Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Modelação Estatística e Análise de Dados

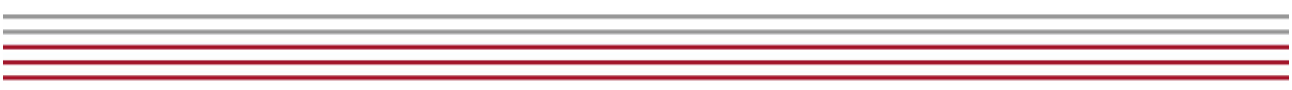
Dissertação

Detecção de Mudanças de Estrutura em Séries Temporais

Ludomilo Rebelo Almeida

Orientador(es) | Dulce Maria de Oliveira Gomes
Lígia Henriques-Rodrigues

Évora 2025



Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Modelação Estatística e Análise de Dados

Dissertação

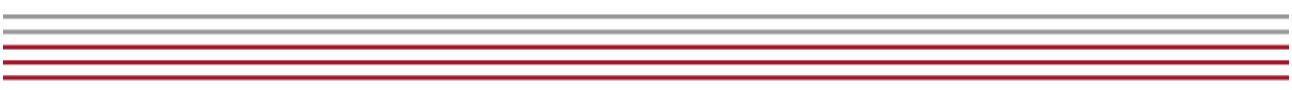
Detecção de Mudanças de Estrutura em Séries Temporais

Ludomilo Rebelo Almeida

Orientador(es) | Dulce Maria de Oliveira Gomes

Lígia Henriques-Rodrigues

Évora 2025



A dissertação foi objeto de apreciação e discussão pública pelo seguinte júri nomeado pelo Diretor da Escola de Ciências e Tecnologia:

Presidente	|	Dulce Gamito Pereira (Universidade de Évora)
Vogais	|	Adelaide de Fátima Baptista Valente Freitas (Universidade de Aveiro) (Arguente) Dulce Maria de Oliveira Gomes (Universidade de Évora)

Évora 2025

Agradecimentos

Em primeiro lugar, agradeço a **Deus**, pela vida, pela força e pela sabedoria concedidas ao longo deste percurso académico e pessoal. Sem a Sua presença e bênção nada disto teria sido possível.

Expresso a minha sincera gratidão às minhas orientadoras, **Doutora Dulce Gomes** e **Doutora Lígia Henriques**, pela orientação científica, pela constante disponibilidade, pelas valiosas sugestões e pela dedicação em cada etapa deste trabalho. O rigor, a paciência e o incentivo de ambas foram determinantes para a concretização deste estudo.

Agradeço igualmente a todos os professores do Mestrado, pelo conhecimento transmitido, pelo apoio e pelas contribuições que enriqueceram a minha formação académica.

Deixo também o meu reconhecimento à **Fundação Calouste Gulbenkian**, pelo apoio concedido através da bolsa que possibilitou a realização e concretização desta grande etapa da minha vida.

Agradeço profundamente à minha família e amigos, pelo apoio incondicional, pela compreensão nos momentos mais exigentes e pela motivação ao longo desta caminhada.

A todos, o meu mais sentido obrigado.

Resumo

Detecção de mudanças de estruturas em séries temporais

O estudo das séries temporais é essencial para compreender e prever fenómenos em diversas áreas, mas a presença de mudanças estruturais compromete os pressupostos de estacionariedade e a fiabilidade das previsões. O estudo dedica-se à análise e comparação dos métodos estatísticos e computacionais de deteção de mudanças de estrutura, nomeadamente em média, tendência e forma da distribuição, incluindo o comportamento das caudas. São analisadas as limitações dos métodos clássicos de segmentação da série, em contextos com dependência serial e falha de normalidade. Através de simulações de Monte Carlo com séries de diferentes propriedades —distribuição, autocorrelação, dimensão e presença de *outliers* — avalia-se a eficácia e robustez dos métodos. Procura-se ainda estudar a deteção de mudanças na distribuição GEV em contexto de dependência serial, usando a estatística de teste `CUSUM` adaptada aos métodos `PWM` e `GPWM`.

Palavras Chaves: Séries temporais, mudanças de estrutura, distribuição generalizada de valores extremos, simulação de Monte Carlo.

Abstract

Structural change detection in time series

Time series analysis plays a key role in understanding and forecasting phenomena across many fields. However, structural changes can violate the assumption of stationarity, reducing model reliability and forecast accuracy. This study focuses on the analysis and comparison of statistical and computational methods for detecting structural changes, including shifts in mean, trend, and distribution shape, with particular attention to tail behavior. The limitations of classical segmentation methods are examined in contexts characterized by serial dependence and deviations from normality. Through Monte Carlo simulations of series with different properties—distribution, autocorrelation, sample size, and the presence of *outliers*—the effectiveness and robustness of the methods are assessed. Furthermore, the study investigates the detection of changes in the Generalized Extreme Value (GEV) distribution under serial dependence, using a `CUSUM`-type test statistic adapted to the `PWM` and `GPWM` methods.

Keywords: Time series, structural change, Generalized Extreme Value distribution, Monte Carlo simulation.

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

Acrónimo	Descrição
ACF	Função de autocorrelação (do inglês, *Autocorrelation Function*)
AIC	Critério de informação de Akaike (do inglês, *Akaike Information Criterion*)
BIC	Critério de informação bayesiano (do inglês, *Bayesian Information Criterion*)
BP	Algoritmo de Bai–Perron (do inglês, *Bai–Perron Algorithm*)
BS	Segmentação binária (do inglês, *Binary Segmentation*)
CDF	Função de distribuição acumulada (do inglês, *Cumulative Distribution Function*)
CUSUM	Gráfico de soma cumulativa (do inglês, *Cumulative Sum Control Chart*)
GARCH	Heterocedasticidade condicional autorregressiva generalizada (do inglês, *Generalized Autoregressive Conditional Heteroskedasticity*)
GEV	Distribuição de valores extremos generalizada (do inglês, *Generalized Extreme Value Distribution*)
GPWM	Momentos ponderados por probabilidade generalizados (do inglês, *Generalized Probability Weighted Moments*)
MBIC	Critério de informação bayesiano modificado (do inglês, *Modified Bayesian Information Criterion*)
MMC	Método de Monte Carlo (do inglês, *Monte Carlo Method*)
MV	Máxima verossimilhança (do inglês, *Maximum Likelihood*)
OLS	Mínimos quadrados ordinários (do inglês, *Ordinary Least Squares*)
OP	Algoritmo de partição ótima (do inglês, *Optimal Partitioning Algorithm*)
PACF	Função de autocorrelação parcial (do inglês, *Partial Autocorrelation Function*)
PL	PELT (do inglês, *Pruned Exact Linear Time Algorithm*)
PWM	Momentos ponderados por probabilidade (do inglês, *Probability Weighted Moments*)
WBS	Wild Binary Segmentation

Sumário

Capítulo 1

Introdução

O estudo das séries temporais tem vindo a ganhar crescente relevância nos últimos anos, devido à necessidade de compreender e prever a evolução de determinados fenómenos em áreas como a economia, as finanças, a climatologia, a saúde, a engenharia e as ciências sociais. No entanto, estes fenómenos estão muitas das vezes sujeitos a mudanças que podem afetar a estrutura subjacente dos dados ao longo do tempo.
Este problema pode representar um grande obstáculo na modelação e previsão de séries temporais, sobretudo na modelação de séries através de modelos do tipo Autoregressivo de Média Móvel (ARMA) (ou Autoregressivo Integrado de Média Móvel (ARIMA)), que partem do pressuposto que está estacionária (ou que pode tornar-se estacionária através de diferenciações). As quebras estruturais quebram este pressuposto de estacionariedade ao introduzir mudanças abruptas na média, tendência e/ou variância da série.
Pelo que basta a existência de uma única mudança de estrutura na série para afetar a modelação e a consequentemente a previsão de valores futuros. A existência de múltiplas mudanças de estrutura agravam ainda mais o problema, sobretudo quando o número destas mudanças é desconhecido, bem como em que tempo ocorreram.

A análise de mudanças de estrutura requer, em primeiro lugar, a avaliação da significância estatística das eventuais mudanças e, em caso afirmativo, a estimação do número de mudanças e as suas possíveis localizações (Chen et al. (2000)). Um dos primeiros métodos para deteção de mudança de estrutura é o CUSUM (`cumulative sum control chart`), inicialmente proposto por Page (1954), para deteção de mudança na média em processos contínuos. Este método na sua formulação clássica (`paramétrica`) assume geralmente condições fortes, como normalidade e variância constante, dado que poderá ter melhor desempenho em séries estacionárias, ou seja em processos que não mudam ao longo do tempo exceto pelas mudanças que se querem detetar.
Conforme discutido em Aue and Horváth (2013), a versão não paramétrica do processo CUSUM procura relaxar algumas condições iniciais do método, como a independência e a normalidade. O artigo aborda a aplicação do método em situações em que existe dependência serial, heterocedasticidade ou variância não constante, e examina de que forma os métodos clássicos devem ser adaptados e quais condições adicionais são necessárias para garantir a validade estatística.

Embora tenham sido realizados alguns avanços nesta área, como a implementação de métodos computacionais, estatisticamente eficientes (Zeileis et al. (2002a), Killick and Eckley (2014)), incluindo métodos modernos baseados em aprendizagem automática, os desafios continuam, principalmente em séries com `mudanças subtis ou pouco visíveis` (Salman et al. (2024)), `séries autocorelacionadas`, `valores atípicos-outliers-`(Fearnhead and Rigaill (2019)) ou ainda em cenários onde há `falha de normalidade` — já que a maioria dos métodos clássicos de deteção requer a normalidade ou pelo menos uma aproximação à normalidade — o que constitui uma limitação adicional, pois em numerosos contextos práticos os dados não seguem essa distribuição. O presente trabalho valoriza-se precisamente por considerar cenários que não verificam essas condições. Estes métodos clássicos, com maior aplicação atual podem ser encontrados em Killick et al. (2010), onde apresentam `métodos de segmentação`, que usam funções de custo, para identificar segmentos da série que contêm mudanças, nomeadamente mudanças na média, variância e ambas simultaneamente, fornecendo ainda uma explicação completa com código aberto.

Para além destas, é igualmente importante considerar mudanças na tendência da série. Identificar corretamente este tipo de mudança é crucial, pois a presença de diferentes regimes de crescimento ou de declínio pode afetar de forma significativa a modelação e a previsão. Trabalhos clássicos nesta área podem ser encontrados em Zeileis et al. (2002a), onde são apresentados métodos para detetar quebras de estrutura em modelos de regressão linear, incluindo mudanças na média e na tendência da série temporal.
Por vezes o nosso interesse não está unicamente na deteção de mudanças de nível ou na tendência da série, mas sim de mudanças na própria forma da distribuição, em particular no comportamento das caudas. Com recurso a teoria de valores extremos, alguns autores como Kojadinovic and Naveau (2017) propuseram um método inovador para detetar mudanças nos parâmetros da distribuição generalizada dos valores extremos (GEV), baseando-se no método dos momentos ponderados. Este é um resultado útil, especialmente na análise de mudanças no parâmetro de forma, embora possa apresentar limitações em contextos com autocorrelação.

O interesse nesta área de investigação está na crescente necessidade de compreender os mecanismos que governam uma série temporal, de forma a tornar a inferência mais precisa e a ajustar modelos que melhor se adequem aos dados, fornecendo assim contribuições mais eficazes para os decisores que lidam com estes fenómenos no seu dia a dia.

1.1 Objetivos

O principal objetivo deste trabalho é comparar e avaliar métodos estatísticos computacionais, usados para detetar mudanças de estrutura em séries temporais, e avaliar a robustez em cenários mais desafiantes.

Com o intuito de alcançar este propósito, estabelecemos os seguintes objetivos específicos:

- Comparar e avaliar os principais métodos estatísticos computacionais de deteção de mudanças de estrutura, nomeadamente de mudanças de nível, de tendência e na forma da distribuição.
- Simular séries com diferentes propriedades: com distribuição normal, aproximadamente normal e não-normal; diferentes graus de autocorrelação; em presença de outliers; diferentes dimensões e diferentes magnitudes na mudança.
- Investigar a deteção de mudanças na forma da distribuição generalizada de valores extremos, em cenários autocorrelacionados.

1.2 Organização da investigação

Com o intuito de assegurar uma leitura fluída e uma compreensão lógica, a estrutura do trabalho foi organizada em cinco capítulos principais, cada um abordando diferentes dimensões do problema em estudo.

O Capítulo 1 corresponde à *Introdução*. Neste capítulo, são apresentados o enquadramento geral do tema, a motivação para o desenvolvimento deste trabalho, os objetivos da investigação e uma breve visão global da estrutura da tese. Este capítulo estabelece o ponto de partida para a compreensão do problema e a sua relevância para a análise de séries temporais.

No Capítulo 2 são discutidos os conceitos *fundamentais* de séries temporais e eventos extremos. É também feita uma exposição dos principais tipos de quebras de estrutura que podem ocorrer em séries temporais.

O Capítulo 3 aborda a *Metodologia*. Este capítulo descreve, detalhadamente, os métodos utilizados ao longo do trabalho. Inicia-se com a apresentação dos métodos de segmentação e de programação dinâmica, aplicado na deteção de mudanças de nível e de tendência. Introduz-se ainda o processo CUSUM clássico e recursivo, bem como outros testes aplicados à deteção de mudanças. Abordamos a metodologia proposta por Kojadinovic and Naveau (2017), que visa a deteção de mudanças nos parâmetros da distribuição generalizada de valores extremos, com foco particular na mudança do parâmetro de forma, utilizando os métodos dos momentos ponderados (PWM) e dos momentos ponderados generalizados (GPWM).

