCH foi aquele a apresentar o melhor desempenho em mais ocasiões, tendo o menor RMSE em 6 municípios

Vale destacar que para os casos no qual o modelo GLARMA se mostrou superior, a diferença para os outros modelos foi significativa, chegando a um ganho de desempenho médio próximo a 60% quando comparado ao segundo melhor modelo de outra família

- AVALIAÇÃO EM UM HORIZONTE LONGO DE PREVISÃO VIA OUT OF SAMPLE

Tabela 6 – RMSE via OSS por município estrato de casos baixo

Município	ARMA	NNETAR	GLA ARMA	GLA Naive	ING ARMA	ING LASSO	ING Naive
Berizal	0.820	2.043	-	-	**0.616**	0.696	0.656
Caiana	0.483	0.655	-	-	**0.427**	0.443	0.452
Cascalho Rico	11.000	10.686	11.000	**10.050**	10.804	10.824	10.813
Cristiano Otoni	0.896	0.900	**0.500**	1.000	0.888	0.883	0.886
Divinésia	0.738	**0.480**	-	-	0.671	0.671	0.642
Dom Viçoso	**0.066**	0.600	-	-	0.422	0.485	0.477
Franciscópolis	**0.752**	7.811	2.550	1.225	1.726	1.771	3.850
Glaucilândia	1.118	1.192	-	-	**0.908**	1.073	1.085
Guidoval	0.354	0.393	**0.027**	0.500	0.354	0.362	0.363
Ipuiúna	0.784	0.521	**0.119**	0.866	0.870	0.718	0.462
Piracema	0.990	0.959	2.062	**0.707**	1.005	0.965	0.971
Ponto Chique	0.433	0.427	0.707	0.707	**0.414**	0.445	0.442
Vazante	1.000	1.441	-	-	1.020	1.030	**0.914**

Passando a avaliar um horizonte de previsão para o intervalo de 4 semanas epidemiológicas, vemos que os modelos de contagem continuam a apresentar um desempenho superior quando comparados aos modelos tradicionais para o estrato "baixo" de casos confirmados. Dos 13 municípios analisados, os modelos desenvolvidos nesse trabalho apresentaram um desempenho superior em 10 casos, sendo 5 vezes para o modelo INGARCH e 5 vezes para o modelo GLARMA

Vale destacar que novamente os métodos de seleção automática de parâmetros ARMA-based se mostraram superior aos demais, sendo o melhor método em 6 dos 10 municípios nos quais os modelos de contagem foram os melhores

Tabela 7 – RMSE via OSS por município estrato de casos médio

Município	ARMA	NNETAR	GLA ARMA	GLA Naive	ING ARMA	ING LASSO	ING Naive
Alpinópolis	1.070	**0.748**	1.118	3.279	2.525	1.258	1.225
Araçuaí	5.355	8.271	**3.571**	5.050	5.217	5.254	5.281
Caxambu	5.196	5.342	-	-	5.231	5.215	**5.183**
Conselheiro Pena	1.952	**1.412**	-	-	1.429	1.422	1.461
Coronel Fabriciano	9.550	12.510	9.618	10.840	9.449	**9.448**	10.029
Cruzília	4.510	19.764	-	-	3.873	2.995	**2.869**
João Pinheiro	**1.649**	2.139	5.635	2.958	2.734	2.962	2.053
Leopoldina	6.128	6.262	7.566	7.036	**6.121**	6.129	6.178
Miraí	**0.470**	4.297	-	-	1.808	1.855	1.963
Ouro Branco	4.361	15.436	-	-	**4.354**	4.370	4.552
Santana do Jacaré	2.710	3.597	-	-	2.913	2.700	**2.557**
São José da Lapa	2.741	15.728	3.428	2.739	2.734	**2.521**	3.238
Veríssimo	**0.000**	2.473	3.606	2.550	0.447	2.548	0.003

Assim como no cenário de um horizonte de previsão curto, os modelos de contagem também apresentaram um desempenho superior no cenário de horizonte de previsão longo considerando um estrato de casos confirmados médios. Dos 13 municípios analisados, os modelos de contagem apresentaram desempenho superior aos modelos tradicionais em 8 ocasiões

Analisando os métodos de seleção automática de ordem de parâmetros, vemos uma boa distribuição de desempenho entre os 3 métodos desenvolvidos nesse trabalho. Diferentemente do cenário para o estrato "baixo" de casos confirmados, onde vimos uma desempenho superior do método ARMA-based, para o cenário atual vemos que RMSE entre familias de modelos iguais tivaram uma diferença mínima. Como para o município de Coronel Fabriciano, onde a delta do RMSE foi de 0,001

Tabela 8 – RMSE via OSS por município estrato de casos alto

Município	ARMA	NNETAR	GLA ARMA	GLA Naive	ING ARMA	ING LASSO	ING Naive
Andradas	12.846	16.304	-	-	13.590	13.644	**12.427**
Arcos	169.748	300.994	184.022	183.899	169.102	**169.066**	173.505
Capelinha	**3.159**	25.824	-	-	4.597	3.982	4.968
Congonhas	15.867	**14.772**	-	-	15.706	15.713	16.025
Curvelo	**14.168**	16.168	-	-	15.197	15.344	15.288
Governador Valadares	**14.215**	42.353	-	-	16.179	17.429	17.245
Itapecerica	**14.702**	15.896	-	-	15.093	15.025	15.701
Itaú de Minas	9.432	9.855	-	-	**8.145**	8.296	8.199
Ituiutaba	**3.742**	8.342	-	-	29.612	18.513	5.870
João Monlevade	12.132	12.236	**11.000**	12.379	12.143	12.173	11.179
Lagoa da Prata	5.469	**2.792**	-	-	5.105	4.575	4.097
Mariana	**33.249**	46.887	-	-	33.577	33.541	34.661
São João del Rei	11.470	**11.370**	-	-	11.472	11.432	11.525

No último cenário avaliado para a variável de casos confirmados, onde foi estudado um horizonte de previsãpo longo com alta volumetria, vimos que pela primeira vez um destaque dos modelos tradicionais em relação aos modelos de contagem.

Dos 13 municipios analisados, os modelos tradicionais apresentaram desempenho superior em 9 ocasioes, com destaque para o modelo ARMA que teve o melhor desempenho em 6 municípios.

Assim como já imaginado e definido via hipótese, o ganho de desempenho que é visto dos modelos de contagem em relação aos modelos tradicionais em volumetria menores, diminui a medida que essa volumetria aumenta. Vimos a diminuição dos modelos de contagem se mostrando como os melhores a medida que a volumetria aumentava, saindo de 10 ocasiões para o estrato "baixo", indo para 8 ocasiões para o estrato "médio" e chegando a apenas 3 ocasiões no estrato "alto"

Um ponte interessante observado aqui foi que essa perda de desempenho não foi observada para um horizonte de previsão curto, como mostrado na seção anterior, onde os modelos de contagem apresentaram o melhor desempenho para os 3 estratos analisados.

Assim, notamos que os modelos de contagem estrapolaram a zona ótima de modelagem descrita na hipótese, onde além de apresentarem um ótimo desempenho para estratos de casos "baixos", eles tambem foram superiores para estratos medios (horizonte de previsão curto e longo) e altos (horizonte de previsão curto)

5.3.2 ÓBITOS

Para avaliação dos modelos de previsão para óbitos, o fluxo de trabalho desenvolvido foi o mesmo daquele utilizado na modelagem de casos confirmados. Ou seja, para avaliação do modelo em um horizonte de previsão curto, o método de TSCV foi utilizado, onde modelos para os estratos de volumetria de óbitos "baixíssimo", "baixo", "médio", "alto". Ja para a avaliação dos modelos em um horizonte de previsão longo, o método OOS foi utilizado

- AVALIAÇÃO EM UM HORIZONTE CURTO DE PREVISÃO VIA TIME SERIES CROSS-VALIDATION

Tabela 9 – RMSE via TSCV por município estrato de óbitos baixíssimo

Município	ARMA	NNETAR	GLA ARMA	GLA Naive	ING ARMA	ING LASSO	ING Naive
Abaeté	0.000	0.066	0.000	0.5	0.061	0.061	0.061
Bom Sucesso	0.503	0.480	1.118	0.5	0.482	0.482	0.482
Coração de Jesus	0.000	0.020	0.500	0.0	0.020	0.020	0.020
Corinto	0.000	0.021	0.000	0.0	0.020	0.020	0.020
Ervália	0.000	0.022	0.000	0.0	0.020	0.020	0.020
Itaúna	0.000	0.022	0.000	0.0	0.020	0.020	0.020
Itinga	0.000	0.021	0.000	0.0	0.020	0.020	0.020
Januária	0.000	0.044	0.000	0.0	0.040	0.040	0.040
Jaíba	0.000	0.042	0.000	0.5	0.040	0.040	0.040
Matutina	0.000	0.021	0.000	0.0	0.020	0.020	0.020
Mutum	0.000	0.000	0.000	0.0	0.020	0.020	0.020
Nova União	0.000	0.021	0.000	0.0	0.020	0.020	0.020

É possível observar que, para o estrato baixíssimo, diversos modelos apresentaram RMSE igual a zero, ou seja, a estimativa pontual produzida foi exatamente igual ao valor real a ser estimado.

Embora isso possa parecer um bom sinal à primeira vista, no contexto de modelos estatísticos e de aprendizado de máquina, espera-se que exista algum erro associado às previsões. Quando um modelo não apresenta erro algum, frequentemente estamos diante de um caso de *overfitting* ou sobreajuste , situação em que o modelo se adapta excessivamente aos dados de treinamento, perdendo a capacidade de generalização para novos dados.

No conjunto de dados analisado, esse fenômeno ocorre principalmente devido à predominância de valores zero ao longo das semanas epidemiológicas. O estrato baixíssimo é composto por municípios que registraram entre 1 e 4 óbitos confirmados durante o ano de 2024, distribuídos ao longo das 52 semanas epidemiológicas. Isso significa que, na maioria dessas semanas, o número de óbitos registrados foi zero, o que influencia fortemente o treinamento dos modelos.