O Capítulo 4 é reservado à *Análise Computacional*. Neste capítulo são avaliadas, através de simulações de Monte Carlo com recurso do software `R Cran`, a eficácia e a robustez dos diferentes métodos (clássicos e recentes) na deteção de mudanças de estrutura. São considerados diversos cenários: variações na magnitude das mudanças de nível, séries simuladas com diferentes níveis de dependência temporal, presença de *outliers* e aproximação à distribuição onde a suposição de normalidade não é atendida. A análise inclui a avaliação do desempenho na deteção de mudanças

na tendência. É explorada a aplicação de métodos baseados na Teoria de Valores Extremos para detetar mudanças no parâmetro de forma ξ .

Por fim, o Capítulo 5 apresenta as *Conclusões*. São sintetizadas as principais contribuições da investigação em função dos cenários propostos, discutidas as limitações do estudo e sugeridas direções para trabalhos futuros. Este capítulo visa refletir criticamente sobre os resultados obtidos e consolidar o conhecimento produzido ao longo do trabalho.

Capítulo 2

Séries temporais

2.1 Conceitos gerais de séries temporais

Uma série temporal é um conjunto de observações quantitativas que evoluem ao longo do tempo. Os modelos que iremos abordar, modelos do tipo ARMA/ARIMA, são modelos matemáticos construídos com base na relação de dependência (autocorrelação) entre os valores no tempo, isto é, no facto das observações anteriores poderem influenciar as subsequentes. Pelo que, neste contexto, considera-se ainda que as observações serem igualmente espaçadas é um aspecto fundamental.

Para sermos mais rigorosos, uma série temporal é uma realização de um processo estocástico. A fim de compreender melhor o conceito de processo estocástico vamos abordar o conceito de variável aleatória.

Seja $(\Omega, \mathcal{A}, P)$ um espaço de probabilidades. Define-se como variável aleatória uma função mensurável de Ω em S, a que se chama usualmente espaço de estados do processo estocástico. De um modo geral, chama-se variável aleatória, a qualquer aplicação X de Ω em S, ou seja

$$X : \Omega \to S$$

Fazemos agora intervir o tempo (designado abreviadamente pela letra t), que se supõe tomar valores em T, a que usualmente se chama espaço do parâmetro do processo estocástico. O modelo matemático que surge para o processo estocástico será, portanto, uma aplicação X de $T \times \Omega$ em S

$$X : T \times \Omega \to S$$

Quando o tempo é fixo

$$\omega \to X_t(\omega) \equiv X(\omega) \equiv X$$

temos uma v.a. sobre (Ω, A, P) com valores em S.

Quando o "acaso" é fixo

$$t \to X_\omega(t) \equiv X(t) \,(\text{ou} \quad X_t)$$

temos uma trajetória do processo estocástico, ou seja, uma série temporal.

No âmbito da modelação, as propriedades desejáveis dos estimadores são geralmente asseguradas quando se trabalha com séries fracamente estacionárias (ou estacionárias de segunda ordem), ou ainda com séries estritamente estacionárias (esta

definição é de difícil verificação na prática). Estas propriedades são tipicamente verificadas na estabilidade de alguns parâmetros fundamentais da série, como, por exemplo, a média e a variância.

Formalmente, uma série $\{X_t\}_{t\in T}$ diz-se estacionária de segunda ordem se, para todo o t, $E[X_t^2] < +\infty$, tal que:

- $E(X_t) = \mu_t = \mu$ (não depende de t)
- $Var(X_t) = E(X_t - \mu_t)^2 = \sigma^2$ (não depende de t)
- $Cov(X_t, X_s) = \gamma(|t-s|)$, $\forall t, s \in T$

nesta última, a covariância entre duas variáveis observadas nos tempos t e s toma sempre o mesmo valor para todo o t e s, dependendo apenas da diferença de tempo entre as variáveis, $|t-s|$.

Usualmente escreve-se esta última condição como $Cov(X_t, X_{t+k}) = Cov(X_t, X_{t-k}) = \gamma(k)$, $\forall k, t \in T$. À quantidade k chama-se usualmente espaçamento ou *lag*.

Um processo é estritamente estacionário se todas as suas propriedades probabilísticas são invariantes no tempo, ou seja, para qualquer n, k e instantes $t_1, t_2, \ldots, t_n$, a distribuição conjunta de $(X_{t_1}, X_{t_2}, \ldots, X_{t_n})$ é idêntica à de $(X_{t_1+k}, X_{t_2+k}, \ldots, X_{t_n+k})$. Tal como referido, dado que a verificação da estacionariedade estrita é, na prática, frequentemente inviável de verificar, ao longo deste trabalho será considerada a hipótese de que os processos serem fracamente estacionários. Esta abordagem revela-se particularmente pertinente na análise de mudanças de estrutura, contexto em que se admite, por vezes, a estacionariedade por partes (*piecewise*), em virtude de possíveis mudanças de nível em segmento específicos da série temporal, ver Figura 2.1.

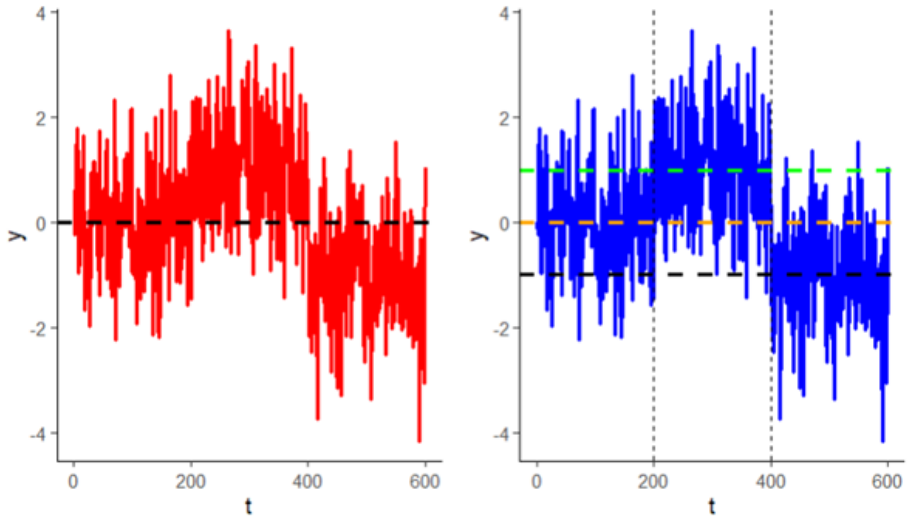


Figura 2.1: Processo estacionário vs estacionário por níveis

As séries que apresentam a componente de tendência e/ou variância não constante não são estacionárias. Na figura 2.2,temos dois processos simulado com tendência (I) e com variância não constante (II).

Para séries com tendência e/ou variância não constante, pode ser útil estabilizar uma delas para ter uma melhor compreensão do comportamento temporal da outra (Nelson and Plosser (1982)). As transformações propostas por Box and Cox (1964) e

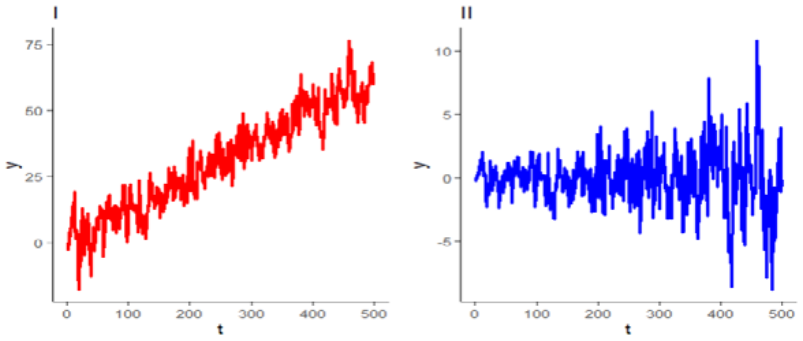


Figura 2.2: Processos não estacionários

mais recentemente de Weisberg (2001) continuam a ser válidas, sendo λ, o parâmetro dessa transformação de Box-Cox que permite estabilizar a variância, um parâmetro a ser estimado. Computacionalmente, pode-se estimar o parâmetro λ pelo método da máxima verosimilhança, obtendo-se assim, a transformação adequada aos dados.[1]

Na modelação de séries temporais, é comum utilizar a metodologia de Box e Jenkins, conforme proposta em Jenkins and Box (1976). Para séries não estacionárias, pode recorrer-se à abordagem de diferenciação da série temporal como um método para remover a tendência linear e/ou a sazonalidade. Tomando como operador de atraso B, tal que $B^n X_t = X_{t-n}$, a primeira diferença, dita simples, é definida por

$$\nabla X_t = (1 - B)X_t = X_t - X_{t-1}.$$

A segunda diferença simples é definida por

$$\nabla^2 X_t = (1 - B)^2 X_t = (1 - 2B + B^2)X_t = X_t - 2X_{t-1} + X_{t-2},$$

e é utilizada quando a primeira diferença simples não é suficiente para tornar a série estacionária em tendência. Na maioria dos casos, a estacionariedade é alcançada com apenas uma diferenciação. No entanto, é importante salientar que este processo pode ser repetido para diferenciações de ordem superior, ou seja, para diferenciações de ordem n. Pode-se aplicar a mesma abordagem de diferenciação (designada por diferenciação sazonal) $\nabla_s X_t = (1-B^s)X_t = X_t - X_{t-s}$, para remover a sazonalidade de uma série temporal, onde s representa o seu período sazonal (ou seja, um período de tempo fixo em que esse padrão de sazonalidade se repete ao longo do tempo).

A classe de modelos de séries temporais usada na modelação e previsão de séries temporais (estacionárias e não estacionárias) que iremos usar neste trabalho é a classe de modelos tradicionais (que tiveram origem nos finais dos anos 20 do século 20 com o trabalho de Yule, e que depois foram evoluindo com a contribuição de vários outros Matemáticos) - mas que continua a ser uma das mais utilizadas, dada a sua enorme flexibilidade e qualidade das previsões - é a classe dos modelos do tipo $\mathbf{ARIMA}(p, d, q)$, em que p corresponde à ordem do processo autoregressivo $\mathbf{AR}(p)$, d o número de diferenciações simples necessárias para se obter uma série estacionária, caso esta não o seja, e q à ordem do processo de médias móveis $\mathbf{MA}(q)$. Se

[1] A transformação de Box-Cox (Box e Cox, 1964) é dada por:

$$y^{(\lambda)} = \begin{cases} \dfrac{y^\lambda - 1}{\lambda}, & \lambda \neq 0, \\ \ln(y), & \lambda = 0,\ y > 0, \end{cases}$$

d=0 estamos, portanto, perante um caso particular deste modelo, nomeadamente, perante um modelo do tipo $\mathbf{ARMA}(p, q)$, apropriado para modelar e prever séries estacionárias.

AR(p)
Seja X_t, $t \in z$, um processo estocástico, tal que :

$$X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \varepsilon_t.$$

X_t representa processo autoregressivo de ordem p, abreviadamente AR(p) se:

- existirem números reais $\phi_1, \phi_2, \ldots, \phi_p$,
- $\phi_p \neq 0$,
- existir uma constante $c \in \mathbb{R}$,
- existir um processo de ruído branco $\{\varepsilon_t\}_{t\in\mathbb{Z}}$. Ou seja, um processo de média nula, variância $\sigma_\varepsilon^2 > 0$, constante, e não correlacionado.

A título de exemplo, consideremos o processo autorregressivo mais simples, de ordem 1, $AR(1)$ definido por:

$$X_t = c + \phi X_{t-1} + \varepsilon_t, \tag{2.1}$$

O processo $AR(1)$ é estacionário se, e somente se, $|\phi| < 1$. Para $|\phi| \geq 1$, o processo não é estacionário: no caso $\phi = 1$ resulta num passeio aleatório, e para $|\phi| > 1$ a variância diverge.

Sob a condição de estacionaridade $|\phi| < 1$, prova-se facilmente que:

$$E(X_t) = \frac{c}{1-\phi}, \tag{2.2}$$

$$\mathrm{Var}(X_t) = \frac{\sigma_\varepsilon^2}{1-\phi^2}. \tag{2.3}$$

De forma geral, a autocovariância no atraso k é:

$$\gamma_k = \mathrm{Cov}(X_t, X_{t-k}) = \frac{\sigma_\varepsilon^2}{1-\phi^2}\,\phi^k, \qquad k \geq 0. \tag{2.4}$$

Assim, e dado $\gamma_0 = \mathrm{Var}(X_t)$, a função de autocorrelação (FAC)[2] de um AR(1) toma a forma:

$$\rho(k) = \frac{\gamma_k}{\gamma_0} = \phi^k, \quad k \geq 0. \tag{2.5}$$

A sucessão das FAC de um processo AR(1) estacionário tende exponencialmente, e/ou de modo sinusoidal, para zero à medida que o valor de k aumenta.

$$\lim_{k\to\infty} \rho(k) \;=\; \lim_{k\to\infty} \phi^k \;=\; 0, \quad \text{para } |\phi| < 1.$$

[2]Em inglês, função de autocorrelação é denominada *autocorrelation function* (ACF).

Prova-se, recorrendo à regra de Crámer, que a sucessão de funções de autocorrelação parcial de um processo AR(1) estacionário anula-se a partir da ordem 2 (decaimento brusco), ver Cryer (1986) e Cryer and Chan (2008):

$$\begin{cases} \phi_{1,1} = \phi, \\ \phi_{k,k} = 0, \quad k \geq 2. \end{cases}$$

Na Figura 2.3 ilustra-se um exemplo do comportamento das ACF e PACF[3] de um AR(1) simulado.

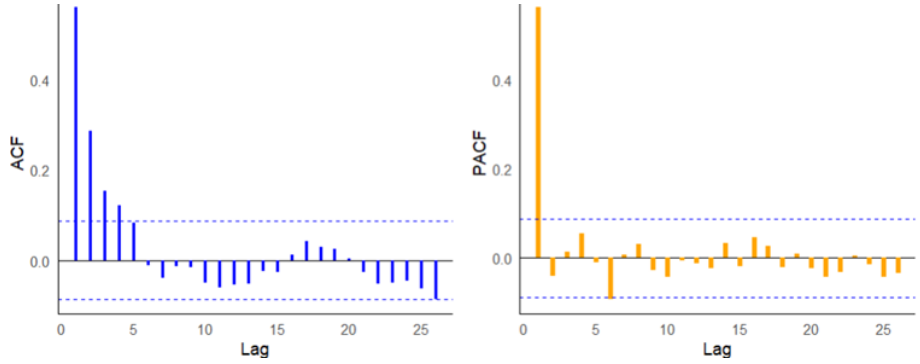


Figura 2.3: ACF e PACF de um processo AR(1): $X_t = 3 + 0.6X_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim N(0, 4)$

Conforme demostrado em (Montgomery et al., 2008, pp. 246–250) a FAC, $\rho(k)$, do processo geral, de ordem p, AR(p), satisfaz as equações de Yule-Walker:

$$\rho(k) = \phi_1\rho(k-1) + \phi_2\rho(k-2) + \cdots + \phi_p\rho(k-p), \quad k \geq 1.$$

Esta é uma equação de diferenças lineares de ordem p. A solução geral depende das raízes z_i do polinómio autoregressivo $\Phi(z)$:

$$\Phi(z) = 1 - \phi_1 z - \phi_2 z^2 - \cdots - \phi_p z^p.$$

Se o polinómio autoregressivo não possuir raízes no círculo unitário então o processo é estacionário pelo que $\rho(k)$ decai exponencialmente e/ou de forma sinusoidal para zero à medida que $k \to \infty$.

A PACF é obtida resolvendo o sistema de Yule-Walker de ordem k:

$$\begin{bmatrix} 1 & \rho(1) & \dots & \rho(k-1) \\ \rho(1) & 1 & \dots & \rho(k-2) \\ \vdots & \vdots & \ddots & \vdots \\ \rho(k-1) & \rho(k-2) & \dots & 1 \end{bmatrix} \begin{bmatrix} \phi_{k1} \\ \phi_{k2} \\ \vdots \\ \phi_{kk} \end{bmatrix} = \begin{bmatrix} \rho(1) \\ \rho(2) \\ \vdots \\ \rho(k) \end{bmatrix}$$

e pode ser igualmente usada para identificar a ordem de um processo AR(p) dado que para $k > p$, os coeficientes $\phi_{kk} = 0$, porque os termos adicionais X_{t-k} já estão

[3]Em inglês, função de autocorrelação Parcial é denominada *Partial autocorrelation function* (PACF).

completamente explicados pelos p primeiros lags. A sucessão de funções de autocorrelação parcial de um processo AR(p) estacionário anulam-se a partir da ordem $p+1$.

Note ainda que o processo AR(p) estacionário é sempre invertível porque pode ser escrito como um processo MA(q) de ordem infinita. Ou seja, apenas à custa dos erros, sempre que este processo seja estacionário ele é causal, uma propriedade fundamental para se poder fazer previsões. Pois permite prever valores futuros através de valores passados.

Apesar de não trabalharmos com os modelos de médias móveis (MA), iremos de seguida apresentá-los, de modo bastante abreviado.

MA(q)

Um processo $\{X_t\}_{t\in Z}$, admite uma representação de médias móveis de ordem q, abreviadamente MA(q), se verifica a seguinte equação estocástica:

$$X_t = c + \varepsilon_t - \theta_1\varepsilon_{t-1} - \theta_2\varepsilon_{t-2} - \cdots - \theta_q\varepsilon_{t-q},$$

onde $(\theta_1, \theta_2, \ldots, \theta_q)$, $\theta_q \neq 0$, são igualmente números reais ε_t é um ruído branco. Por ser uma combinação linear finita de processos de ruído branco, X_t admite sempre estacionariedade fraca.
Prova-se em que MA(q) é invertível caso o polinómio de médias móveis de ordem q não possua raízes no círculo unitário, ver Hamilton (2020).

Considere novamente o caso mais simples, o modelo MA(1);

$$X_t = c + \epsilon_t - \theta\epsilon_{t-1},$$

onde a função de autocovariância do modelo é dada por:

$$\begin{aligned}\gamma_x(0) &= \sigma^2(1+\theta^2)\\ \gamma_x(1) &= -\theta\sigma^2\\ \gamma_x(k) &= 0, \quad k > 1\end{aligned}$$

Similarmente, temos a função de autocorrelação como

$$\rho_x(1) = \frac{-\theta}{1+\theta^2} \tag{2.6}$$

$$\rho_x(k) = 0, \quad k > 1 \tag{2.7}$$

A partir da equação 2.6 , podemos ver que a primeira autocorrelação de defasagem em MA(1) é limitada como

$$|\rho_y(1)| = \frac{|\theta|}{1+\theta^2} \leq \frac{1}{2}$$

e a função de autocorrelação corta após a defasagem 1. O proceso MA(1) pode ser representado como um processo AR(∞), ou seja:

$$X_t = -\theta X_{t-1} + \theta^2 X_{t-2} - \theta^3 X_{t-3} + \cdots + \varepsilon_t.$$

Essa expansão só converge se $|\theta| < 1$. Assim:

$$\begin{cases} |\theta| < 1 & \implies \text{processo MA(1) invertível,}\\ |\theta| \geq 1 & \implies \text{processo MA(1) não invertível.}\end{cases}$$

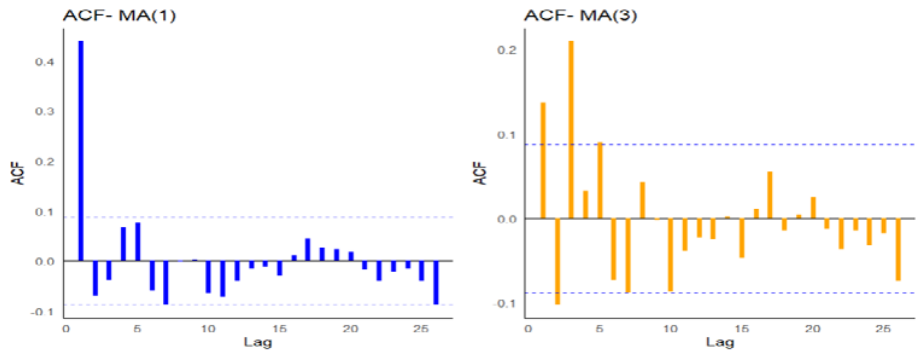


Figura 2.4: ACF dos modelos MA(1): $X_t = 3+\varepsilon_t+0.7\,\varepsilon_{t-1}, \quad \varepsilon_t \sim N(0,4)$ e MA(3): $X_t = 5+\varepsilon_t+0.5\,\varepsilon_{t-1}-0.3\,\varepsilon_{t-2}+0.4\,\varepsilon_{t-3}, \quad \varepsilon_t \sim N(0,4)$

Para ilustrar o comportamento da função de autocorrelação, considere os processos M(1) e MA(3) da Figura 2.4.

De igual modo, iremos introduzir o caso geral do modelo do tipo autoregressivo de médias móveis de ordem, compostos, como o nome indica, por ambas as componentes, autoregressivas e de médias móveis.

ARMA(p,q)

Se juntarmos os dois processos AR(p) e MA(q), obtemos o processo autoregressivo de médias móveis de ordem (p,q), abreviadamente ARMA(p,q), dado pela equação :

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{i=0}^{q} \theta_i \varepsilon_{t-i}$$

ou

$$\Phi(B)X_t = c + \Theta(B)\varepsilon_t$$

onde

$$\Phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p,$$
$$\Theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q$$

são, respetivamente, os polinómios autoregressivos e de médias móveis.

Caso estes polinómios $\Phi(B)$ e $\Theta(B)$ não tenham raízes comuns, o processo admite uma solução estacionária e única e é invertível se estes não possuírem raízes no círculo unitário.

A FAC de um modelo ARMA(p,q) satisfaz a seguinte equação:

$$\rho(k) - \phi_1\rho(k-1) - \cdots - \phi_p\rho(k-p) = 0, \quad k \geq q+1$$

1. Para um processo ARMA(p,q) estacionário, as funções de autocorrelação $\rho(k)$ tendem exponencialmente e/ou de forma sinusoidal para zero à medida que k aumenta (Hamilton, 2020, p. 255-256).

2. De forma semelhante, as funções de autocorrelação parcial de um processo ARMA(p,q) estacionário também tendem exponencialmente e/ou de forma sinusoidal para zero à medida que k aumenta (Hamilton, 2020, p. 255-256).

Este comportamento pode ser visto na Figura 2.5, onde apresenta-se dois exemplos simulados de séries do tipo ARMA(1,1) e ARMA(2,2), com as respetivas funções de autocorrelação e autocorrelação parcial.

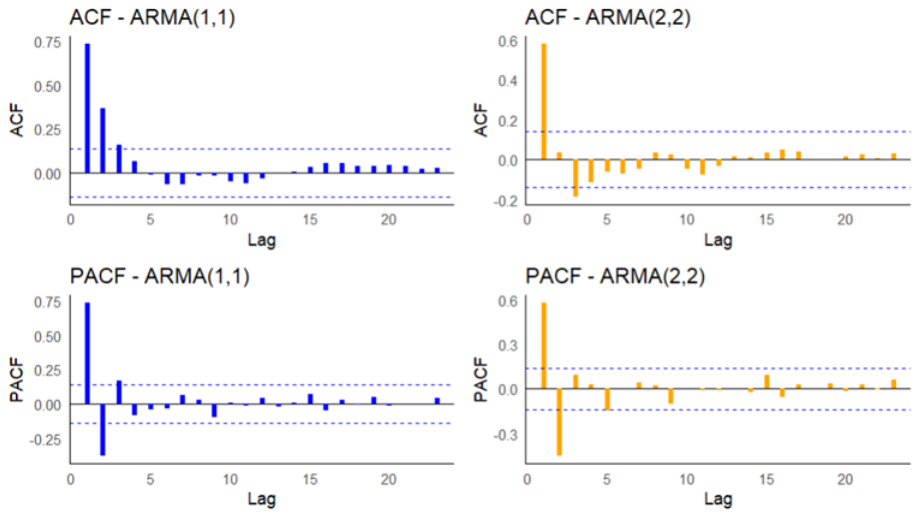


Figura 2.5: ACF e PACF de processos ARMA(1,1):$X_t = 4 + 0.6X_{t-1} + 0.5\varepsilon_{t-1} + +\varepsilon_t, \quad \varepsilon_t \sim N(0,4)$ e ARMA(2,2):$X_t = 2 + 0.5X_{t-1} - 0.3X_{t-2} + 0.4\varepsilon_{t-1} + 0.2\varepsilon_{t-2} + \varepsilon_t, \quad \varepsilon_t \sim N(0,4)$

ARIMA(p,d,q)

Vimos que, para séries não estacionárias — na maioria dos casos, séries que possuem tendência e/ou variância não constante —, pode-se obter a estacionariedade aplicando-se transformações às séries, como a transformação de Box-Cox e/ou o operador da diferenciação simples da série (um número d de vezes necessário até obter a estacionariedade), ou seja,

$$\nabla^d X_t = (1-B)^d X_t.$$

Feito isto, passamos a obter um processo autorregressivo integrado de média móvel (ARIMA) de ordens p, d, q.

Um ARIMA(p,d,q) admite, assim, a seguinte representação:

$$\Phi(B)(1-B)^d X_t = c + \Theta(B)\varepsilon_t.$$

O gráfico da Figura 2.6 mostra o processo $(1-0.7B)(1-B)X_t = \varepsilon_t + 0.5\varepsilon_{t-1}$. Observa-se que a FAC apresenta decaimento lento devido à diferenciação, enquanto a PACF corta após o lag $p+1$.

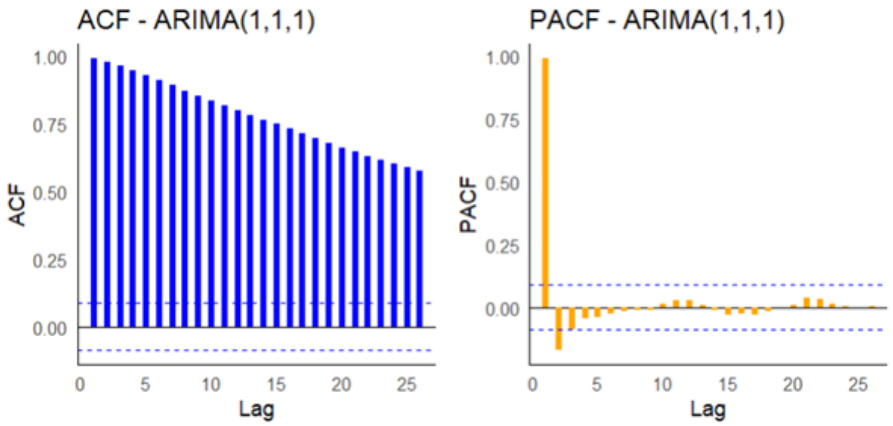


Figura 2.6: ACF e PACF de processos ARMA(1,1,1)

2.2 Séries temporais em extremos

A análise estatística de extremos em séries temporais é uma componente tradicional na Hidrologia e no setor de Seguros, com aplicações cada vez mais fortes em Finanças. A dependência temporal é comum em extremos univariados, podendo ser originada por autocorrelação, efeitos de outras variáveis ou por ambos, o que exige um tratamento teórico adequado.