Diante dessa alta frequência de zeros, os modelos acabam também gerando estimativas iguais a zero, o que leva ao falso indício de desempenho perfeito. No entanto, esse

tipo de ajuste pode ser problemático, pois aumenta o risco de o modelo não ser capaz de identificar mudanças no padrão dos óbitos ao longo do tempo. Ou seja, ao se ajustar perfeitamente ao conjunto de treinamento, o modelo tende a apresentar estimativas enviesadas, frequentemente prevendo valores iguais a zero e falhando em detectar possíveis aumentos no número de óbitos.

Para series temporais com esse tipo de fenômeno, os modelos de contagem apresentados podem ser utilizados, mas é importante destacar novamente que esses possuem uma baixa capacidade de se adaptarem em mudanças rápidas quando esse fenômeno ocorre

Para captar essas mudanças, outros tipos de abordagens podem ser recomendadas, onda essas estão além do escopo desse trabalho, mas valem ser mencionadas caso o leitor se interesse em se aprofundar no tema.

- Testes de hipótese para detecção de pontos de mudança (change point detection): Essa abordagem busca identificar momentos em que há uma mudança estatisticamente significativa na distribuição da série temporal como uma mudança na média, na variância ou na tendência. Métodos como o CUSUM, Pelt, ou Bayesian Change Point Detection podem ser utilizados para detectar possíveis transições, mesmo em séries com muitos valores constantes.

- Modelos com variáveis exógenas (ARMAX, GLARMAX, INGARCHX): Incorporar variáveis externas como o número de casos confirmados em semanas anteriores pode ajudar os modelos a capturar padrões latentes e antecipar mudanças, mesmo quando a variável resposta apresenta muitos zeros. O sufixo X representa uma referencia ao termo *eXogenous variables*, ou variáveis exógenas em português.

- Modelos para dados esparsos ou inflacionados com zeros: Em situações com excesso de zeros, modelos específicos como o Zero-Inflated Poisson (ZIP) ou o Zero-Inflated Negative Binomial (ZINB) podem ser mais adequados. Esses modelos assumem que os zeros podem vir de dois processos distintos: um que sempre gera zero (como a ausência real de eventos) e outro que segue uma distribuição de contagem.

- Modelos bayesianos e hierárquicos: Abordagens bayesianas permitem incorporar informações a priori e podem ser úteis para suavizar previsões em séries com baixa incidência de eventos. Modelos hierárquicos também podem aproveitar informações de diferentes municípios ou regiões para melhorar as estimativas em locais com poucos dados.

Tabela 10 – RMSE via TSCV por município estrato de óbitos baixo

Município	ARMA	NNETAR	GLA ARMA	GLA Naive	ING ARMA	ING LASSO	ING Naive
Barbacena	0.142	0.093	**0.0**	**0.0**	0.142	0.087	0.087
Boa Esperança	0.142	0.112	0.5	**0.000**	0.142	0.114	0.142
Formiga	0.162	0.182	**0.0**	0.500	0.162	0.162	0.162
Frutal	0.182	0.212	**0.0**	0.000	0.182	0.182	0.182
Ituiutaba	0.119	**0.028**	-	-	0.100	0.115	0.121
João Monlevade	0.101	0.094	**0.0**	0.500	0.101	0.170	0.101
Manhuaçu	0.121	0.123	**0.0**	**0.0**	0.121	0.141	0.121
Mariana	0.101	0.254	**0.0**	0.577	0.101	0.235	0.101
Poços de Caldas	0.548	**0.499**	-	-	0.514	0.515	0.514
Ribeirão das Neves	**0.006**	0.165	-	-	0.142	0.106	0.123
São Sebastião do Paraíso	0.336	0.187	-	-	0.360	0.146	**0.142**
Unaí	0.142	0.159	0.500	0.500	0.142	**0.108**	0.142

Observamos novamente a ocorrência do fenômeno de overfitting em alguns modelos, com destaque para os ajustes realizados pelo GLARMA. Esse resultado evidencia que, mesmo os modelos de contagem, teoricamente mais adequados para lidar com dados discretos, podem apresentar comportamentos inesperados e desempenho insatisfatório quando aplicados a dados reais, sujeitos a variabilidade e imperfeições de medição. Ainda assim, é possível notar que o modelo INGARCH se mostrou mais robusto e consistente nesse tipo de cenário, uma vez que nenhum dos ajustes dessa classe apresentou resíduos nulos, indicando uma modelagem mais estável e adequada às características do conjunto de dados analisado.

Tabela 11 – RMSE via TSCV por município estrato de óbitos medio

Município	ARMA	NNETAR	GLA ARMA	GLA Naive	ING ARMA	ING LASSO	ING Naive
Araxá	0.243	0.254	**0.000**	0.500	0.243	0.243	0.243
Conselheiro Lafaiete	0.583	0.619	0.707	0.707	0.572	**0.437**	0.580
Contagem	0.960	1.059	1.118	1.118	**0.958**	1.013	1.087
Divinópolis	0.632	0.758	1.414	1.291	0.618	0.619	0.618
Governador Valadares	1.035	1.079	1.225	1.118	**0.989**	**0.989**	**0.989**
Teófilo Otoni	1.247	6.013	-	-	**0.276**	1.070	1.070
Três Corações	0.202	0.228	**0.000**	0.707	0.202	0.189	0.202

Para o estrato de óbitos classificado como médio, que compreende os municípios com uma faixa de 10 a 20 óbitos ao longo de 2024, observou-se que o modelo GLARMA apresentou resíduo igual a zero em dois municípios, indicando possível instabilidade no ajuste.

Nos cinco municípios restantes, o modelo INGARCH demonstrou desempenho superior em relação aos demais, confirmando a tendência já observada nos outros estratos. Esse resultado reforça que o INGARCH é particularmente eficaz em contextos caracterizados por séries temporais longas como as 52 semanas epidemiológicas analisadas , mas com baixa volumetria de eventos por período, isto é, médias inferiores a 2 óbitos por mês. Tal configuração evidencia a capacidade do modelo em lidar adequadamente com contagens reduzidas e dispersas, preservando estabilidade e boa qualidade preditiva

Tabela 12 – RMSE via TSCV por município estrato de óbitos alto

Município	ARMA	NNETAR	GLA ARMA	GLA Naive	ING ARMA	ING LASSO	ING Naive
Belo Horizonte	1.780	**1.710**	2.550	2.121	2.204	2.137	1.807
Juiz de Fora	**0.275**	0.376	0.707	0.866	0.546	0.474	0.546
Uberaba	0.633	**0.467**	0.707	0.707	0.643	0.621	0.643
Uberlândia	1.668	3.097	1.658	2.121	**1.322**	1.733	1.733

Para o estrato de óbitos alto, que abrange municípios com maior volumetria de registros, como Belo Horizonte, Juiz de Fora, Uberaba e Uberlândia, observamos um comportamento distinto em relação aos estratos anteriores. Nesses casos, as séries temporais apresentam menor predominância de zeros e maior variabilidade semanal, o que torna o ajuste dos modelos mais desafiador, mas também mais informativo para a comparação de desempenho.

De forma geral, o ARMA e o NNETAR mantiveram desempenho competitivo, com destaque para o modelo baseado em rede neural, que apresentou o menor RMSE em dois dos quatro municípios analisados (Belo Horizonte e Uberaba). Esse resultado sugere que, em contextos com maior densidade de observações e padrões temporais mais complexos, abordagens não lineares são capazes de capturar melhor as dinâmicas locais dos dados, beneficiando-se da maior quantidade de variação disponível para o treinamento.

Por outro lado, os modelos de contagem (GLARMA e INGARCH), que haviam se destacado nos estratos de menor volumetria, apresentaram desempenho inferior nesse grupo. O modelo GLARMA, em particular, mostrou RMSEs elevados, ultrapassando 2.0 em Belo Horizonte e 0.7 em Juiz de Fora, indicando uma possível limitação na sua capacidade de acomodar flutuações mais amplas e padrões mais irregulares quando o número de eventos é alto.

Entre os modelos da classe INGARCH, o método baseado em um modelo ARMA apresentou o melhor desempenho em Uberlândia (RMSE = 1.322), superando os demais modelos, inclusive o NNETAR. Esse comportamento reforça que, embora os modelos de contagem sejam mais adequados para dados discretos, seu desempenho é sensível ao regime de variabilidade: eles tendem a ser mais estáveis em séries com baixa média e dispersão controlada, mas podem perder acurácia conforme aumenta a amplitude das contagens semanais.

- AVALIAÇÃO EM UM HORIZONTE LONGO DE PREVISÃO VIA OUT OF SAMPLE

Tabela 13 – RMSE via OSS por município estrato de óbitos baixíssimo

Município	ARMA	NNETAR	GLA ARMA	GLA Naive	ING ARMA	ING LASSO	ING Naive
Abaeté	**0.0**	0.060	**0.0**	**0.0**	0.066	0.063	0.067
Bom Sucesso	0.5	0.480	0.5	0.5	0.479	0.480	**0.474**
Coração de Jesus	**0.0**	0.021	**0.0**	**0.0**	0.020	0.020	0.021
Corinto	**0.0**	0.019	**0.0**	**0.0**	0.016	0.022	0.022
Ervália	**0.0**	0.021	**0.0**	0.5	0.023	0.021	0.019
Itaúna	**0.0**	0.023	**0.0**	**0.0**	0.021	0.022	0.017
Itinga	**0.0**	0.019	**0.0**	**0.0**	0.023	0.019	0.020
Januária	**0.0**	0.041	0.5	**0.0**	0.047	0.042	0.046
Jaíba	**0.0**	0.044	**0.0**	**0.0**	0.041	0.046	0.045
Matutina	**0.0**	0.022	**0.0**	**0.0**	0.020	0.019	0.021
Mutum	**0.0**	**0.0**	**0.0**	**0.0**	0.019	0.022	0.018
Nova União	**0.0**	0.020	**0.0**	**0.0**	0.025	0.023	0.020