A dependência de curto alcance, que resulta em aglomerados de extremos, é frequentemente observada em séries financeiras devido à aglomeração de volatilidade. Da mesma forma, máximos de fluxos de rios geralmente ocorrem logo após uma tempestade. Variações em grande escala, decorrentes de tendências, sazonalidades ou mudanças de regimes são tipicamente abordadas com modelos adequados, com intuito de estudar o impacto da autocorrelação sob condições de mistura que limitam a influência da dependência nos extremos (Bücher and Zhou (2021)).

2.2.1 Distribuições exata e assintótica do máximo

Considere uma amostra aleatória $X_1, X_2, \ldots, X_n$ de variáveis aleatórias independentes e identicamente distribuídas com função de distribuição acumulada (CDF) $F(x)$. Seja $M_n = \max(X_1, \ldots, X_n)$, a variável aleatória que representa o máximo da coleção de variáveis aleatórias $\{X_i\}_{i=1}^n$. A CDF de M_n é dada por

$$F_{M_n}(x) = P(M_n \leq x) = P(X_1 \leq x, \ldots, X_n \leq x) = \prod_{i=1}^{n} P(X_i \leq x) = [F(x)]^n.$$

De forma análoga, considerando, $m_n = \min(X_1, \ldots, X_n)$, a variável aleatória que representa o mínimo da coleção de variáveis aleatórias $\{X_i\}_{i=1}^n$, a CDF de m_n é

$$F_{m_n}(x) = P(m_n \leq x) = 1 - P(X_1 > x, \ldots, X_n > x) = 1 - [1 - F(x)]^n.$$

Note que $m_n = -\max(-X_1, \ldots, -X_n)$, assim o problema do mínimo é *dual* ao do máximo e portanto, qualquer resultado para o mínimo pode ser traduzido para o máximo aplicando a transformação $X_i \mapsto -X_i$. Por esta razão, neste trabalho analisaremos apenas mudanças de estrutura influenciadas por valores máximos.

O nosso interesse estará no comportamento de M_n, para n muito grande, ou seja, na distribuição de M_n quando $n \to \infty$,

$$\lim_{n\to\infty} P(M_n \leq x) = \lim_{n\to\infty} [F(x)]^n$$

$$\begin{cases} \text{Se } 0 \leq F(x) < 1, \text{ então } \lim_{n\to\infty} [F(x)]^n = 0. \\ \text{Se } F(x) = 1, \text{ então } \lim_{n\to\infty} [F(x)]^n = 1. \end{cases}$$

Portanto, a distribuição limite de M_n é uma distribuição degenerada, que assume o valor 0 quando $0 \leq F(x) < 1$ e 1 quando $F(x) = 1$, o que não é muito útil para a análise de valores extremos. De forma a obter uma distribuição limite não degenerada para M_n é necessário considerar sequências de constantes de normalização $a_n > 0$ e $b_n \in \mathbb{R}$, à semelhança do que ocorre no teorema do limite central,

$$\lim_{n\to\infty} P\left(\frac{M_n - b_n}{a_n} \leq x\right) = \lim_{n\to\infty} P\left(M_n \leq b_n + a_n x\right) = \lim_{n\to\infty} \left[F(b_n + a_n x)\right]^n.$$

Este limite é descrito pelo Teorema de *Tipos Extremais*, que fornece a distribuição limite do máximo. Assim, se existem sucessões de constantes normalizadoras $a_n > 0$ e $b_n \in R$ tais que:

$$[F(b_n + a_n x)]^n \xrightarrow{d} G(x), \quad \text{quando} \quad n \to \infty,$$

para alguma CDF não degenerada $G(x)$, e onde $\xrightarrow{d}$, representa a convergência em distribuição, então $G(x)$ só pode ser do mesmo tipo que uma das seguintes distribuições:

$$G(x) := \begin{cases} \exp\left\{-e^{-x}\right\}, x \in \mathbb{R} & \text{(Gumbel)} \\ \begin{cases} 0, & x \leq 0, \\ \exp\left\{-x^{-\alpha}\right\}, & x > 0, \end{cases} & \text{(Fréchet, } \alpha > 0) \\ \begin{cases} \exp\left\{-(-x)^{\alpha}\right\}, & x \leq 0, \\ 1, & x > 0, \end{cases} & \text{(máx-Weibull, } \alpha > 0). \end{cases}$$

Estas três distribuições limite foram unificadas por Von Mises (1936) e Jenkinson (1955) numa única distribuição, designada de distribuição *Generalizada de Valores Extremos* (GEV).

Uma variável aleatória X segue a distribuição GEV, e escreve-se $X \sim \text{GEV}(\mu, \sigma, \xi)$, onde $\xi \in \mathbb{R}$ é o parâmetro de forma, $\mu \in \mathbb{R}$ o parâmetro de localização e $\sigma \in \mathbb{R}^+$ o parâmetro de escala, se a sua CDF for dada por:

$$F_{\text{GEV}}(x) := \begin{cases} \exp\left[-\left(1 + \xi \cdot \frac{x-\mu}{\sigma}\right)^{-\frac{1}{\xi}}\right], & \text{para} \quad 1 + \xi \cdot \frac{x-\mu}{\sigma} > 0 \\ \exp\left\{-\exp\left[-\frac{x-\mu}{\sigma}\right]\right\}, & \quad x \in \mathbb{R} \end{cases} \tag{2.8}$$

O parâmetro ξ é conhecido como **índice de valores extremos** (ou *extreme value index* (EVI)) e desempenha um papel fundamental, pois controla o comportamento da cauda da distribuição GEV. As distribuições limite para máximos, *Gumbel*, *Fréchet* e *máx-Weibull*, podem ser obtidas como casos particulares da GEV, como ilustrado abaixo:

(i) Se $\xi < 0$, a CDF de X, em (2.8), pode ser escrita como;

$$\begin{aligned} F_{\text{GEV}}(x) &= \exp\left\{-\left(1-\frac{\xi\mu}{\sigma}+\frac{\xi x}{\sigma}\right)^{-\frac{1}{\xi}}\right\} \\ &= \exp\left\{-\left(-\left(\frac{\xi\mu}{\sigma}-1+\frac{-\xi}{\sigma}\cdot x\right)\right)^{-\frac{1}{\xi}}\right\}. \end{aligned}$$

Definindo convenientemente $b = \frac{\xi\mu}{\sigma} - 1$, e $a = \frac{-\xi}{\sigma}$, obtemos

$$F_{\text{GEV}}(x) = \exp\left\{-\left(-(b+a\cdot x)\right)^{-\frac{1}{\xi}}\right\}.$$

Como $\xi < 0$, temos que $\alpha := -\frac{1}{\xi} > 0$, e portanto

$$F_{\text{GEV}}(x) = \exp\left\{-\left(-(ax+b)\right)^{\alpha}\right\} = \exp\left\{-(-y)^{\alpha}\right\},$$

Com $y = ax + b$, temos que

$$F_{\text{GEV}}(x) = \exp\left\{-(-y)^{\alpha}\right\},$$

y<0(2.9) correspondendo à CDF da distribuição *máx-Weibull*, que é um caso particular da distribuição GEV para $\xi < 0$. As distribuições dizem-se de cauda curta ou leve e possuem limite superior do suporte finito.

(ii) Para $\xi > 0$,

$$F_{\text{GEV}}(x) = \exp\left\{-\left(1-\frac{\xi\mu}{\sigma}+\frac{\xi x}{\sigma}\right)^{-\frac{1}{\xi}}\right\}.$$

Definindo,

$$b = 1-\frac{\xi\mu}{\sigma}, \quad a = \frac{\xi}{\sigma},$$

a função $G(x)$ é reescrita por

$$F_{\text{GEV}}(x) = \exp\left\{-\left(ax+b\right)^{-\frac{1}{\xi}}\right\},$$

ou seja

$$F_{\text{GEV}}(y) = \exp\left\{-y^{-\alpha}\right\}, \quad y > 0, \tag{2.10}$$

onde $\alpha = \frac{1}{\xi}$ e $y = ax+b$, correspondendo à CDF da distribuição *Fréchet*, que é um caso particular da distribuição GEV para $\xi > 0$. As distribuições dizem-se de cauda pesada ou com cauda de tipo Pareto e possuem limite superior do suporte infinito.

(iii) Para $\xi = 0$, a CDF da distribuição *Gumbel* é obtida como sendo o limite, quando $\xi \to 0$, da CDF da distribuição GEV,

$$F_{\text{GEV}}(x) = \exp\left\{-\exp\left[-\frac{x-\mu}{\sigma}\right]\right\}, \quad x \in \mathbb{R}$$

e considerando $y = \frac{x-\mu}{\sigma}$, temos

$$F_{\text{GEV}}(x) = \exp\left\{-\exp\left(-y\right)\right\}, \quad y \in \mathbb{R}, \tag{2.11}$$

correspondendo à CDF da distribuição *Gumbel*, que é um caso particular da distribuição GEV para $\xi = 0$.

O interesse em optar por uma distribuição específica ou por uma distribuição unificada está frequentemente relacionado com as características do fenómeno em análise. Associado ao índice de valores extremos, ξ, existem outros parâmetros, usualmente designados de parâmetros de acontecimentos extremos que permitem descrever de forma mais precisa a frequência e a magnitude dos eventos raros, como o quantil extremal e o nível de retorno a T anos.

Quantil da distribuição GEV: O quantil x_p é o valor tal que a probabilidade acumulada $\mathrm{P}(X \leq x_p) = F_{\mathrm{GEV}}(x_p) = p$. Para determinar a função quantil, basta resolver a equação abaixo em ordem a x_p:

$$\exp\left\{-\left(1+\frac{\xi(x_p-\mu)}{\sigma}\right)^{-\frac{1}{\xi}}\right\} = p \iff 1+\frac{\xi(x_p-\mu)}{\sigma} = (-\log p)^{-\xi}$$

obtendo-se o quantil

$$x_p = \mu + \sigma\left[(-\log p)^{-\xi} - 1\right]/\xi.$$

Para encontrar a *mediana* $x_{1/2}$, substituímos $p = 1/2$ na fórmula geral do quantil:

$$x_{1/2} = \mu + \sigma\left[(\log 2)^{-\xi} - 1\right]/\xi.$$

Os quantis extremais caracterizam-se por terem uma probabilidade de excedência muito pequena, que é, em geral, da ordem de 0.01 (quantil $x_{0.99}$), 0.001 (quantil $x_{0.999}$) e 0.0001 (quantil $x_{0.9999}$).

Nível de retorno da distribuição GEV: Um *nível de retorno* T (unidades de tempo) é o nível que se espera ser ultrapassado, em média, uma vez a cada T. Seja x_T o nível de retorno com período T. Então, temos:

$$P(X > x_T) = \frac{1}{T} \iff P(X \leq x_T) = 1 - \frac{1}{T}$$

Como $F_{\mathrm{GEV}}(x_T) = P(X \leq x_T)$, segue que:

$$F_{\mathrm{GEV}}(x_T) = 1 - \frac{1}{T} \iff \exp\left[-\left(1+\frac{\xi(x_T-\mu)}{\sigma}\right)^{-\frac{1}{\xi}}\right] = 1 - \frac{1}{T}$$

$$x_T = \mu + \frac{\sigma}{\xi}\left[(-\log(1-\frac{1}{T}))^{-\xi} - 1\right]. \tag{2.13}$$

2.2.2 Método dos momentos ponderados de probabilidade e extensões

No âmbito da estatística clássica existem diversos métodos usados para estimar parâmetros estatísticos de interesse, entre os quais se destacam o método dos momentos, o método da máxima verosimilhança (MV) e ainda métodos derivados ou inspirados nestes. O método da MV é amplamente utilizado pela sua eficiência assintótica e pelas suas boas propriedades estatísticas, mas pode revelar-se exigente do ponto de vista computacional, sobretudo quando aplicado a distribuições mais complexas ou em situações em que a função de verosimilhança é de difícil manipulação. Neste trabalho, optamos por detalhar o método dos Momentos Ponderados de Probabilidade

(MPP) (ou PWM, do inglês *Probability Weighted Moments*), uma generalização do método dos momentos, que apresenta vantagens práticas em termos de simplicidade e robustez, sendo também o método utilizado na deteção de mudança no parâmetro de forma ξ, de particular interesse para este trabalho.

Os PWM de ordem (p, r, s), para uma variável aleatória X com CDF F foram introduzidos por Greenwood et al. (1979) e são definidos por:

$$M_{p,r,s} = E\{X^p[F(X)]^r[1-F(X)]^s\}, \quad p, r, s \in \mathbb{R}. \tag{2.14}$$

O caso específico da estimação PWM dos parâmetros da distribuição GEV foi desenvolvido por Hosking et al. (1985). Para o caso em que $\xi \neq 0$, em particular com $\xi < 1$, tem-se para $p = 1$, $r = 0, 1, 2, \ldots$ e $s = 0$ que o PWM de ordem (1,r,0), $\mathrm{M}_{1,r,0}$, assume a forma:

$$M_{1,r,0} = E\{X[F(X)]^r\} = \frac{1}{r+1}\left\{\mu - \frac{\sigma}{\xi}[1-(r+1)^{\xi}\Gamma(1-\xi)]\right\},$$

onde $\Gamma(\cdot)$ é a função gama dada por $\Gamma(t) = \int_0^{+\infty} x^{t-1}e^{-x}dx$, $t \geq 0$. Os estimadores PWM $(\hat{\xi}, \hat{\mu}, \hat{\sigma})$ para (ξ, μ, σ) são a solução do sistema de equações:

$$\begin{cases} M_{1,0,0} = \mu - \dfrac{\sigma}{\xi}[1-\Gamma(1-\xi)] \\ 2M_{1,1,0} - M_{1,0,0} = \dfrac{\sigma}{\xi}\Gamma(1-\xi)(2^{\xi}-1) \\ \dfrac{3M_{1,2,0} - M_{1,0,0}}{2M_{1,1,0} - M_{1,0,0}} = \dfrac{3^{\xi}-1}{2^{\xi}-1}. \end{cases}$$

Substituindo em seguida $M_{1,r,0}$, com $r = 0, 1, 2$, pelo estimador centrado dado por Landwehr et al. (1979):

$$\hat{M}_{1,r,0} = \frac{1}{n}\sum_{j=1}^{n}\left(\prod_{l=1}^{r}\frac{(j-l)}{(n-l)}\right)\cdot X_{j,n}, \tag{2.15}$$

onde $(X_{1,n}, \ldots, X_{n,n})$ representam as estatísticas de ordem de uma amostra proveniente de uma distribuição com CDF F. Reescrevendo as duas primeiras equações, em ordem a μ e a σ, respetivamente, obtém-se os estimadores PWM $(\hat{\xi}, \hat{\mu}, \hat{\sigma})$:

$$\hat{\mu} = \hat{M}_{1,0,0} + \frac{\hat{\sigma}}{\hat{\xi}}[1-\Gamma(1-\hat{\xi})]$$

$$\hat{\sigma} = \frac{\hat{\xi}\,(2\hat{M}_{1,1,0} - \hat{M}_{1,0,0})}{\Gamma(1-\hat{\xi})(2^{\hat{\xi}}-1)}$$

$$\frac{3\hat{M}_{1,2,0} - \hat{M}_{1,0,0}}{2\hat{M}_{1,1,0} - \hat{M}_{1,0,0}} = \frac{3^{\hat{\xi}}-1}{2^{\hat{\xi}}-1}.$$

Note-se que, para se obter uma estimativa de $\hat{\xi}$ terá de se recorrer a métodos numéricos. Já na situação em que $\xi = 0$, ou seja, o caso da distribuição Gumbel, tem-se para $p = 1$, $r = 0, 1, 2, \ldots$ e $s = 0$, que o PWM de ordem (1,r,0) é dado por

$$M_{1,r,0} = E\{X[F(X)]^r\} = \frac{1}{r+1}\left[\mu + \sigma(-\psi(1) + \log(1+r))\right], \tag{2.16}$$

em que, $\psi(x) = \frac{d}{dx}\log\Gamma(x)$ é a função digama e $-\psi(1) \simeq 0.5772156649$ é a constante de Euler-Mascheroni. Neste caso, apenas os parâmetros de localização e escala (μ, σ) precisam de ser estimados. Assim, substituindo em (2.16) a variável r por 0 e 1 e, resolvendo as equações obtidas em ordem aos dois parâmetros, obtém-se

$$\begin{cases} \mu = M_{1,0,0} + \sigma\psi(1) \\ \sigma = \frac{2M_{1,1,0} - M_{1,0,0}}{\log 2}. \end{cases}$$

Novamente usando o estimador para $M_{1,r,0}$, apresentado em (2.15), e substituindo em (2.16), obtém-se os estimadores PWM para a localização e escala do modelo GEV, com $\xi = 0$:

$$\hat{\mu} = \hat{M}_{1,0,0} + \sigma\psi(1)$$

$$\hat{\sigma} = \frac{2\hat{M}_{1,1,0} - \hat{M}_{1,0,0}}{\log 2}.$$

Para estes estimadores baseados nos PWM, para $\xi < 1$ e quando a dimensão da amostra de máximos em estudo, $n \to \infty$, verifica-se que

$$\sqrt{n}\big((\hat{\xi}, \hat{\mu}, \hat{\sigma}) - (\xi, \mu, \sigma)\big)$$

é assintoticamente Normal. Mais detalhes poderão ser vistos em Beirlant et al. (2006).

Em cenários onde se exige mais flexibilidade e robustez para estimar os parâmetros da distribuição GEV, é comum a utilização dos método dos Momentos Ponderados de Probabilidade Generalizados (GPWM). É uma extensão do método clássico PWM que se baseia no cálculo de momentos da forma:

$$\nu_\omega = E[X\omega(F_{\text{GEV}})] = \int_{-\infty}^{\infty} x\,\omega(F_{\text{GEV}}(x))\; dF_{\text{GEV}}(x), \tag{2.17}$$

onde ω é uma função contínua adequada. Fazendo uma mudança de variáveis, este momento pode ser reescrito como

$$\nu_\omega = \int_0^1 F_{\text{GEV}}^{-1}(u)\,\omega(u)\,du.$$

Definindo a função $W(t) = \int_0^t \omega(u)\,du$, com $W(0) = 0$, um estimador natural para ν_ω é dado por

$$\widehat{\nu}_{\omega,n} = \int_0^1 \mathbb{F}_n^{-1}(u)\,\omega(u)\,du,$$

onde $\mathbb{F}_n$ denota a função de distribuição empírica baseada na amostra $(X_1, \ldots, X_n)$. As propriedades assintóticas de $\widehat{\nu}_{\omega,n}$ para a distribuição GEV podem ser consultadas em Diebolt et al. (2007).

2.2.3 Modelos para séries temporais em extremos

A análise de valores extremos em séries temporais constitui uma área em crescimento, motivada pela necessidade de compreender fenómenos raros e de grande impacto, como tempestades, vagas de calor ou flutuações financeiras intensas. Ao

contrário do caso independente, os extremos em processos temporais revelam frequentemente dependência serial e agrupamento de excedências, o que torna necessária a adaptação das metodologias clássicas da Teoria dos Valores Extremos.

Entre as abordagens disponíveis, continuam a ser fundamentais os métodos que recorrem a máximos de blocos e às excedências acima de limiares elevados. No entanto, a sua utilização em séries temporais exige que se verifique a chamada condição de mistura (Leadbetter et al. (1983)), que garante a validade dos resultados assintóticos mesmo na presença de dependência.

A presença de agrupamentos ou *clusters* de excedências pode ser quantificada através do *índice extremal* $\theta \in (0, 1]$:

$$\theta = \lim_{n\to\infty} \frac{\mathbb{P}(M_n \leq u_n)}{[\mathbb{P}(X_1 \leq u_n)]^n}.$$

O índice extremal pode ser visto como o recíproco do limite do tamanho médio dos grupos de excedências e por isso, quando $\theta < 1$ há evidência de *clusters* de extremos, e o caso $\theta = 1$ corresponde a um cenário de sucessões i.i.d.

Na área financeira, modelos GARCH são aplicados com frequência por conseguirem capturar a heteroscedasticidade e períodos de forte volatilidade associados a valores extremos (Embrechts et al. (1997)). Versões destes modelos têm sido adaptadas para séries de contagem no contexto do estudo de sequências periódicas com dependência extremal Scotto et al. (2015).

Abordagens computacionais e utilização de técnicas de reamostragem como o *bootstrap* foram também utilizadas para aumentar a precisão das previsões e a robustez dos intervalos de previsão em séries com caudas pesadas (Cordeiro and Neves (2014, 2019)).

A metodologia proposta em Kojadinovic and Naveau (2017), para deteção de mudanças nos parâmetros (μ, σ, ξ) da distribuição GEV foi desenvolvida no contexto de séries i.i.d., sendo ainda válida em condições de dependência fraca. Neste trabalho propomos avaliar, via simulação de Monte Carlo, a robustez e o desempenho da metodologia na presença de níveis de dependência mais acentuados.

2.3 Tipo de mudanças de estruturas

O estudo das mudanças de estrutura em séries temporais assume uma importância central em diversas áreas do conhecimento. A identificação e compreensão dessas mudanças são fundamentais para interpretar fenómenos complexos e dar resposta a problemas com especial relevância na atualidade. Em numerosos contextos práticos, torna-se necessário recorrer a metodologias estatísticas robustas que permitam detetar, localizar e analisar alterações no comportamento estrutural das séries. Estas técnicas são particularmente relevantes quando ocorrem mudanças nas propriedades estatísticas da série, como mudanças na média, na variância, na tendência ou noutros parâmetros que indiquem transições significativas.

Considera-se que uma série temporal apresenta mudanças de estrutura quando há uma alteração nas suas propriedades ao longo do tempo. A deteção de mudanças na média da série foi inicialmente proposta por Page (1954, 1955), tendo sido posteriormente aprofundada pelos trabalhos de Hinkley (1970), Bai (1997) e Killick et al. (2010). No campo da econometria, o foco do estudo das mudanças na tendência

da série está na análise da instabilidade dos coeficientes nos modelos de regressão — veja-se, por exemplo, Andrews (1993) Bai and Perron (2003b) e Zeileis et al. (2002a).

Outros tipos de mudanças, como mudanças na variância, têm sido explorados, conforme demonstrado em Killick and Eckley (2014), Chen and Gupta (1997) e Horváth (1993). Adicionalmente, modificações na estrutura de dependência (ou correlação) também têm sido alvo de investigação (Hamilton (2020) e Barry and Hartigan (1993), entre outros).

Até à data, já se encontram disponíveis diversos resultados teóricos — e alguns com aplicação prática — para lidar com os variados tipos de mudanças de estrutura que os dados podem apresentar, com o objetivo de possibilitar inferências mais fiáveis (veja-se, por exemplo, Shao and Zhang (2010),Kojadinovic and Naveau (2017) e Casini and Perron (2018)).

Neste trabalho, em particular, propomos-nos estudar as mudanças na média da série, na tendência e mudanças na forma, mais concretamente, no comportamento da cauda da distribuição generalizada de valores extremos.

2.3.1 Mudanças na média

Considere-se uma série temporal univariada $\{X_t,\, t = 1, \ldots, n\}$ com variância constante.

Vamos supor, por simplicidade, que se pretende testar a existência de uma única mudança de média num ponto temporal desconhecido k, tal que $k \in [1, n-1[$.

Então, o que se pretende testar são as seguintes hipóteses

$$\begin{cases} H_0: & E[X_1] = E[X_2] = \cdots = E[X_n] = \mu \\ H_1: & E[X_1] = E[X_2] = \cdots = E[X_k] \neq E[X_{k+1}] = \cdots = E[X_n] \end{cases}$$

Ou seja, não existe mudança de média *versus* existe mudança de média no ponto $t = k$.

Na Figura 2.7, apresentamos uma série temporal com uma mudança de média no ponto $k = 100$.

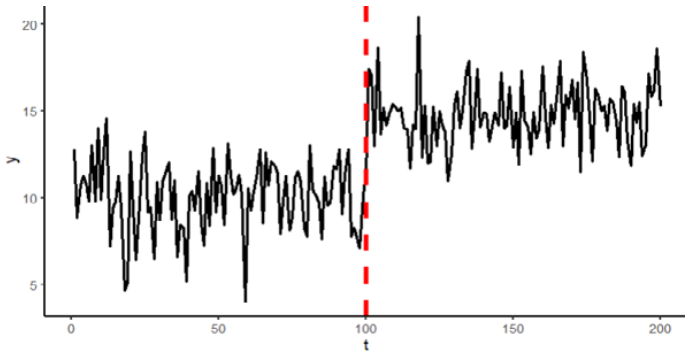


Figura 2.7: Uma mudança de nível

2.3.2 Mudanças na tendência

Considere-se uma série temporal univariada $\{X_t,\, t = 1, \ldots, n\}$ com tendência, que pode ser descrita, numa forma simplificada, pelo modelo de regressão linear:

$$X_t = \beta_0 + \beta_1\, t + \varepsilon_t, \quad t = 1, \ldots, n, \tag{2.18}$$

onde β_0 representa a ordenada na origem, β_1 o coeficiente de tendência e $\varepsilon_t \sim \mathrm{WN}(0, \sigma_\varepsilon^2)$.

Admitindo a existência de um ponto de mudança $k \in \{1, \ldots, n-1\}$, podem ser consideradas duas situações distintas.

No primeiro caso, assume-se que a série temporal é contínua no ponto de mudança, ocorrendo apenas uma alteração no declive após $t = k$. O modelo pode ser escrito como:

$$X_t = \begin{cases} \beta_0 + \beta_1\, t + \varepsilon_t, & t \leq k, \\ \beta_0 + \beta_2\, t + \varepsilon_t, & t > k, \end{cases} \tag{2.19}$$

garantindo continuidade em $t = k$.

As hipóteses a testar são então:

$$\begin{cases} H_0: & \beta_1 = \beta_2, \\ H_1: & \beta_1 \neq \beta_2, \end{cases}$$

ou seja, a ausência de mudança na tendência *versus* a existência de uma mudança no declive no ponto $t = k$.

Pode ser de igual modo relevante estudar casos em que ocorrem uma alteração conjunta da ordenada na origem e da inclinação da tendência. Neste caso, o modelo é dado por:

$$X_t = \begin{cases} \beta_{01} + \beta_1\, t + \varepsilon_t, & t \leq k, \\ \beta_{02} + \beta_2\, t + \varepsilon_t, & t > k, \end{cases} \quad k = 1, \ldots, n-1. \tag{2.20}$$

As hipóteses de interesse passam a ser:

$$\begin{cases} H_0: & \beta_{01} = \beta_{02} \text{ e } \beta_1 = \beta_2, \\ H_1: & \text{pelo menos um dos parâmetros difere.} \end{cases}$$

Na Figura 2.8 apresenta-se uma série temporal simulada que ilustra uma mudança simultânea no ponto k=50.