Tabela 14 – RMSE via OSS por município estrato de óbitos baixo

Município	ARMA	NNETAR	GLA ARMA	GLA Naive	ING ARMA	ING LASSO	ING Naive
Barbacena	0.146	0.095	**0.0**	0.5	0.144	0.101	0.107
Boa Esperança	0.146	0.173	0.5	**0.0**	0.143	0.136	0.141
Formiga	0.167	0.174	**0.0**	**0.0**	0.157	0.159	0.162
Frutal	0.188	0.183	**0.0**	0.5	0.193	0.175	0.188
Ituiutaba	0.152	**0.063**	-	-	0.127	0.113	0.119
João Monlevade	0.104	0.109	**0.0**	0.5	0.099	0.209	0.100
Manhuaçu	0.125	0.109	**0.0**	**0.0**	0.111	0.128	0.135
Mariana	0.104	0.308	0.5	**0.0**	0.094	0.274	0.104
Poços de Caldas	0.500	0.**455**	-	-	0.468	0.463	0.459
Ribeirão das Neves	**0.006**	0.145	-	-	0.144	0.128	0.154
São Sebastião do Paraíso	0.372	0.396	-	-	0.465	**0.146**	0.149
Unaí	0.146	0.140	**0.0**	1.0	0.134	0.128	0.154

Ao avançar para o estrato de óbitos baixo, a leve elevação na frequência de registros permite uma análise mais sensível das diferenças entre modelos. Novamente, vemos que o modelo GLARMA apresentou o fenomeno de overffitng, tendo RMSE igual a 0 em 6 dos 12 municípios analisados. O modelo INGARCH não apresentou esse problema, e se mostrou como o modelo com melhor desempenho em 6 casos (excluindo o GLARMA com overffitng da análise)

Tabela 15 – RMSE via OSS por município estrato de óbitos médio

Município	ARMA	NNETAR	GLA ARMA	GLA Naive	ING ARMA	ING LASSO	ING Naive
Araxá	0.250	0.271	0.500	**0.000**	0.265	0.260	0.250
Conselheiro Lafaiete	0.579	0.505	0.707	1.225	0.581	**0.391**	0.576
Contagem	0.958	0.957	1.118	0.707	0.966	**0.881**	0.953
Divinópolis	0.640	0.734	1.118	1.118	0.630	0.630	**0.599**
Governador Valadares	1.118	**0.968**	1.118	1.323	0.987	1.000	0.976
Teófilo Otoni	1.708	8.262	-	-	**0.333**	1.454	1.487
Três Corações	0.208	0.196	**0.000**	**0.000**	0.204	0.201	0.209

No estrato de óbitos médio, onde há maior regularidade e densidade de eventos, os resultados indicam uma transição no desempenho relativo entre famílias de modelos.

Embora ainda haja forte presença de modelos de contagem entre os melhores, os ganhos em relação aos métodos tradicionais se tornam mais sutis, com RMSEs próximos entre as abordagens. Casos como Governador Valadares e Conselheiro Lafaiete ilustram essa convergência: a diferença de desempenho entre o melhor modelo INGARCH e o modelo ARMA foi inferior a 0.1 ponto no RMSE. Ainda assim, os modelos INGARCH (nas variantes LASSO e Naive) se destacaram em três dos sete municípios, mostrando boa adaptabilidade às flutuações moderadas das séries. A constância dos resultados para o GLARMA e a estabilidade dos erros reforçam a coerência metodológica observada também nos horizontes curtos.

Tabela 16 – RMSE via OSS por município estrato de óbitos alto

Município	ARMA	NNETAR	GLA ARMA	GLA Naive	ING ARMA	ING LASSO	ING Naive
Belo Horizonte	**1.658**	1.808	3.391	2.915	2.267	2.128	1.661
Juiz de Fora	**0.317**	0.523	0.707	0.707	0.574	0.549	0.573
Uberaba	0.567	0.892	1.118	**0.500**	0.576	0.585	0.587
Uberlândia	1.374	1.548	1.936	1.871	**1.300**	1.405	1.400

Por fim, no estrato de óbitos alto, caracterizado por séries mais longas, regulares e com padrões temporais mais estruturados, observa-se um comportamento distinto do verificado nos estratos com menor volumetria. Nesse grupo, os modelos tradicionais (ARMA e NNETAR) voltam a apresentar desempenho competitivo, com destaque para o ARMA, que apresentou o menor RMSE em Belo Horizonte e Juiz de Fora. Entretanto, diferentemente do que ocorreu no cenário de uma previsão 1 passo a frente, os modelos de contagem não perdem completamente relevância: em Uberlândia, o modelo INGARCH com método ARMA-Based obteve o menor erro preditivo, enquanto em Uberaba o modelo GLARMA com método Naive-Search se destaca. Assim, há um equilíbrio mais pronunciado entre os dois blocos de modelagem, com dois municípios favorecendo modelos tradicionais e dois favorecendo modelos de contagem.

Essa mudança de tendência reforça a hipótese formulada inicialmente no trabalho: o ganho dos modelos de contagem é maior em contextos de baixa volumetria e tende a se reduzir progressivamente à medida que a frequência de observações aumenta, perdendo relevância quando o comportamento da série se aproxima de uma dinâmica quase contínua.

5.3.3 AVALIAÇÃO TEMPO DE EXECUÇÃO DE CADA MODELO

As tabelas apresentadas a seguir descrevem os tempos de estimação, previsão e tempo total de execução dos modelos considerados, organizados por estrato (baixo, médio e alto).

Para o cálculo do tempo de execução dos modelos, cada método foi executado em triplicata, onde o resultado final foi a média das 3 execuções.

Para estimação e previsão dos modelos, o conjunto de dados de casos confirmados da COVID-19 foi utilizado novamente. Para o calculo de tempo das previsão, foi estabelecido um intervalo de previsão 1passo a frente

Tabela 17 – Tempo de estimação de cada modelo (segundos)

Estrato	ARMA	NNETAR	GLA ARMA	GLA Naive	ING ARMA	ING LASSO	ING Naive
Baixo	0.170	0.074	0.377	0.384	1.476	4.675	23.198
Médio	0.209	0.077	0.386	0.396	1.697	5.886	24.858
Alto	0.189	0.090	0.341	0.353	1.532	6.262	34.249

Considerando apenas o tempo de estimação dos parâmetros de cada modelo, observa-se que o NNETAR se destaca por ser o único a apresentar tempo inferior a 1 centésimo de segundo.

Os modelos ARMA e GLARMA apresentaram tempos semelhantes, variando entre 0,1 e 0,4 segundos para a estimação.

Os modelos INGARCH foram os que demandaram maior tempo computacional. Entre eles, os métodos ARMA-Based levaram menos de 2 segundos, enquanto o método baseado em LASSO apresentou tempo médio de 5 segundos.

Destaca-se o INGARCH via Naive-Search, que registrou tempo médio de 27 segundos, sendo o modelo/método com maior tempo de estimação. Esse resultado já era esperado, pois esse método estima o maior número de modelos preliminares entre os procedimentos de seleção automática das ordens dos parâmetros. Ainda assim, nota-se que o tempo de execução é significativamente superior ao observado no modelo GLARMA. Essa diferença decorre da instabilidade dos modelos GLARMA, que, como já discutido na seção de previsão de casos confirmados e óbitos e nas tabelas de comparação de RMSE, apresentam dificuldades de convergência para determinadas ordens de parâmetros. Quando o método numérico não converge, o processo de estimação é interrompido, reduzindo o tempo final de execução do algoritmo.

Ao analisar a relação entre o tempo de estimação e a volumetria modelada (estratos baixo, médio e alto), observa-se que, para modelos e métodos de rápida estimação (tempo inferior a 5 segundos), não é possível identificar uma relação clara. Já entre aqueles cujo tempo ultrapassa 5 segundos, nota-se que as maiores volumetrias estão associadas aos maiores tempos de estimação.

Tabela 18 – Tempo de previsão de cada modelo (segundos)

Estrato	ARMA	NNETAR	GLA ARMA	GLA Naive	ING ARMA	ING LASSO	ING Naive
Baixo	0.037	104.460	0.031	0.030	0.031	0.032	0.031
Médio	0.034	99.904	0.028	0.028	0.031	0.032	0.029
Alto	0.033	99.570	0.027	0.027	0.034	0.034	0.032

Analisando o tempo médio de previsão, em segundos, obtido para cada modelo nos diferentes estratos de volumetria. Observa-se que o NNETAR apresentou um tempo de previsão substancialmente superior aos demais, variando entre aproximadamente 99 e 104 segundos, enquanto os demais modelos demandaram menos de 0,04 segundos em todos os estratos. Em termos práticos, o tempo de previsão do NNETAR é cerca de 2.600 vezes superior ao observado nos demais modelos, evidenciando o elevado custo computacional associado à sua etapa preditiva.

Esse comportamento pode ser explicado pela estrutura computacional adotada na etapa de previsão do NNETAR, conforme implementado no pacote fable. Diferentemente dos modelos lineares (ARMA, GLARMA e INGARCH), que possuem expressões fechadas ou recorrências analíticas para gerar previsões multi-passo de forma direta, o NNETAR realiza previsões de forma recursiva: cada passo previsto é utilizado como entrada para o passo seguinte.

Além disso, o método *forecast.NNETAR()*, implementado no pacote fable, executa, por padrão, múltiplas simulações para compor os intervalos de previsão. Assim, o algoritmo repete o processo preditivo diversas vezes, gerando trajetórias futuras independentes e, em seguida, agregando-as para estimar a distribuição dos valores previstos, realizando portanto um bootstrap não paramétrico. Esse processo, ainda que essencial para a caracterização da incerteza do modelo, aumenta substancialmente o custo computacional.

Outro fator relevante é que, diferentemente dos modelos lineares, o NNETAR não armazena uma matriz de coeficientes ou estrutura paramétrica simples para cálculo direto da previsão. Cada iteração envolve a execução completa da rede neural o que inclui operações de multiplicação de matrizes e aplicação de funções de ativação para cada simulação e para cada passo à frente no horizonte de previsão.

E portanto, apesar de o NNETAR apresentar tempo de estimação reduzido, como discutido anteriormente, o custo computacional é transferido para a fase de previsão, tornando-o o modelo com o maior tempo de execução entre os avaliados

Tabela 19 – Tempo de total (estimação + previsão) de cada modelo (segundos)

Estrato	ARMA	NNETAR	GLA ARMA	GLA Naive	ING ARMA	ING LASSO	ING Naive
baixo	0.207	104.535	0.408	0.414	1.508	4.707	23.229
medio	0.242	99.980	0.414	0.424	1.728	5.917	22.887
alto	0.222	99.660	0.368	0.380	1.567	6.296	34.281

Ao considerar o tempo total de execução, isto é, a soma dos tempos de estimação e previsão, observa-se que o modelo NNETAR apresenta, de forma consistente em todos os estratos, o maior tempo total entre os métodos avaliados, variando entre aproximadamente 100 e 104 segundos. Esse resultado reforça a conclusão anterior de que, embora o NNETAR possua uma etapa de estimação rápida, seu elevado custo computacional na

fase de previsão domina o tempo total do processo.

Em comparação, o INGARCH via Naive-Search, que já havia se destacado por ser o método de estimação mais demorado (cerca de 23 a 34 segundos), apresenta menos da metade do tempo total de execução do NNETAR. Essa diferença evidencia o peso desproporcional que o processo de previsão exerce sobre o desempenho computacional do NNETAR, uma vez que os demais modelos, mesmo os mais complexos na estimação, mantêm tempos de previsão praticamente desprezíveis.

O modelo ARMA apresentou os menores tempos de execução, variando entre 0.207 e 0.242 segundos, refletindo a simplicidade estrutural do modelo e o fato de que tanto a estimação quanto a previsão são computacionalmente diretas, sem necessidade de simulações ou processos iterativos complexos.

Os métodos GLARMA, mostraram tempos intermediários, próximos de 0,368 a 0,424 segundos. Embora a previsão desses modelos seja praticamente instantânea devido à linearidade da estrutura, a estimação iterativa dos parâmetros GLARMA eleva o tempo total em relação ao ARMA.

Entre os modelos INGARCH, observa-se uma diferenciação clara nos tempos de execução. O método ARMA-Based apresentou tempos entre 1,508 e 1,728 segundos, evidenciando o custo da estimação iterativa sem impactos relevantes na previsão. O via LASSO demandou um tempo médio maior, entre 4,707 e 6,296 segundos, devido à aplicação da regularização LASSO na seleção dos parâmetros do modelo. O INGARCH via Naive-Search destacou-se como o método mais custoso entre os INGARCH, com tempos variando de 22,887 a 34,281 segundos. Nesse caso, a elevada duração deve-se à necessidade de avaliar múltiplas combinações de ordens de parâmetros, processo intensivo de estimação, embora a previsão permaneça rápida.

5.4 ANÁLISE FINAL DAS HIPÓTESES

Com base nos resultados obtidos ao longo das análises, é possível avaliar o grau de evidência empírica em relação às duas hipóteses formuladas neste estudo.

A primeira hipótese considerava que os modelos de contagem (GLARMA e INGARCH) apresentariam desempenho superior aos modelos tradicionais (ARMA e NNETAR), especialmente em séries com baixa ou média volumetria de eventos. Os resultados obtidos confirmam essa hipótese de forma consistente. Nos três estratos de casos confirmados analisados para o horizonte de previsão curto, os modelos de contagem apresentaram o menor erro em 8, 9 e 10 dos 13 municípios analisados, respectivamente. Essa superioridade também se manteve, em grande parte, no horizonte de previsão longo, sobretudo para os estratos baixo e médio, com 10 e 8 vitórias sobre os modelos tradicionais, respectivamente. Tais resultados indicam que os modelos de contagem capturam de forma mais

eficiente a estrutura discreta e heterocedástica das séries epidemiológicas, o que os torna mais adequados para modelar dados de baixa frequência e alta dispersão.

Entretanto, essa vantagem diminui à medida que a volumetria aumenta. Nos municípios classificados com alta volumetria, especialmente no horizonte de previsão longo, observou-se um desempenho mais equilibrado entre as famílias de modelos, com vantagem inclusive para os modelos tradicionais em 9 dos 13 casos analisados. Esse resultado reforça o argumento de que, em cenários com maior regularidade temporal, menor dispersão relativa e maior volumetria absoluta de dados, a modelagem baseada em distribuições contínuas como a gaussiana assumida pelos modelos ARMA tende a ser suficientemente eficiente, reduzindo a necessidade de abordagens de contagem.

A segunda hipótese estabelecia que os modelos de contagem exigem menor tempo de treinamento do que o modelo NNETAR, o que os tornaria mais escaláveis para aplicações envolvendo grande número de séries, como no caso de análises municipais. Os resultados obtidos também confirmam essa hipótese. Embora o NNETAR tenha apresentado bom desempenho em termos de acurácia em alguns contextos, o tempo médio de estimação por série foi substancialmente maior do que o observado para os modelos GLARMA e INGARCH. Essa diferença decorre da natureza iterativa do processo de otimização utilizado em redes neurais, que demanda múltiplos ciclos de *backpropagation* e ajustes de pesos, em contraste com a estimação direta e mais eficiente dos parâmetros nos modelos de contagem. Dessa forma, os modelos de contagem mostraram-se não apenas mais adequados sob o ponto de vista estatístico para séries com baixa contagem, mas também computacionalmente mais viáveis em cenários que exigem ajuste em larga escala, reforçando a vantagem prática de sua adoção no contexto epidemiológico considerado.

Em síntese, ambas as hipóteses formuladas foram confirmadas pelos resultados empíricos. Os modelos de contagem demonstraram desempenho superior em estratos de baixa volumetria e apresentaram tempos de treinamento significativamente menores em comparação com o NNETAR, evidenciando que a proposta do pacote `fableCount` é coerente tanto do ponto de vista estatístico quanto computacional.

6 POPULARIDADE E PLANOS FUTUROS

Ao longo de mais de um ano de existência, o pacote já acumula mais de 4.4 mil downloads, segundo dados do site DataScienceMeta.

Rank dos pacotes mais baixados no R - DataScienceMeta

Rank	Package Name	Downloads	Author	Maintainer
19887	StepGWR	4,437		
19888	SensIAT	4,432		
19889	rmass2	4,432		
19890	arcpy	4,432		
19891	SNVLFDR	4,431		
19892	hubEnsembles	4,431		
19893	fableCount	4,427		
19894	rtms	4,424		

Elaborado pelo autor (2025)

O pacote foi lançado inicialmente em abril de 2024, com sua primeira versão (0.0.1) incluindo os modelos INGARCH e GLARMA. Nessa fase inicial, entretanto, o pacote ainda não possuía toda a estrutura de modelagem automatizada apresentada ao longo deste trabalho.

Em maio de 2024, foi lançada a versão 1.0.0, considerada a primeira versão completa do pacote. Essa atualização introduziu os métodos de modelagem Naive-Search e ARMA-Based para os modelos INGARCH e GLARMA.

A versão 1.0.1 consistiu em uma atualização de correção de bugs e adicionou alguns conjuntos de dados de exemplo, permitindo que o usuário se familiarizasse com o pacote e sua sintaxe de código.

Mais recentemente, a versão 1.1.1 incorporou o método de seleção automática de ordens de parâmetros via Post-LASSO, atualmente disponível apenas para o modelo INGARCH.

Como perspectivas futuras para o desenvolvimento do pacote, há diversas linhas de pesquisa e aprimoramento que podem ser exploradas. O trabalho apresentado aplicou os dados em conjunto de dados reais, mas é importante entendermos as melhorias dos modelos de contagem em relação aos modelos usuais em um estudo de simulação e portanto trabalhos futuros com estudos de simulação devem ser realizados, com o objetivo de avaliar o comportamento dos estimadores e dos métodos de seleção em cenários con-

trolados, variando fatores como tamanho amostral, estrutura de dependência temporal e características de dispersão. Diferentemente das aplicações em dados reais, que refletem contextos específicos, a simulação permite investigar de forma sistemática o desempenho dos modelos INGARCH e GLARMA sob diferentes configurações, possibilitando identificar situações em que cada abordagem apresenta vantagens ou limitações. Esse tipo de estudo também poderia servir de base para propor ajustes nos algoritmos de estimação, aperfeiçoar a eficiência computacional e avaliar a robustez dos critérios de seleção automática implementados.

Outro caminho relevante envolve a aplicação dos modelos em séries com sobredispersão, um fenômeno comum em dados de contagem que apresentam variância superior à média. Embora o pacote já disponha de um método para seleção automática de distribuições, este ainda não foi explorado neste trabalho, nem em simulações nem em estudos empíricos, a fim de manter o escopo do projeto focado e conciso. No entanto, investigar o comportamento do pacote sob condições de sobredispersão pode gerar insights importantes sobre a adequação das distribuições implementadas como Poisson, Binomial Negativa e suas variantes e sobre a sensibilidade do processo de seleção automática a diferentes graus de dispersão. Essa análise poderia resultar em melhorias na heurística de escolha da distribuição, tornando o pacote mais robusto e aplicável a uma gama maior de séries temporais de contagem.

Por fim, uma direção natural de evolução para o pacote desenvolvido é o desenvolvimento e integração do método Post-LASSO para o modelo GLARMA. Atualmente, a funcionalidade de seleção automática de ordens de parâmetros via LASSO está implementada apenas para o modelo INGARCH, limitando parcialmente o potencial de automação do pacote. A extensão desse método ao GLARMA exigirá um estudo detalhado sobre a estrutura matemática e os aspectos de estimação iterativa característicos desse modelo, de forma a garantir estabilidade numérica e eficiência computacional. A inclusão dessa funcionalidade traria maior simetria entre os modelos suportados, além de fortalecer o propósito central do pacote de fornecer um ambiente unificado e automatizado para modelagem de séries temporais de contagem dentro do ecossistema `tidyverts`.

7 CONCLUSÃO

No início deste trabalho, identificou-se a necessidade de adoção de novos modelos para a modelagem preditiva epidemiológica, diante das limitações observadas nas abordagens então utilizadas pela plataforma JF Salvando Todos. Os modelos ARIMA e NNETAR, até então empregados, apresentaram uma série de desafios relacionados à estimação, interpretação dos resultados e desempenho computacional, especialmente quando aplicados a séries com baixos valores de contagem e elevada granularidade temporal. Essas limitações motivaram o desenvolvimento de alternativas mais adequadas à natureza discreta dos dados epidemiológicos, capazes de fornecer previsões mais consistentes e interpretáveis.