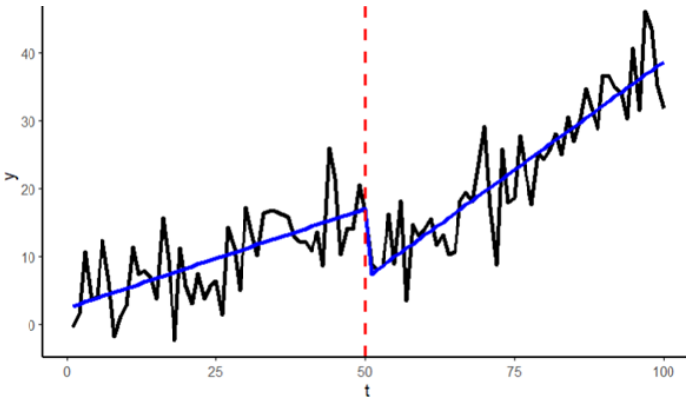


Figura 2.8: Uma mudança na tendência

2.3.3 Mudanças no parâmetro forma da distribuição GEV

Na Secção 2.2, foi apresentado o enquadramento teórico da teoria dos valores extremos. Nesta secção, analisamos possíveis mudanças no parâmetro de forma (ξ) da distribuição GEV, uma vez que este parâmetro controla a natureza das caudas da distribuição. Mudanças em ξ permitem distinguir entre diferentes regimes de comportamento extremo.

Seja X_t a variável aleatória que representa o valor extremo observado no instante $t \in \{1, 2, \ldots, T\}$. Admitimos que os dados seguem uma distribuição GEV,

$$X_t \sim \text{GEV}(\mu_t, \sigma_t, \xi_t),$$

em que:

- $\mu_t \in \mathbb{R}$ é o parâmetro de localização no instante t;
- $\sigma_t > 0$ é o parâmetro de escala no instante t;
- $\xi_t \in \mathbb{R}$ é o parâmetro de forma no instante t, associado ao regime de cauda.

O problema de interesse consiste na deteção de mudanças no parâmetro de forma da distribuição GEV ao longo do tempo, isto é, pretende-se testar:

$$H_0 : \xi_1 = \xi_2 = \cdots = \xi_T = \xi \quad \text{versus} \quad H_1 : \exists\, t, s \in \{1, \ldots, T\} \text{ tais que } \xi_t \neq \xi_s.$$

Os gráficos apresentados mostram séries temporais de **valores extremos simulados** a partir de uma distribuição GEV em dois cenários distintos. Consideramos observações *i.i.d.* (independentes e identicamente distribuídas), com parâmetros de localização $\mu = 0$ e escala $\sigma = 1$, constantes ao longo do tempo. O gráfico I representa os valores simulados com $\xi = 0.2$ constante, enquanto que o gráfico II apresenta os valores simulados de dois modelos GEV(0, 1, ξ_t) em que o parâmetro de forma muda no instante $t = 50$, passando de $\xi = 0.2$ para $\xi = 0.5$.

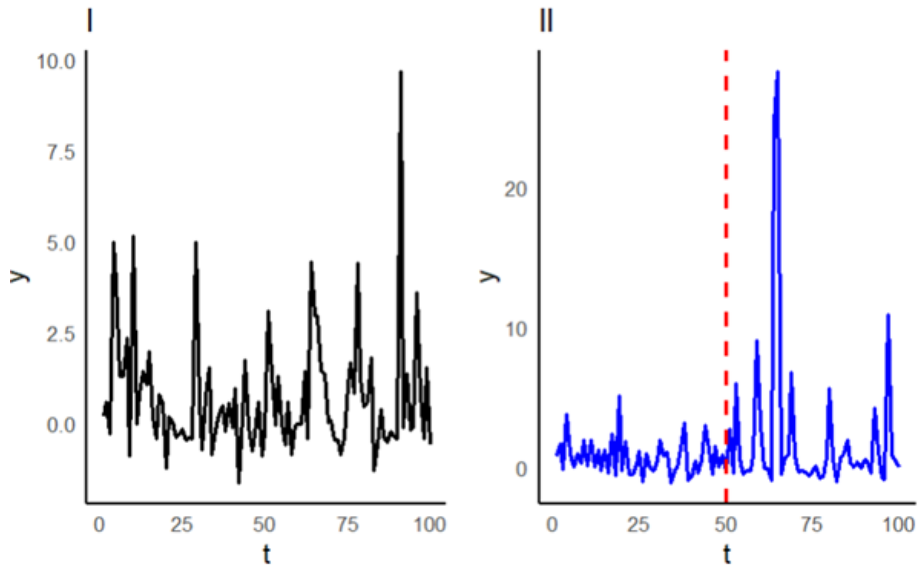


Figura 2.9: Simulação de dois modelos GEV sem (I) e com (II) mudança no parâmetro de forma

Capítulo 3

Metodologias de deteção de mudanças de estrutura

A metodologia adotada nesta investigação assenta, numa primeira fase, na aplicação de técnicas clássicas amplamente utilizadas para a deteção de mudanças de estrutura em séries temporais, nomeadamente para análise de mudanças de média, de tendência e mudanças na forma da distribuição, com especial atenção a mudança no parâmetro de forma, ξ, da distribuição generalizada de valores extremos, `GEV`. O enfoque principal é a análise do desempenho dos algoritmos de deteção existentes, com recurso a estudos de simulação de Monte Carlo. Para a realização dos estudos empíricos, geraram-se m séries temporais, construídas com base em diferentes estruturas estatísticas, de modo a testar a robustez dos métodos disponíveis na literatura.

Focamo-nos na abordagem **segmentária**, como o principal método de análise deste trabalho, com foco no algoritmo de segmentação binária, inicialmente proposto por Scott and Knott (1974) e mais recentemente aprofundado em Killick et al. (2010), o algoritmo *Pruned Exact Linear Time* (PELT), os testes `F` e das somas cumulativas (CUSUM), propostos em Zeileis et al. (2002b) baseados em algoritimos de programação dinâmica para avaliar a estabilidade dos parâmetros da série ao longo do tempo, em especial mudanças de média e de tendência. Para mudanças no parâmetro forma da distribuição GEV foram feitas simulações usando o método proposto em Fearnhead and Rigaill (2019), nomeadamente em contextos de dependência. As principais bibliotecas do software `R CRAN` utilizadas foram `changepoint`, `strucchange` e `NPCP`.

Para análise das mudanças de *média* e de *tendência*, exploramos o desempenho dos métodos em quatro cenários:

- Mudanças fracas ou pouco visíveis;
- Séries com dependência fraca, moderada e forte;
- Falha da suposição de normalidade;
- Séries com presença de *outliers*.

Quanto à análise de mudanças na forma da distribuição, analisamos:

- Mudanças no parâmetro de forma, ξ, da distribuição GEV, para distribuições de cauda pesada, ou seja, quando $\xi > 0$.

De acordo com metodologias usuais para estudos de simulação com séries temporais, e de forma a acomodar a variabilidade intrínseca que existe no processo de simulação das séries temporais, definimos uma metodologia para a seleção de uma janela de tolerância γ em torno do verdadeiro ponto de mudança k. Ou seja, sempre que o modelo deteta um ponto dentro do intervalo $k-\gamma, k+\gamma$, aceitamos como um ponto de mudança. Para tal, simulou-se um conjunto de séries com uma mudança de nível acentuada, um cenário onde se espera que os métodos acertem 100% das vezes. Avaliamos a taxa de acerto do algoritmo de segmentação para diferentes valores de γ. O valor de γ foi escolhido como aquele que proporcionou uma taxa de acerto próxima de 90%, sendo este posteriormente utilizado como referência nas análises subsequentes. Nas secções seguintes apresentamos a teoria estatística subjacente aos algoritmos de deteção usados neste trabalho.

3.1 Métodos de estimação de mudança de média

Um dos primeiros métodos originalmente proposto para detetar uma única mudança na série temporal estacionária é o `CUSUM clássico` Page (1955). Este método consiste num teste de hipóteses onde se testa a instabilidade na média de uma série temporal $\{X_t,\, t = 1, \ldots, n\}$. Ou seja,

$$H_0 :\ \mu_1 = \mu_2 = \cdots = \mu_n$$

contra a hipótese alternativa,

$$H_1 :\ \exists\ k \in \{1, \ldots, n-1\} \text{ tal que } \mu_1 = \mu_2 = \cdots = \mu_k\ \neq\ \mu_{k+1} = \cdots = \mu_n,$$

onde $E[X_t] = \mu_t$.

Dada a estatística de teste

$$C_k = \sqrt{\frac{k(n-k)}{n}}\left|\bar{x}_{1:k} - \bar{x}_{(k+1):n}\right|,$$

onde $\bar{x}_{1:k}$ e $\bar{x}_{(k+1):n}$ são as médias empíricas de cada segmento, calculadas por: $\bar{x}_{l:u} = \frac{1}{u-l+1}\sum_{t=l}^{u} x_t$, a estatística CUSUM compara, para um $k \in \{1, \ldots, n-1\}$ fixo, a média empírica antes de k com a média empírica depois de k.

Sob a hipótese de independência e/ou dependência fraca e assumindo que $X_t \sim \mathcal{N}(\mu, \sigma^2)$, então sob H_0, a estatística de teste $\frac{C_k}{\sigma}$ segue uma normal padrão com média 0 e variância 1, e $\frac{C_k^2}{\sigma^2} \sim \chi_1^2$. Existe uma mudança em k se:

$$\frac{C_k^2}{\sigma^2} > c,$$

onde $c \in R^+$ é um valor de limiar escolhido. A validade assintótica é garantida também pelo Teorema Central do Limite (TCL), ver Yao and Davis (1986).

Agora precisamos considerar todas as possíveis localizações de pontos de mudança e escolher aquela que maximiza a estatística de teste. Assim, consideramos a extensão do teste CUSUM:

$$C^2_{\max} = \max_{k\in\{1,\ldots,n-1\}} \frac{C_k^2}{\sigma^2}.$$

Se detetarmos um ponto de mudança (ou seja, se $C^2_{\max} > c$), podemos estimar a sua localização da seguinte forma:

$$\hat{k} = \arg \max_{k\in\{1,\ldots,n-1\}} C_k^2, \tag{3.1}$$

onde $\hat{k}$ é o valor de k que maximiza a estatística CUSUM.

Uma estimativa simples da magnitude do salto é dada por:

$$\Delta\tilde{\mu} = \bar{x}_{(\hat{k}+1):n} - \bar{x}_{1:\hat{k}}. \tag{3.2}$$

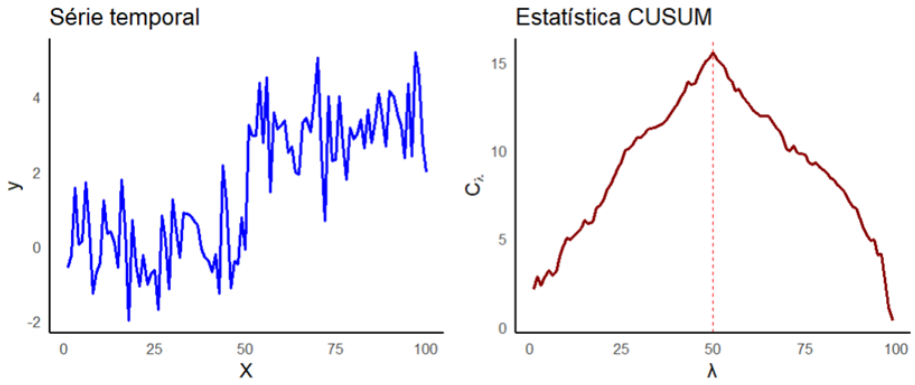


Figura 3.1: Estimação de $\hat{k}$ com Estatística CUSUM

A operação de máximo introduz complexidade adicional devido à dependência entre as estatísticas para diferentes k, por outro lado, a natureza não regular do problema afeta a distribuição assintótica da estatística de teste e estas são diferenças fundamentais entre testar um ponto de mudança conhecido *versus* desconhecido.

Em Yao and Davis (1986) prova-se que $(C_1, \ldots, C_{n-1})/\sigma$ converge para um processo Gaussiano com média 0 e covariância conhecida e que o máximo de um conjunto de variáveis aleatórias Gaussianas converge para uma distribuição de Gumbel, conforme a equação:

$$\lim_{n\to\infty} \Pr\left\{a_n^{-1}\left(\max_k \frac{C_k}{\sigma} - b_n\right) \le u_\alpha\right\} = \exp\left\{-(2\pi)^{-1/2}\exp(-u_\alpha)\right\},$$

onde $a_n = (2\log\log n)^{-1/2}$ (constante de escala) e $b_n = a_n^{-1} + 0.5a_n \log\log\log n$ (constante de localização).

Estas constantes de normalização são necessárias para evitar que o máximo se torne degenerado à medida que $n \to \infty$. Como já mencionado na secção 2.2.1, a distribuição normal está no domínio de atração da Gumbel, ou seja, o máximo

de variáveis normais, devidamente normalizado, converge para uma distribuição Gumbel.

Para determinar o limiar c que controla a taxa de falsos positivos, igualamos CDF da distribuição Gumbel a $1-\alpha$ e resolvemos em ordem a u_α obtendo:

$$u_\alpha = -\log\left(-\frac{\log(1-\alpha)}{(2\pi)^{-1/2}}\right).$$

Deste modo, o valor crítico é dado por $\tilde{c} = a_n u_\alpha + b_n$ e para obter o limiar c_α, como $\max_k(C_k^2/\sigma^2) > c$, basta elevar $\tilde{c}$ ao quadrado: $c = \tilde{c}^2$.