Nesse ambito, o presente trabalho apresentou o desenvolvimento, implementação e aplicação dos modelos GLARMA e INGARCH no ambiente estatístico R, culminando na criação do pacote `fableCount`, um pacote inédito que buscou integrar a modelagem de séries temporais de contagem ao ecossistema `fable`. Ao longo do estudo, foi possível aliar a fundamentação teórica dos modelos às etapas de programação, apresentando não apenas suas formulações matemáticas e propriedades estatísticas, mas também a forma como foram traduzidas em funções computacionais voltadas à usabilidade, eficiência e automação.

A implementação dos modelos de contagem permitiu que o pacote incorporasse funcionalidades avançadas de estimação e previsão, contemplando a utilização das distribuições Poisson e Binomial Negativa, além de métodos computacionais robustos para cálculo de previsões e utilização em fluxo ideal de modelagem, chamado de *pipeline* de modelagem. A etapa de desenvolvimento computacional buscou seguir a filosofia de automação e reprodutibilidade proposta por (Wickham e Bryan 2023), resultando em uma ferramenta que combina praticidade para o usuário com rigor estatístico.

Outro ponto de destaque foi a implementação dos algoritmos de modelagem automatizada, que tornam o pacote capaz de executar de forma autônoma a seleção de distribuições, a busca de ordens de parâmetros e a escolha do melhor modelo preditivo. Os métodos Naive-Search, ARMA-Based e Post-LASSO deram aos usuários do pacote, flexibilidade e capacidade de adaptação a diferentes tipos de séries temporais, promovendo uma modelagem mais eficiente e acessível, especialmente em contextos que exigem alto volume de processamento, como aplicações em plataformas epidemiológicas.

A aplicação empírica em dados reais reforçou a importância dos modelos de contagem na prática estatística. Os resultados demonstraram ganhos expressivos em termos de desempenho preditivo e computacional, sobretudo em séries com baixos valores de contagem, nas quais os modelos clássicos, como ARIMA, e os modelos baseados em redes neurais apresentaram um tempo de treinamento substancialmente maior que os demais

modelos. Assim, comprovou-se que os modelos de contagem, além de oferecerem melhor aderência à natureza discreta dos dados, também produzem previsões mais coerentes e com menor custo computacional em contextos epidemiológicos.

O pacote `fableCount` já apresenta uma estrutura sólida e funcional, reunindo os principais elementos teóricos e computacionais necessários para a modelagem de séries temporais de contagem. Entretanto, há espaço para avanços significativos. Estudos futuros podem contemplar a realização de experimentos de simulação em ambientes controlados, com o objetivo de avaliar o desempenho dos estimadores sob diferentes configurações de dependência temporal e dispersão; a aplicação do método de seleção automática de distribuições em séries com sobredispersão, permitindo testar sua efetividade em situações práticas; e, finalmente, o desenvolvimento do método Post-LASSO para o modelo GLARMA, de forma a expandir as funcionalidades já existentes para o modelo INGARCH.

Em síntese, este trabalho consolida uma contribuição relevante tanto para a literatura sobre séries temporais de contagem quanto para a comunidade usuária do software R, ao oferecer uma ferramenta prática, automatizada e alinhada às tendências modernas de modelagem estatística. Acredita-se que o `fableCount` possa servir como base para novas pesquisas e aplicações, estimulando o avanço de métodos voltados à análise de dados discretos e fortalecendo a integração entre teoria estatística e desenvolvimento computacional no contexto das séries temporais.

REFERÊNCIAS

Benjamin e Stasinopoulos 1998BENJAMIN, R. R. M.; STASINOPOULOS, M. Modelling exponential family time series data. In: *Statistical Modelling: Proceedings of the 13th International Workshop on Stastical Modelling*. [S.l.: s.n.], 1998.

Bollerslev 1986BOLLERSLEV, T. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, Elsevier, v. 31, n. 3, p. 307–327, 1986.

Cameron e Trivedi 1990CAMERON, A. C.; TRIVEDI, P. K. Regression-based tests for overdispersion in the poisson model. *Journal of econometrics*, Elsevier, v. 46, n. 3, p. 347–364, 1990.

Christou e Fokianos 2014CHRISTOU, V.; FOKIANOS, K. Quasi-likelihood inference for negative binomial time series models. *Journal of Time Series Analysis*, Wiley Online Library, v. 35, n. 1, p. 55–78, 2014.

5CREAL, D.; KOOPMAN, S. J.; LUCAS, A. A general framework for observation driven time-varying parameter models. Tinbergen Institute Discussion paper, 2008.

6DAVIS, R. A.; DUNSMUIR, W. T.; STREETT, S. B. Observation-driven models for poisson counts. *Biometrika*, Oxford University Press, v. 90, n. 4, p. 777–790, 2003.

Davis et al. 2021DAVIS, R. A. et al. Count time series: A methodological review. *Journal of the American Statistical Association*, Taylor & Francis, v. 116, n. 535, p. 1533–1547, 2021.

Davis e Liu 2012DAVIS, R. A.; LIU, H. Theory and inference for a class of observation-driven models with application to time series of counts. *arXiv preprint arXiv:1204.3915*, 2012.

Dunsmuir e Scott 2015DUNSMUIR, W. T.; SCOTT, D. J. The glarma package for observation-driven time series regression of counts. *Journal of Statistical Software*, v. 67, p. 1–36, 2015.

10FERLAND, R.; LATOUR, A.; ORAICHI, D. Integer-valued garch process. *Journal of time series analysis*, Wiley Online Library, v. 27, n. 6, p. 923–942, 2006.

Fokianos 2011FOKIANOS, K. Some recent progress in count time series. *Statistics*, Taylor & Francis, v. 45, n. 1, p. 49–58, 2011.

Fokianos 2012FOKIANOS, K. Count time series models. In: *Handbook of statistics*. [S.l.]: Elsevier, 2012. v. 30, p. 315–347.

13FOKIANOS, K.; RAHBEK, A.; TJØSTHEIM, D. Poisson autoregression. *Journal of the American Statistical Association*, Taylor & Francis, v. 104, n. 488, p. 1430–1439, 2009.

Genetski 1978GENETSKI, S. A. R. J. Longrange forecasting: From crystal ball to computer. *jscottarmstrong.com*, 1978.

Heinen 2003HEINEN, A. Modelling time series count data: an autoregressive conditional poisson model. *Available at SSRN 1117187*, 2003.

Hyndman e Athanasopoulos 2021HYNDMAN, R. J.; ATHANASOPOULOS, G. *Forecasting: Principles and Practice (3rd ed.)*. Melbourne, Australia: OTexts, 2021. Disponível em: <https://otexts.com/fpp3/>.

Hyndman e Khandakar 2008HYNDMAN, R. J.; KHANDAKAR, Y. Automatic time series forecasting: the forecast package for r. *Journal of statistical software*, v. 27, p. 1–22, 2008.

Hyndman e Koehler 2006HYNDMAN, R. J.; KOEHLER, A. B. Another look at measures of forecast accuracy. *International journal of forecasting*, Elsevier, v. 22, n. 4, p. 679–688, 2006.

Miranda 2014MIRANDA, I. P. Heringer de. *Comparação de diferentes Métodos de Previsão em Séries Temporais com valores discrepantes*. Monografia (Monografia) — Universidade Federal de Juiz de Fora, Juiz de Fora, MG, 2014.

Nelder e Wedderburn 1972NELDER, J. A.; WEDDERBURN, R. W. Generalized linear models. *Journal of the Royal Statistical Society Series A: Statistics in Society*, Oxford University Press, v. 135, n. 3, p. 370–384, 1972.

Pacheco 2021PACHECO, P. H. d. M. *Modelagem de dados longitudinais complexos no R: desenvolvimento de um pacote estatístico*. 2021. Monografia (Graduação em Estatística) Universidade Federal de Juiz de Fora, Juiz de Fora, MG. Disponível em: <https://repositorio.ufjf.br/jspui/bitstream/ufjf/13437/1/pedrohenriquedemesquitapacheco.pdf>.

Parzen 1961PARZEN, E. An approach to time series analysis. *The Annals of Mathematical Statistics*, Institute of Mathematical Statistics, v. 32, n. 4, p. 951–989, 1961.

Prado 2022PRADO, M. Relato de experiência: comunicando ciência com a plataforma jf de acompanhamento estatístico da pandemia de covid-19. *Intercom Sociedade Brasileira de Estudos Interdisciplinares da Comunicação 45ž Congresso Brasileiro de Ciências da Comunicação UFPB*, 2022. Acessado em [data de acesso]. Disponível em: <https://db138ea9-cf2c-4ef8-9a00-41762a1078d0.filesusr.com/ugd/b82b28$_5$ac48403829a4274b142ab42f8b80c8e.pdf>.

Rydberg 2000RYDBERG, T. H. Realistic statistical modelling of financial data. *International Statistical Review*, Wiley Online Library, v. 68, n. 3, p. 233–258, 2000.

Streett 2000STREETT, S. B. *Some observation driven models for time series*. [S.l.]: Colorado State University, 2000.

Tjøstheim 2012TJØSTHEIM, D. Some recent theory for autoregressive count time series. *Test*, Springer, v. 21, n. 3, p. 413–438, 2012.

Tran e Reed 2004TRAN, N.; REED, D. A. Automatic arima time series modeling for adaptive i/o prefetching. *IEEE Transactions on parallel and distributed systems*, IEEE, v. 15, n. 4, p. 362–377, 2004.

Wang e Li 2011WANG, C.; LI, W. K. On the autopersistence functions and the autopersistence graphs of binary autoregressive time series. *Journal of Time Series Analysis*, Wiley Online Library, v. 32, n. 6, p. 639–646, 2011.

Weiß e Schweer 2015WEISS, C. H.; SCHWEER, S. Detecting overdispersion in inarch (1) processes. *Statistica Neerlandica*, Wiley Online Library, v. 69, n. 3, p. 281–297, 2015.