Este resultado assintótico indica que o limiar c para $\frac{C_k^2}{\sigma^2}$ cresce com n a uma taxa de $\mathcal{O}(2\log\log n)$. É um resultado assintótico útil quando o tamanho da amostra é suficientemente grande.

Para amostras pequenas, é recomendável utilizar o *método de Monte Carlo* (MMC), que se baseia na distribuição empírica dos dados originais, sob a hipótese nula H_0, geralmente é mais conservador. Para M séries $\{x_t^{(m)}\}_{t=1}^n$, $m = 1, \ldots, M$, sob a hipótese nula H_0 (sem ponto de mudança), o limiar MMC começa por simular:

$$C_{\max}^{(m)} = \max_{k\in\{1,\ldots,n-1\}} \frac{\left(C_k^{(m)}\right)^2}{\sigma^2}.$$

Com base nos M valores simulados obtém-se a distribuição empírica de $C_{\max}^2$ e define-se o limiar como:

$$c^{(\mathrm{MMC})} = \mathrm{Quantil}_{1-\alpha}\left(\left\{C_{\max}^{(m)}\right\}_{m=1}^M\right).$$

A decisão a tomar é a de Rejeitar H_0 se $C_{\max}^2 > c^{(\mathrm{MMC})}$. Note que o teste assume que sob H_0, não existe um ponto de mudança na série.

O problema da deteção de mudanças na média adquire uma complexidade significativamente maior quando a análise incide sobre séries temporais muito instáveis ao longo do tempo ou seja quando apresenta múltiplas mudanças. O teste `CUSUM clássico` já enunciado, fora inicialmente proposto para identificar uma única mudança de estrutura na média da série; contudo, a sua aplicação torna-se inadequada em contextos com mais do que uma mudança. Os métodos de segmentação, resolvem este problema ao dividir recursivamente a série em segmentos, aplicando a estatística de mudança em cada segmento e permitindo a identificação eficiente de múltiplos pontos de mudança na série temporal.

3.1.1 Métodos de segmentação

Os processos para análise de mudanças de estrutura, nomeadamente, mudanças de média, mais utilizado é provavelmente, o processo de segmentação da série, cuja origem remonta aos trabalhos de Scott and Knott (1974) e Sen and Srivastava (1975) e posteriormente implementado em bibliotecas do software *R CRAN* por Killick et al. (2010). De forma sucinta, este tipo de processo aplica primeiro um teste de ponto de mudança único a toda a série. Se for identificado um ponto de mudança, a série é dividida nesse ponto. Em seguida, o procedimento é repetido nos dois novos subconjuntos de dados — antes e depois da mudança. Se forem detetadas novas mudanças em qualquer um desses subconjuntos, eles são novamente divididos. Este

processo continua até que já não sejam encontrados pontos de mudança em nenhuma das partes, sendo usualmente denominado métodos recursivos de deteção.

Por exemplo, métodos de segmentação como o `BinSeg` (*Binary Segmentation*) oferecem uma aproximação eficiente para a deteção de mudanças, com complexidade $\mathcal{O}(n \log n)$, ao considerar apenas um subconjunto das $2^n - 1$ segmentações possíveis. Em contraste, algoritmos como o `PELT` procuram a segmentação ótima global, mantendo elevada eficiência computacional graças ao corte inteligente, enquanto abordagens exatas completas testam todas as segmentações possíveis, com custo computacional muito elevado. Uma variação relevante é o algoritmo `WBS` (*Wild Binary Segmentation*) Fryzlewicz (2014), que melhora a abordagem do `BinSeg` ao aplicar a segmentação em múltiplos intervalos aleatórios, tornando a deteção de mudanças múltiplas mais eficaz, sobretudo quando estas ocorrem em posições próximas. Esses métodos clássicos geralmente pressupõem que os dados sejam independentes ou apresentem dependência fraca dentro de cada segmento e tendem a funcionar melhor em dados normais ou aproximadamente normais, ver Basseville et al. (1993), Truong et al. (2020) e (Horváth and Rice, 2024, pp. 49–50).

Para compreendermos a noção do processo de `segmentação`, vamos assumir que X_t é um processo particionado (em segmentos), com distribuição Gaussiana (ou normal de parâmetro θ, onde θ é o vetor que contém μ e σ. Nestas condições, define-se o custo associado a um segmento, dado por:

$$\mathcal{L}(x_{s+1:t}) = \min_{\theta} \sum_{i=s+1}^{t} -2 \log f(x_i; \theta) \tag{3.3}$$

onde $f(x_i; \theta)$ é a função densidade de probabilidade da normal.

Aplicando o logaritmo à função densidade de probabilidade e somando para $i = s + 1$ até t, obtém-se:

$$\sum_{i=s+1}^{t} -2 \log f(x_i; \theta) = (t - s) \log(2\pi\sigma^2) + \frac{1}{\sigma^2} \sum_{i=s+1}^{t} (x_i - \mu)^2.$$

Para minimizar a expressão relativamente a μ, derivamos:

$$\frac{d}{d\mu} \left(\sum_{i=s+1}^{t} (x_i - \mu)^2 \right) = 2(t - s)\mu - 2 \sum_{i=s+1}^{t} x_i,$$

$$\Rightarrow \mu = \frac{1}{t - s} \sum_{i=s+1}^{t} x_i = \bar{x}_{s+1:t}.$$

Substituindo $\mu = \bar{x}_{s+1:t}$:

$$\mathcal{L}(x_{s+1:t}) = \frac{1}{\sigma^2} \sum_{i=s+1}^{t} (x_i - \bar{x}_{s+1:t})^2.$$

O custo da segmentação completa será dado por:

$$\sum_{m=0}^{M} \mathcal{L}(x_{k_m+1:k_{m+1}})$$

, onde M é o número total de pontos de mudanças. No caso em que $M = 1$ e $k_1 = k$, temos dois segmentos, $x_{1:k}$ e $x_{k+1:n}$, cujo custo de segmentação é dado por:

$$\mathcal{L}(x_{1:k}) + \mathcal{L}(x_{k+1:n}) = \frac{1}{\sigma^2}\left[\sum_{i=1}^{k}(x_i - \bar{x}_{1:k})^2 + \sum_{i=k+1}^{n}(x_i - \bar{x}_{k+1:n})^2\right].$$

Contudo, como não temos um critério de paragem, podemos incorrer no risco de aumentar significativamente o erro do tipo I (ou seja, detetar uma mudança quando não houve nenhuma de facto) e enfrentar problemas de sobreajustamento. Um dos métodos usados para lidar com este problema é o algoritmo de `segmentação binária`, que começa por verificar se o custo de uma segmentação satisfaz a seguinte condição:

$$\mathcal{L}(x_{1:n}) - \Big(\mathcal{L}(x_{1:k}) + \mathcal{L}(x_{k+1:n})\Big) > \beta \tag{3.4}$$

onde $\mathcal{L}(\cdot)$ está definido em (3.3), e $\beta \in \mathbb{R}$ representa uma penalização ou critério de paragem.

Se esta condição for satisfeita para algum $k \in \{1, \ldots, n\}$, considera-se que o custo foi reduzido, e o teste é repetido recursivamente nos dois segmentos gerados por essa divisão. O processo continua até já não haver nenhuma divisão que reduza o custo (isto é, quando, para todos os k, não se verifica a condição em 3.4).

No caso em que haja proximidade entre as mudanças, o algoritmo `Wild Binary Segmentation (WBS)`, um procedimento recursivo que melhora em alguns cenários as estimativa dos pontos de mudança ao considerar múltiplos subintervalos aleatórios da sequência, para uma melhor discussão sobre este método ver Fryzlewicz (2014).

A noção de custo associada à `segmentação ótima` é dada por:

$$Q_{n,\beta} = \min_{M\in\mathbb{N}}\left[\min_{k_1,\ldots,k_M}\sum_{m=0}^{M}\mathcal{L}(x_{k_m+1:k_{m+1}}) + \beta M\right], \tag{3.5}$$

onde $Q_{n,\beta}$ representa o custo ótimo de segmentar os dados até ao instante n, com uma penalização β que aumenta com um novo ponto de mudança. A equação (3.5) corresponde precisamente à formulação de *Optimal Partitioning* (OP), isto é, o problema de encontrar a segmentação globalmente ótima através da minimização do custo penalizado.

Neste problema, podemos reduzir o número de verificações a serem realizadas em cada iteração, diminuindo a complexidade. Esta operação é chamada de *pruning*. Especificamente, dada uma função de custo $\mathcal{L}(\cdot)$, sob a condição de que existe uma constante C tal que, para todo $l < C < u$:

$$\mathcal{L}(x_{l+1:k}) + \mathcal{L}(x_{k+1:u}) + C \leq \mathcal{L}(x_{l+1:u}) \tag{3.6}$$

é possível realizar a poda sem recorrer a uma aproximação. Para muitas funções de custo, como a *função de custo Gaussiana*, tal constante C existe.

Então, para qualquer $k < t$, se

$$Q_{k,\beta} + \mathcal{L}(x_{k+1:t}) \;\geq\; Q_{t,\beta} - C \tag{3.7}$$

for verdadeiro, então para qualquer $T > t$, k **nunca poderá ser o ponto de mudança ótimo antes do tempo** T.

Usando a condição da Equação 3.7, o algoritmo **PELT** –Killick et al. (2012) resolve exatamente a minimização penalizada da Equação 3.5 com um custo computacional esperado que pode ser **linear em (O(n)** – mantendo, no entanto, complexidade $\mathcal{O}(n^2)$ no pior caso, particularmente quando a condição de poda raramente é satisfeita.

Ou seja, ao definir $\kappa = 0$, se

$$Q_{k,\beta} + \mathcal{L}(x_{k+1:t}) \geq Q_{t,\beta},$$

então podemos efetuar o corte com segurança no custo do segmento relacionado com k, já que k nunca será o ponto de mudança ótimo até qualquer tempo $T > t$ no futuro. Substituindo o custo anterior no lado direito:

$$Q_{k,\beta} + \mathcal{L}(x_{k+1:t}) \geq \min_{0 \leq k < t} \big[Q_{k,\beta} + \mathcal{L}(x_{k+1:t}) + \beta \big],$$

vemos como o β desempenha novamente um papel central, já que está ausente no lado esquerdo. A intuição é que, uma vez introduzido um novo ponto de mudança candidato, é então possível realizar o corte.

A seleção de β pode ser fixa, ou escolhida com base em critérios estatísticos. AS penalizações mais usadas nos softweres, correspondem aos critérios `AIC, BIC` e `MBIC`.

A penalização `AIC` assume o valor de $2p$, onde p é o número de parâmetros adicionados ao modelo. Embora seja simples de aplicar, o AIC é conhecido por ser *assintoticamente inconsistente* e tende a sobrestimar o número de pontos de mudança à medida que o tamanho da amostra aumenta. Intuitivamente, isto ocorre porque o AIC é projetado para minimizar o *erro de previsão*, e não necessariamente para identificar a estrutura verdadeira do modelo. Favorece modelos que se ajustam bem aos dados, o que frequentemente leva à inclusão de mais pontos de mudança do que o necessário. O `BIC` (Critério de Informação Bayesiano) tem como penalização $p \log(n)$. Nas nossas abordagens, isto traduz-se em $\beta = 2\log(n)$, sendo adicionada para cada ponto de mudança adicional. O BIC é geralmente mais conservador que o AIC e é *consistente*, ou seja, não sobrestima o número de pontos de mudança à medida que o tamanho da amostra cresce.

O `MBIC` (BIC Modificado) foi proposto por Zhang and Siegmund (2007), como uma extensão do BIC que inclui um termo extra para considerar o espaçamento entre os pontos de mudança. Na prática, esta penalização pode ser aproximada por: $\beta = 3\log(n)$. É ainda mais conservadora que o BIC. Em termos práticos, estas escolhas de penalização representam um compromisso fundamental entre *sobreajustamento* (detetar pontos de mudança em excesso) e *subajustamento* (não detetar mudanças que de facto existem).

3.2 Métodos de estimação de mudanças na tendência

No contexto da deteção de mudanças na tendência de uma série temporal, recorremos às metodologias propostas em Zeileis et al. (2002b), baseadas na regressão linear.

Considere-se, assim, a equação linear:

$$y_i = x_i^\top \beta_i + \varepsilon_i \quad (i = 1, \ldots, n), \tag{3.8}$$

onde, onde, no tempo i, y_i representa a variável dependente observada, $x_i = (1, x_{i2}, \ldots, x_{ip})^\top$ é um vetor $p \times 1$ de variáveis independentes (com o primeiro termo constante), β_i é o vetor $(p \times 1)$ dos coeficientes de regressão, assumido constante dentro de cada segmento e podendo variar entre segmentos, e ε_i são os erros i.i.d. de média nula e variância constante σ^2.

Os testes de mudança de estrutura têm como objetivo testar a hipótese nula:

$$H_0 : \ \beta_1 = \beta_2 = \cdots = \beta_n,$$

contra a hipótese alternativa de que existe pelo menos um instante em que os coeficientes diferem:

$$H_1 : \ \exists\, i, j \in \{1, \ldots, n\} \text{ tal que } \beta_i \neq \beta_j.$$

Assume-se que os regressores são não estocásticos, com $\|x_i\| = \mathcal{O}(1)$, e que:

$$\frac{1}{n} \sum_{i=1}^{n} x_i x_i^\top \to Q \tag{3.9}$$

para alguma matriz Q finita e não singular.