Wickham 2014WICKHAM, H. Tidy data. *Journal of statistical software*, v. 59, p. 1–23, 2014.

Wickham e Bryan 2023WICKHAM, H.; BRYAN, J. *R packages*. [S.l.]: "O'Reilly Media, Inc.", 2023.

Zou 2006ZOU, H. The adaptive lasso and its oracle properties. *Journal of the American statistical association*, Taylor & Francis, v. 101, n. 476, p. 1418–1429, 2006.

APÊNDICE A – Código utilizado para aplicação

Listing .1 – Trecho de código utilizado na modelagem preditiva

```
1
2
3
4  covid_dataset =
5    dplyr::bind_rows(
6      readr::read_delim("D:/TCC/dados/HIST_PAINEL_COVIDBR_01ago2025/HIST_
         PAINEL_COVIDBR_2020_Parte1_01ago2025.csv", delim = ";"),
7      readr::read_delim("D:/TCC/dados/HIST_PAINEL_COVIDBR_01ago2025/HIST_
         PAINEL_COVIDBR_2020_Parte2_01ago2025.csv", delim = ";"),
8      readr::read_delim("D:/TCC/dados/HIST_PAINEL_COVIDBR_01ago2025/HIST_
         PAINEL_COVIDBR_2021_Parte1_01ago2025.csv", delim = ";"),
9      readr::read_delim("D:/TCC/dados/HIST_PAINEL_COVIDBR_01ago2025/HIST_
         PAINEL_COVIDBR_2021_Parte2_01ago2025.csv", delim = ";"),
10     readr::read_delim("D:/TCC/dados/HIST_PAINEL_COVIDBR_01ago2025/HIST_
         PAINEL_COVIDBR_2022_Parte1_01ago2025.csv", delim = ";"),
11     readr::read_delim("D:/TCC/dados/HIST_PAINEL_COVIDBR_01ago2025/HIST_
         PAINEL_COVIDBR_2022_Parte2_01ago2025.csv", delim = ";"),
12     readr::read_delim("D:/TCC/dados/HIST_PAINEL_COVIDBR_01ago2025/HIST_
         PAINEL_COVIDBR_2023_Parte1_01ago2025.csv", delim = ";"),
13     readr::read_delim("D:/TCC/dados/HIST_PAINEL_COVIDBR_01ago2025/HIST_
         PAINEL_COVIDBR_2023_Parte2_01ago2025.csv", delim = ";"),
14     readr::read_delim("D:/TCC/dados/HIST_PAINEL_COVIDBR_01ago2025/HIST_
         PAINEL_COVIDBR_2024_Parte1_01ago2025.csv", delim = ";"),
15     readr::read_delim("D:/TCC/dados/HIST_PAINEL_COVIDBR_01ago2025/HIST_
         PAINEL_COVIDBR_2024_Parte2_01ago2025.csv", delim = ";")
16   )
17
18 covid_br_ts = covid_dataset |>
19   dplyr::filter(regiao == "Brasil") |>
20   dplyr::select(-2,-3,-4,-5,-6,-7) |>
21   dplyr::mutate(semana_epi = lubridate::epiweek(data)) |>
22   dplyr::relocate(semana_epi, .before = data) |>
23   dplyr::select(-data, -semanaEpi) |>
24   dplyr::group_by(semana_epi) |>
25   dplyr::summarise(
26     casos_conf = sum(casosNovos),
27     obitos_conf = sum(obitosNovos)
28   ) |>
29   tsibble::as_tsibble(index = semana_epi)
30
31 ###############
32 set.seed(4390)
33
```

```r
# Amostrando indices de municipos via amostragem estratificada

casos_por_municipio = covid_dataset |>
  dplyr::filter(
    estado == "MG",
    is.na(codmun) == F,
    data >= lubridate::ymd("2024-01-01"),
    data <= lubridate::ymd("2024-12-31")
  ) |>
  dplyr::group_by(codmun, municipio) |>
  dplyr::summarise(total_casos = sum(casosNovos, na.rm = TRUE), .groups
     = "drop") |>
  dplyr::filter(total_casos >= 10) |>  # remover municípios com menos de
      10 casos
  dplyr::mutate(
    estrato = dplyr::case_when(
      total_casos > 250 ~ "alto",
      total_casos >= 100 ~ "medio",
      total_casos >= 10 ~ "baixo"
    )
  ) |>
  dplyr::group_by(estrato) |>
  dplyr::slice_sample(n = 13) |>
  dplyr::ungroup()

obitos_por_municipio = covid_dataset |>
  dplyr::filter(
    estado == "MG",
    is.na(codmun) == F,
    data >= lubridate::ymd("2024-01-01"),
    data <= lubridate::ymd("2024-12-31")
  ) |>
  dplyr::group_by(codmun, municipio) |>
  dplyr::summarise(total_obitos = sum(obitosNovos, na.rm = TRUE), .
     groups = "drop") |>
  dplyr::filter(total_obitos >= 1) |>  # remover municípios com menos de
      5 óbitos
  dplyr::mutate(
    estrato = dplyr::case_when(
      total_obitos > 20 ~ "alto",
      total_obitos >= 10 ~ "medio",
      total_obitos >= 5 ~ "baixo",
      total_obitos >= 1 ~ "baixissimo"
    )
  ) |>
  dplyr::group_by(estrato) |>
  dplyr::slice_sample(n = 12) |>
```

```
77    dplyr::ungroup()
78
79  #1. Retirando dados por municipios amostrados
80  dados_obitos_mg = covid_dataset |>
81    dplyr::filter(
82      estado == "MG",
83      data >= lubridate::ymd("2024-01-01") & data <= lubridate::ymd("
          2024-12-31"),
84    ) |>
85    dplyr::filter(codmun %in% obitos_por_municipio$codmun) |>
86    dplyr::select(-regiao,
87                  -estado,
88                  -coduf,
89                  -codRegiaoSaude,
90                  -nomeRegiaoSaude,
91                  -populacaoTCU2019,
92    ) |>
93    dplyr::mutate(semana_epi = lubridate::epiweek(data)) |>
94    dplyr::relocate(semana_epi, .before = data) |>
95    dplyr::select(-data, -semanaEpi) |>
96    dplyr::group_by(semana_epi, municipio) |>
97    dplyr::summarise(
98      obitos_conf = sum(obitosNovos)
99    ) |>
100   dplyr::mutate(obitos_conf = dplyr::if_else(obitos_conf < 0, 0, obitos_
        conf)) |>
101   dplyr::ungroup() |>
102   dplyr::mutate(obitos_conf = dplyr::if_else(obitos_conf >=0,obitos_conf
        ,0)) |>
103   tsibble::as_tsibble(index = semana_epi, key = municipio)
104
105
106 dados_casos_mg = covid_dataset |>
107   dplyr::filter(
108     estado == "MG",
109     data >= lubridate::ymd("2024-01-01") & data <= lubridate::ymd("
          2024-12-31"),
110   ) |>
111   dplyr::select(-regiao,
112                 -estado,
113                 -coduf,
114                 -codRegiaoSaude,
115                 -nomeRegiaoSaude,
116                 -populacaoTCU2019,
117   ) |>
118   dplyr::filter(codmun %in% casos_por_municipio$codmun) |>
119   dplyr::mutate(semana_epi = lubridate::epiweek(data)) |>
```

```r
  dplyr::relocate(semana_epi, .before = data) |>
  dplyr::select(-data, -semanaEpi) |>
  dplyr::group_by(semana_epi, municipio) |>
  dplyr::summarise(
    casos_conf = sum(casosNovos)
  ) |>
  dplyr::mutate(casos_conf = dplyr::if_else(casos_conf < 0, 0, casos_conf)) |>
  dplyr::ungroup() |>
  tsibble::as_tsibble(index = semana_epi, key = municipio)


# 3.
# Modelando Casos Confirmados - Via TSCV

casos_mg_model_tscv = dados_casos_mg |>
  tsibble::stretch_tsibble(.init = 48, .step = 1) |>
  fabletools::model(
    arma = ARIMA(casos_conf),
    nnetar = NNETAR(casos_conf),
    ing_naive = INGARCH(casos_conf, algorithm = "naive_search", distr = "poisson"),
    ing_arma = INGARCH(casos_conf, algorithm = "arma_to_ingarch", distr = "poisson"),
    ing_lasso = INGARCH(casos_conf , algorithm = "post_lasso", distr = "poisson"),
    gla_arma = GLARMA(casos_conf, algorithm = "arma_to_glarma", distr = "poisson"),
    gla_naive = GLARMA(casos_conf, algorithm = "naive_search_glarma", distr = "poisson")
  )


## 3.1 - Hipotese de RMSE
casos_mg_forecast_tscv = casos_mg_model_tscv |>
  forecast(h = 1) |>
  accuracy(dados_casos_mg) |>
  dplyr::filter(is.nan(RMSE) == F)


casos_mg_forecast_tscv_raw = casos_mg_model_tscv |>
  dplyr::select(arma, nnetar) |>
  dplyr::slice_head(n = 4) |>
  forecast(h = 1)
```

```r
casos_mg_tscv_estratos = casos_mg_forecast_tscv |>
  dplyr::left_join(casos_por_municipio, by = "municipio") |>
  dplyr::select(1,2,5, 14) |>
  tidyr::pivot_wider(names_from = .model, values_from = RMSE)




tempo_casos_mg_forecast_tscv = microbenchmark(
  dados_bench |>
    fabletools::model(arma = ARIMA(casos_conf)),
  dados_bench |>
    fabletools::model(ing_naive = INGARCH(casos_conf, algorithm = "naive_search", distr = "poisson")),
  dados_bench |>
    fabletools::model(ing_arma = INGARCH(casos_conf, algorithm = "arma_to_ingarch", distr = "poisson")),
  dados_bench |>
    fabletools::model(ing_lasso = INGARCH(casos_conf , algorithm = "post_lasso", distr = "poisson")),
  dados_bench |>
    fabletools::model(gla_arma = GLARMA(casos_conf, algorithm = "arma_to_glarma", distr = "poisson")),
  dados_bench |>
    fabletools::model(gla_naive = GLARMA(casos_conf, algorithm = "naive_search_glarma", distr = "poisson")),
  times = 1)

##

# Modelando Casos Confirmados - Via OOS
casos_mg_model_oos = dados_casos_mg |>
  dplyr::filter(semana_epi <= 48) |>
  fabletools::model(
    arma = ARIMA(casos_conf),
    nnetar = NNETAR(casos_conf),
    ing_naive = INGARCH(casos_conf, algorithm = "naive_search", distr = "poisson"),
    ing_arma = INGARCH(casos_conf, algorithm = "arma_to_ingarch", distr = "poisson"),
    ing_lasso = INGARCH(casos_conf , algorithm = "post_lasso", distr = "poisson"),
    gla_arma = GLARMA(casos_conf, algorithm = "arma_to_glarma", distr = "poisson"),
    gla_naive = GLARMA(casos_conf, algorithm = "naive_search_glarma", distr = "poisson")
  )
```

```
198
199 casos_mg_forecast_oss_raw = casos_mg_model_oos |>
200   forecast(h = 4)
201
202 casos_mg_forecast_oss = casos_mg_forecast_oss_raw |>
203   accuracy(dados_casos_mg) |>
204   dplyr::filter(is.nan(RMSE) == F)
205
206
207 casos_mg_oss_estratos = casos_mg_forecast_oss |>
208   dplyr::left_join(casos_por_municipio, by = "municipio") |>
209   dplyr::select(1,2,5, 14) |>
210   tidyr::pivot_wider(names_from = .model, values_from = RMSE)
211
212
213
214 # 4.
215 # Modelando Obitos Confirmados - Via TSCV
216
217 obitos_mg_model_tscv = dados_obitos_mg |>
218   tsibble::stretch_tsibble(.init = 48, .step = 1) |>
219   fabletools::model(
220     arma = ARIMA(obitos_conf),
221     nnetar = NNETAR(obitos_conf),
222     ing_naive = INGARCH(obitos_conf, algorithm = "naive_search", distr =
          "poisson"),
223     ing_arma = INGARCH(obitos_conf, algorithm = "arma_to_ingarch", distr
          = "poisson"),
224     ing_lasso = INGARCH(obitos_conf , algorithm = "post_lasso", distr =
         "poisson"),
225     gla_arma = GLARMA(obitos_conf, algorithm = "arma_to_glarma", distr =
          "poisson"),
226     gla_naive = GLARMA(obitos_conf, algorithm = "naive_search_glarma",
         distr = "poisson")
227   )
228
229
230
231 ## 4.1 - Hipotese de RMSE
232 obitos_mg_forecast_tscv = obitos_mg_model_tscv |>
233   forecast(h = 1) |>
234   accuracy(dados_obitos_mg) |>
235   dplyr::filter(is.nan(RMSE) == F)
236
237
238 obitos_mg_tscv_estratos = obitos_mg_forecast_tscv |>
239   dplyr::left_join(obitos_por_municipio, by = "municipio") |>
```

```
  dplyr::select(1,2,5, 14) |>
  tidyr::pivot_wider(names_from = .model, values_from = RMSE)

# Modelando Obitos Confirmados - Via OSS

obitos_mg_model_oos = dados_obitos_mg |>
  dplyr::filter(semana_epi <= 48) |>
  fabletools::model(
    arma = ARIMA(obitos_conf),
    nnetar = NNETAR(obitos_conf),
    ing_naive = INGARCH(obitos_conf, algorithm = "naive_search", distr = "poisson"),
    ing_arma = INGARCH(obitos_conf, algorithm = "arma_to_ingarch", distr = "poisson"),
    ing_lasso = INGARCH(obitos_conf , algorithm = "post_lasso", distr = "poisson"),
    gla_arma = GLARMA(obitos_conf, algorithm = "arma_to_glarma", distr = "poisson"),
    gla_naive = GLARMA(obitos_conf, algorithm = "naive_search_glarma", distr = "poisson")
  )

obitos_mg_forecast_oss_raw = obitos_mg_model_oos |>
  forecast(h = 4)

obitos_mg_forecast_oss = obitos_mg_forecast_oss_raw |>
  accuracy(dados_obitos_mg) |>
  dplyr::filter(is.nan(RMSE) == F)


obitos_mg_oss_estratos = obitos_mg_forecast_oss |>
  dplyr::left_join(obitos_por_municipio, by = "municipio") |>
  dplyr::select(1,2,5, 14) |>
  tidyr::pivot_wider(names_from = .model, values_from = RMSE)
```

Listing .2 – Códifo utilizado para análise do tempo de execução de cada modelo

```


## --- Definição de métodos e estratos ---
metodos = c("arma", "nnetar", "gla_arma", "gla_naive",
            "ing_arma", "ing_naive", "ing_lasso")

estratos = c("baixo", "medio", "alto")

## --- Inicialização da matriz 3D de tempos ---
tempos_execucao = array(NA, dim = c(length(metodos), 3, length(estratos)),
```

```
                          dimnames = list(
                            metodo = metodos,
                            repeticao = paste0("rep_", 1:3),
                            estrato = estratos
                          ))



## --- Lista para facilitar iteração entre estratos ---
dados_lista = list(baixo = dados_casos_mg_baixo,
                   medio = dados_casos_mg_medio,
                   alto  = dados_casos_mg_alto)

## - Matriz dos modelos

modelos_estim_lista <- vector("list", length(metodos))
names(modelos_estim_lista) <- metodos


## --- Loop principal ---
for (rep in 1:3) {
  cat("\nIniciando repetição", rep, "...\n")

  for (estrato in estratos) {
    cat("  Estrato:", estrato, "\n")
    dados = dados_lista[[estrato]]

    # ARIMA
    t_0 = Sys.time()
    modelos_estim_lista[["arma"]][[estrato]] = fabletools::model(
      dados,
      arma = ARIMA(casos_conf)
    )
    t_1 = Sys.time()
    tempos_execucao["arma", rep, estrato] = as.numeric(t_1 - t_0, units
       = "secs")

    # NNETAR
    t_0 = Sys.time()
    modelos_estim_lista[["nnetar"]][[estrato]] = fabletools::model(
      dados,
      nnetar = NNETAR(casos_conf)
    )
    t_1 = Sys.time()
    tempos_execucao["nnetar", rep, estrato] = as.numeric(t_1 - t_0,
       units = "secs")

```

```
56      # GLARMA (arma_to_glarma)
57      t_0 = Sys.time()
58      modelos_estim_lista[["gla_arma"]][[estrato]] = fabletools::model(
59        dados,
60        gla_arma = GLARMA(casos_conf, algorithm = "arma_to_glarma", distr
            = "poisson")
61      )
62      t_1 = Sys.time()
63      tempos_execucao["gla_arma", rep, estrato] = as.numeric(t_1 - t_0,
          units = "secs")
64
65      # GLARMA (naive_search_glarma)
66      t_0 = Sys.time()
67      modelos_estim_lista[["gla_naive"]][[estrato]] = fabletools::model(
68        dados,
69        gla_naive = GLARMA(casos_conf, algorithm = "naive_search_glarma",
            distr = "poisson")
70      )
71      t_1 = Sys.time()
72      tempos_execucao["gla_naive", rep, estrato] = as.numeric(t_1 - t_0,
          units = "secs")
73
74      # INGARCH (arma_to_ingarch)
75      t_0 = Sys.time()
76      modelos_estim_lista[["ing_arma"]][[estrato]] = fabletools::model(
77        dados,
78        ing_arma = INGARCH(casos_conf, algorithm = "arma_to_ingarch",
            distr = "poisson")
79      )
80      t_1 = Sys.time()
81      tempos_execucao["ing_arma", rep, estrato] = as.numeric(t_1 - t_0,
          units = "secs")
82
83      # INGARCH (naive_search)
84      t_0 = Sys.time()
85      modelos_estim_lista[["ing_naive"]][[estrato]] = fabletools::model(
86        dados,
87        ing_naive = INGARCH(casos_conf, algorithm = "naive_search", distr
            = "poisson")
88      )
89      t_1 = Sys.time()
90      tempos_execucao["ing_naive", rep, estrato] = as.numeric(t_1 - t_0,
          units = "secs")
91
92      # INGARCH (post_lasso)
93      t_0 = Sys.time()
94      modelos_estim_lista[["ing_lasso"]][[estrato]] = fabletools::model(
```

```
95        dados,
96        ing_lasso = INGARCH(casos_conf, algorithm = "post_lasso", distr =
              "poisson")
97      )
98      t_1 = Sys.time()
99      tempos_execucao["ing_lasso", rep, estrato] = as.numeric(t_1 - t_0,
            units = "secs")
100   }
101 }
102
103 tempos_estim_spec = tempos_execucao |>
104   as_tibble(rownames = "metodo") |>
105   pivot_longer(-metodo) |>
106   mutate(estrato = stringr::str_extract(name, "(?<=\\.).+$"),
107          rep = stringr::str_extract(name, "(?<=rep_)[0-9]+")) |>
108   select(-name) |>
109   group_by(metodo, estrato) |>
110   summarise(tempo = mean(value))
111
112 tempos_estim_spec |>
113   pivot_wider(names_from = metodo, values_from = tempo) |>
114   relocate(nnetar, .before = gla_arma)
115
116
117 # Definir métodos e estratos
118 metodos = c("arma", "nnetar", "gla_arma", "gla_naive",
119             "ing_arma", "ing_naive", "ing_lasso")
120 estratos = c("baixo", "medio", "alto")
121
122 # Inicializar array para armazenar tempos de previsão (método 𝔼
      repetição 𝔼 estrato)
123 tempos_predict = array(NA,
124                        dim = c(length(metodos), 3, length(estratos)),
125                        dimnames = list(metodos, paste0("rep_", 1:3),
                               estratos))
126
127
128 horizonte = 1  # número de passos à frente para forecast
129
130 for (rep in 1:3) {
131   cat("\nRodando␣previsão␣-␣repetição", rep, "...\n")
132
133   for (e in estratos) {
134     cat("␣Estrato:", toupper(e), "\n")
135
136     for (m in metodos) {
137       t_0 = Sys.time()
```

```
138         modelos_estim_lista[[m]][[e]] |> forecast(h = horizonte)
139         t_1 = Sys.time()
140 
141         tempos_predict[m, rep, e] = as.numeric(t_1 - t_0, units = "secs")
142       }
143     }
144 }
145 
146 tempos_predict_spec = tempos_predict |>
147   as_tibble(rownames = "metodo") |>
148   pivot_longer(-metodo) |>
149   mutate(estrato = stringr::str_extract(name, "(?<=\\.).+$"),
150          rep = stringr::str_extract(name, "(?<=rep_)[0-9]+")) |>
151   select(-name) |>
152   group_by(metodo, estrato) |>
153   summarise(tempo = mean(value))
154 
155 tempos_predict_spec |>
156   pivot_wider(names_from = metodo, values_from = tempo) |>
157   relocate(nnetar, .before = gla_arma)
158 
159 
160 # Loop de 3 repetições
161 for (rep in 1:3) {
162   cat("\nRodando previsão - repetição", rep, "...\n")
163 
164   # ---------- ESTRATO BAIXO ----------
165   cat(" Estrato: BAIXO\n")
166 
167   t_0 = Sys.time(); modelos_estim_lista$arma$baixo |> forecast(h = 1); t
      _1 = Sys.time()
168   tempos_predict["arma", rep, "baixo"] = as.numeric(t_1 - t_0, units = "
      secs")
169 
170   t_0 = Sys.time(); modelos_estim_lista$nnetar$baixo |> forecast(h = 1);
       t_1 = Sys.time()
171   tempos_predict["nnetar", rep, "baixo"] = as.numeric(t_1 - t_0, units =
       "secs")
172 
173   t_0 = Sys.time(); modelos_estim_lista$gla_arma$baixo |> forecast(h =
      1); t_1 = Sys.time()
174   tempos_predict["gla_arma", rep, "baixo"] = as.numeric(t_1 - t_0, units
       = "secs")
175 
176   t_0 = Sys.time(); modelos_estim_lista$gla_naive$baixo |> forecast(h =
      1); t_1 = Sys.time()
177   tempos_predict["gla_naive", rep, "baixo"] = as.numeric(t_1 - t_0,
```

```
    units = "secs")
178
179 t_0 = Sys.time(); modelos_estim_lista$ing_arma$baixo |> forecast(h =
    1); t_1 = Sys.time()
180 tempos_predict["ing_arma", rep, "baixo"] = as.numeric(t_1 - t_0, units
     = "secs")
181
182 t_0 = Sys.time(); modelos_estim_lista$ing_naive$baixo |> forecast(h =
    1); t_1 = Sys.time()
183 tempos_predict["ing_naive", rep, "baixo"] = as.numeric(t_1 - t_0,
    units = "secs")
184
185 t_0 = Sys.time(); modelos_estim_lista$ing_lasso$baixo |> forecast(h =
    1); t_1 = Sys.time()
186 tempos_predict["ing_lasso", rep, "baixo"] = as.numeric(t_1 - t_0,
    units = "secs")
187
188 # ---------- ESTRATO MÉDIO ----------
189 cat("␣Estrato:␣MÉDIO\n")
190
191 t_0 = Sys.time(); modelos_estim_lista$arma$medio |> forecast(h = 1); t
    _1 = Sys.time()
192 tempos_predict["arma", rep, "medio"] = as.numeric(t_1 - t_0, units = "
    secs")
193
194 t_0 = Sys.time(); modelos_estim_lista$nnetar$medio |> forecast(h = 1);
     t_1 = Sys.time()
195 tempos_predict["nnetar", rep, "medio"] = as.numeric(t_1 - t_0, units =
     "secs")
196
197 t_0 = Sys.time(); modelos_estim_lista$gla_arma$medio |> forecast(h =
    1); t_1 = Sys.time()
198 tempos_predict["gla_arma", rep, "medio"] = as.numeric(t_1 - t_0, units
     = "secs")
199
200 t_0 = Sys.time(); gla_naive_medio_estim |> forecast(h = 1); t_1 = Sys.
    time()
201 tempos_predict["gla_naive", rep, "medio"] = as.numeric(t_1 - t_0,
    units = "secs")
202
203 t_0 = Sys.time(); ing_arma_medio_estim |> forecast(h = 1); t_1 = Sys.
    time()
204 tempos_predict["ing_arma", rep, "medio"] = as.numeric(t_1 - t_0, units
     = "secs")
205
206 t_0 = Sys.time(); ing_naive_medio_estim |> forecast(h = 1); t_1 = Sys.
    time()
```

```
207   tempos_predict["ing_naive", rep, "medio"] = as.numeric(t_1 - t_0,
          units = "secs")
208
209   t_0 = Sys.time(); ing_lasso_medio_estim |> forecast(h = 1); t_1 = Sys.
          time()
210   tempos_predict["ing_lasso", rep, "medio"] = as.numeric(t_1 - t_0,
          units = "secs")
211
212   # ---------- ESTRATO ALTO ----------
213   cat("␣Estrato:␣ALTO\n")
214
215   t_0 = Sys.time(); arma_alto_estim |> forecast(h = 1); t_1 = Sys.time()
216   tempos_predict["arma", rep, "alto"] = as.numeric(t_1 - t_0, units = "
          secs")
217
218   t_0 = Sys.time(); nnetar_alto_estim |> forecast(h = 1); t_1 = Sys.time
          ()
219   tempos_predict["nnetar", rep, "alto"] = as.numeric(t_1 - t_0, units =
          "secs")
220
221   t_0 = Sys.time(); gla_arma_alto_estim |> forecast(h = 1); t_1 = Sys.
          time()
222   tempos_predict["gla_arma", rep, "alto"] = as.numeric(t_1 - t_0, units
          = "secs")
223
224   t_0 = Sys.time(); gla_naive_alto_estim |> forecast(h = 1); t_1 = Sys.
          time()
225   tempos_predict["gla_naive", rep, "alto"] = as.numeric(t_1 - t_0, units
           = "secs")
226
227   t_0 = Sys.time(); ing_arma_alto_estim |> forecast(h = 1); t_1 = Sys.
          time()
228   tempos_predict["ing_arma", rep, "alto"] = as.numeric(t_1 - t_0, units
          = "secs")
229
230   t_0 = Sys.time(); ing_naive_alto_estim |> forecast(h = 1); t_1 = Sys.
          time()
231   tempos_predict["ing_naive", rep, "alto"] = as.numeric(t_1 - t_0, units
           = "secs")
232
233   t_0 = Sys.time(); ing_lasso_alto_estim |> forecast(h = 1); t_1 = Sys.
          time()
234   tempos_predict["ing_lasso", rep, "alto"] = as.numeric(t_1 - t_0, units
           = "secs")
235 }
236
237 ## Tempo de estimação do modelo
```

```
# Definir métodos e inicializar matriz de tempos de previsão
metodos = c("arma", "nnetar", "gla_arma", "gla_naive",
            "ing_arma", "ing_naive", "ing_lasso")

tempos_predict = matrix(NA, nrow = length(metodos), ncol = 3,
                        dimnames = list(metodos, paste0("rep_", 1:3)))

# Loop de 3 repetições
for (rep in 1:3) {

  cat("\nRodando previsão - repetição", rep, "...\n")

  # ARIMA
  t_0 = Sys.time()
  arma_baixo_estim |> forecast(h = 1)
  t_1 = Sys.time()
  tempos_predict["arma", rep] = as.numeric(t_1 - t_0, units = "secs")

  # NNETAR
  t_0 = Sys.time()
  nnetar_baixo_estim |> forecast(h = 1)
  t_1 = Sys.time()
  tempos_predict["nnetar", rep] = as.numeric(t_1 - t_0, units = "secs")

  # GLARMA (arma_to_glarma)
  t_0 = Sys.time()
  gla_arma_baixo_estim |> forecast(h = 1)
  t_1 = Sys.time()
  tempos_predict["gla_arma", rep] = as.numeric(t_1 - t_0, units = "secs"
      )

  # GLARMA (naive_search_glarma)
  t_0 = Sys.time()
  gla_naive_baixo_estim |> forecast(h = 1)
  t_1 = Sys.time()
  tempos_predict["gla_naive", rep] = as.numeric(t_1 - t_0, units = "secs
      ")

  # INGARCH (arma_to_ingarch)
  t_0 = Sys.time()
  ing_arma_baixo_estim |> forecast(h = 1)
  t_1 = Sys.time()
  tempos_predict["ing_arma", rep] = as.numeric(t_1 - t_0, units = "secs"
      )

  # INGARCH (naive_search)
  t_0 = Sys.time()
```

```
  ing_naive_baixo_estim |> forecast(h = 1)
  t_1 = Sys.time()
  tempos_predict["ing_naive", rep] = as.numeric(t_1 - t_0, units = "secs
      ")

  # INGARCH (post_lasso)
  t_0 = Sys.time()
  ing_lasso_baixo_estim |> forecast(h = 1)
  t_1 = Sys.time()
  tempos_predict["ing_lasso", rep] = as.numeric(t_1 - t_0, units = "secs
      ")
}



tempo_casos_mg_forecast_tscv = microbenchmark(
  dados_bench |>
    fabletools::model(arma = ARIMA(casos_conf)),
  dados_bench |>
    fabletools::model(ing_naive = INGARCH(casos_conf, algorithm = "naive
        _search", distr = "poisson")),
  dados_bench |>
    fabletools::model(ing_arma = INGARCH(casos_conf, algorithm = "arma_
        to_ingarch", distr = "poisson")),
  dados_bench |>
    fabletools::model(ing_lasso = INGARCH(casos_conf , algorithm = "post
        _lasso", distr = "poisson")),
  dados_bench |>
    fabletools::model(gla_arma = GLARMA(casos_conf, algorithm = "arma_to
        _glarma", distr = "poisson")),
  dados_bench |>
    fabletools::model(gla_naive = GLARMA(casos_conf, algorithm = "naive_
        search_glarma", distr = "poisson")),
  times = 1)
```