Seja $\hat{\beta}(n)$ o estimador dos Mínimos Quadrados Ordinários (MQO) dos coeficientes de regressão baseado em todas as observações até ao instante n. Os resíduos são então dados por

$$\hat{\varepsilon}_i = y_i - x_i^\top \hat{\beta}(n),$$

e a variância dos resíduos é estimada pela forma usual:

$$\hat{\sigma}^2 = \frac{1}{n-p} \sum_{i=1}^{n} \hat{\varepsilon}_i^2.$$

Nos testes de mudança de tendência são frequentemente utilizados resíduos recursivos. Denotando por $\hat{\beta}(i, j)$ os MQO dos coeficientes de regressão, com base nas observações de $i+1$ até $i+j$, e por $\hat{\beta}(i) = \hat{\beta}(0, i)$ os estimadores baseados em todas as observações até o tempo i, os resíduos recursivos são definidos como:

$$\hat{w}_i = \frac{y_i - x_i^\top \hat{\beta}(i-1)}{\sqrt{1 + x_i^\top \left(X^{(i-1)\top} X^{(i-1)}\right)^{-1} x_i}}, \quad i = p+1, \ldots, n,$$

onde $X^{(i)}$ é a matriz dos regressores até à observação i.

Os autores Zeileis et al. (2002b) demonstram que, sob a hipótese nula, estes resíduos recursivos têm média nula e variância σ^2, sendo esta última estimada como:

$$\hat{\sigma}^2 = \frac{1}{n-p} \sum_{i=p+1}^{n} (\hat{w}_i - \bar{\hat{w}})^2. \tag{3.10}$$

, onde

Ainda em Zeileis et al. (2002b), salienta-se que as suposições de que os resíduos são i.i.d., de que os regressores x_i são não estocásticos e de que a condição

$$\frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top \to Q \tag{3.11}$$

converge para uma matriz finita e não singular podem não ser válidas em certos modelos dinâmicos. Nestes casos, são necessárias adaptações metodológicas adequadas. Propomos, portanto, avaliar estes métodos também em condições para as quais originalmente não foram concebidos, de modo a testar a sua robustez e aplicabilidade fora dos pressupostos standard.

De forma geral, os testes de deteção de mudanças de estrutura podem ser divididos em duas grandes classes: **testes baseados nos parâmetros** e **testes baseados nos resíduos**. Os primeiros avaliam diretamente a estabilidade dos coeficientes de regressão ao longo do tempo, enquanto os segundos analisam as flutuações dos resíduos para detetar eventuais quebras na estrutura do modelo.

O teste mais simples e clássico para a deteção de mudanças de estrutura é o **teste de Chow** Chow (1960), baseado na estatística F, no qual se testa a significância de um ponto de mudança conhecido, denotado por t_0. Este teste baseia-se na hipótese nula de que os coeficientes de regressão permanecem constantes ao longo do tempo.

A ideia consiste em ajustar duas regressões lineares com p variáveis explicativas às subamostras antes e depois de t_0 (designadas por amostras A e B) e comparar estas com o modelo restrito ajustado a toda a amostra. A estatística de teste é dada por

$$F_{t_0} = \frac{SMQO_p - (SMQO_A + SMQO_B)}{SMQO_A + SMQO_B}\frac{n_A + n_B - 2p}{p}, \tag{3.12}$$

onde $SMQO_p$ representa a soma dos quadrados dos erros do modelo ajustado à amostra total, e $SMQO_A$ e $SMQO_B$ representam, respetivamente, as somas dos quadrados dos erros dos modelos ajustados às amostras A e B. Em Chow (1960) prova-se que F_{t_0} segue uma distribuição F com p e $n_A + n_B - 2p$ graus de liberdade.

Quando o ponto de mudança é desconhecido, conforme discutido em Andrews (1993) e Andrews and Ploberger (1994), constroem-se extensões da estatística F_{t_0} ao longo do intervalo $t_0 \in \{p+1, \ldots, n-p\}$, combinando-se as evidências através das seguintes estatísticas:

$$\begin{aligned}
\sup F &= \sup_{t_0 \in \{p+1,\ldots,n-p\}} F_{t_0}, \\
\text{ave}F &= \frac{1}{n-2p+1}\sum_{t_0=p+1}^{n-p} F_{t_0}, \\
\exp F &= \log\left(\frac{1}{n-2p+1}\sum_{t_0=p+1}^{n-p} \exp\left(\frac{1}{2}F_{t_0}\right)\right).
\end{aligned}$$

Estas estatísticas resumem a evidência de mudança estrutural ao longo do tempo, sendo os testes ave$F$ e exp$F$ particularmente relevantes por apresentarem propriedades de otimalidade em certos contextos, conforme discutido em Andrews and Ploberger (1994).

Os testes baseados nos resíduos avaliam a estabilidade do modelo através da análise das flutuações dos resíduos ou das estimativas dos parâmetros. Um exemplo

clássico são os testes de flutuação, que englobam os processos *CUSUM* e *MOSUM* Kleiber (2002). A ideia central consiste em ajustar o modelo aos dados e derivar um processo empírico que capture a flutuação das quantidades de interesse ao longo do tempo.

Para estes processos empíricos, são conhecidos os processos-limite, permitindo o cálculo de limites críticos cuja probabilidade de serem ultrapassados sob H_0 é α. Se o caminho do processo empírico ultrapassar esses limites, considera-se a flutuação improvavelmente grande, levando à rejeição da hipótese nula ao nível de significância α. Na biblioteca `strucchange`, estes testes são implementados através da função `efp` (*empirical fluctuation process*).

Quando as observações não são independentes — como é comum em séries temporais —, os processos de flutuação podem ser ajustados utilizando estimativas consistentes da matriz de covariância, como os estimadores HAC (*heteroskedasticity and autocorrelation consistent*) Andrews (1991); Lumley and Heagerty (1999), garantindo validade assintótica mesmo na presença de autocorrelação ou heterocedasticidade.

O processo **Rec-CUSUM**, proposto por Brown et al. (1975), é definido por

$$W_n(t) = \frac{1}{\tilde{\sigma}\sqrt{\eta}} \sum_{i=p+1}^{p+\lfloor t\eta \rfloor} \tilde{\varepsilon}_i, \quad 0 \leq t \leq 1, \tag{3.13}$$

onde $\eta = n - p$ é o número de resíduos recursivos. Sob H_0, o processo limite de $W_n(t)$ é um **processo de Wiener** $W(t)$, de acordo com o Teorema do Limite Central Funcional (FCLT)[1]. Sob a hipótese alternativa, caso exista um ponto de mudança estrutural t_0, o processo tende a desviar-se da média zero após esse ponto.

O teste **OLS-CUSUM** baseia-se nas somas cumulativas dos resíduos obtidos por MQO, sendo definido por

$$W_n^0(t) = \frac{1}{\hat{\sigma}\sqrt{n}} \sum_{i=1}^{\lfloor nt \rfloor} \hat{\varepsilon}_i, \quad 0 \leq t \leq 1. \tag{3.14}$$

O processo limite correspondente é a **ponte browniana**[2], definida por $W^0(t) = W(t) - tW(1)$. Este processo inicia e termina em zero, apresentando desvios locais quando ocorre uma quebra estrutural. Ambos os processos estão disponíveis na função `efp`, especificando o argumento `type` como `"Rec-CUSUM"` ou `"OLS-CUSUM"`, respetivamente. As propriedades teóricas destes processos são detalhadas em Chu et al. (1995).

Para a análise de mudanças na média e, em particular, mudanças na tendência, recorre-se ao método de programação dinâmica proposto por **Bai e Perron** Bai and Perron (2003a), que permite estimar simultaneamente múltiplos pontos de quebra $i_1, \ldots, i_m$ através da minimização da soma dos quadrados dos resíduos segmentados:

$$RSS(i_1, \ldots, i_m) = \sum_{j=1}^{m+1} rss(i_{j-1} + 1, i_j), \tag{3.15}$$

[1] O *Teorema do Limite Central Funcional* (FCLT) estabelece que as somas parciais de variáveis aleatórias centradas e normalizadas convergem, em distribuição, para um processo de Wiener (movimento browniano).

[2] Uma *ponte browniana* é um processo estocástico obtido a partir de um movimento browniano $W(t)$, condicionado a começar e terminar em zero, sendo definido por $W^0(t) = W(t) - tW(1)$.

onde rss representa a soma dos quadrados dos resíduos em cada segmento e RSS a soma total, com $i_0 = 0$ e $i_{m+1} = n$. Os pontos de quebra estimados são dados por

$$(\hat{i}_1, \ldots, \hat{i}_m) = \arg\min_{i_1,\ldots,i_m} RSS(i_1, \ldots, i_m), \tag{3.16}$$

sujeito à restrição $i_j - i_{j-1} \geq n_h$, onde n_h representa o número mínimo de observações em cada segmento.

Para mais de duas quebras, uma busca exaustiva seria computacionalmente inviável, com complexidade $\mathcal{O}(n^m)$. No entanto, o uso de **programação dinâmica** reduz a complexidade para $\mathcal{O}(n^2)$, recorrendo à recursão baseada no princípio de Bellman Bellman (1957):

$$RSS(I_m, n) = \min_{mn_h \leq i \leq n - n_h} \left[RSS(I_{m-1}, i) + rss(i+1, n) \right]. \tag{3.17}$$

Este algoritmo encontra-se implementado na função `breakpoints` do pacote `strucchange`, permitindo a estimação eficiente de múltiplas quebras estruturais.

3.3 Método de deteção de mudanças nos parâmetros da distribuição GEV

A abordagem utilizada neste trabalho é baseada em Kojadinovic and Naveau (2017), que propõem o uso dos estimadores PWM e GPWM para deteção de mudanças nos parâmetros da GEV. A metodologia proposta no artigo permite testar a existência de mudanças na localização, na escala e na forma da distribuição GEV e assenta no pressuposto de que a amostra de máximos é composta por variáveis aleatórias independentes.

A principal inovação deste trabalho consiste em incorporar dependência serial nos dados simulados, de modo a avaliar a robustez dos métodos PWM e GPWM sob diferentes cenários: dependência fraca ($\phi_1 = 0.2$) onde se prevê que a metodologia proposta em Kojadinovic and Naveau (2017) ainda seja válida, moderada ($\phi_1 = 0.5$) e dependência forte ($\phi_1 = 0.8$), onde poderá haver mais dificuldade do método em detetar mudança, uma vez que esta metodologia não foi projetada para estes cenários.

Estatísticas de teste

A deteção de mudanças no parâmetro de forma ξ pode ser efetuada através de testes baseados em estatísticas do tipo CUSUM desenvolvidas na secção 3.1 e aplicadas aos estimadores de ξ. A ideia central consiste em avaliar, ao longo da amostra, possíveis quebras na homogeneidade da distribuição dos máximos por bloco, recorrendo a comparações sistemáticas entre subconjuntos de dados.

No caso do método **PWM**, a estatística de teste para ξ é definida por:

$$S_{g_\xi, n} = \max_{1 \leq k \leq n-1} \frac{k(n-k)}{n^{3/2}} \mathbf{1}(\hat{\beta}_{1:k} \in D_\xi, \hat{\beta}_{k+1:n} \in D_\xi) \left| g_\xi(\hat{\beta}_{1:k}) - g_\xi(\hat{\beta}_{k+1:n}) \right|, \tag{3.18}$$

onde:

- $\mathbf{1}(a \in A)$ representa a função indicatriz;
- $\hat{\beta}_{1:k}$ e $\hat{\beta}_{k+1:n}$ são, respetivamente, os vetores de estimadores PWM calculados a partir das observações anteriores e posteriores ao ponto de corte k;
- $g_\xi(\cdot)$ é a função que estabelece a relação entre os momentos ponderados e o parâmetro ξ;
- D_ξ representa o domínio admissível dos estimadores.

De forma análoga, para o método **GPWM**, a estatística de teste assume a forma:

$$S_{h_\xi,n} = \max_{1\leq k\leq n-1} \frac{k(n-k)}{n^{3/2}} \mathbf{1}(\hat{\beta}_{1:k} \in D_h, \hat{\beta}_{k+1:n} \in D_h) \left| h_\xi(\hat{\beta}_{1:k}) - h_\xi(\hat{\beta}_{k+1:n}) \right|, \quad (3.19)$$

em que $h_\xi(\cdot)$ corresponde à função de ligação definida no contexto GPWM e D_h ao respetivo domínio.

Em ambos os casos, as estatísticas são de natureza *não paramétrica*, no sentido em que não exigem explicitamente que os dados seguem uma distribuição GEV. Contudo, nos estudos de simulação, as amostras são geradas de acordo com uma distribuição GEV. Esta formulação permite que os testes sejam particularmente sensíveis a mudanças no parâmetro ξ, sendo o método PWM mais simples e clássico, enquanto o GPWM, pela sua generalização funcional, oferece maior flexibilidade e, em certos cenários, melhor desempenho em termos de poder estatístico (ver secção 3 de Kojadinovic and Naveau (2017)).

Procedimento de simulação

A distribuição GEV não é fechada para soma (convolução), ou seja, se $X \sim \text{GEV}(\mu_X, \sigma_X, \xi_X)$ e $Y \sim \text{GEV}(\mu_Y, \sigma_Y, \xi_Y)$, em geral

$$Z = aX + bY \not\sim \text{GEV}.$$

Deste modo, a solução encontrada para gerar processos GEV com dependência temporal modelada por processos de tipo AR(1), consiste em:

1º) Gerar um processo, por exemplo AR(1):

$$Z_t = \phi_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0,1) \quad (3.20)$$

2º) Transformar a série via Transformação Uniformizante:

$$U_t = \Phi(Z_t),$$

onde $U_t \sim U(0,1)$ com a estrutura de dependência AR(1).

3º) Utilizar a função inversa da GEV para transformar U_t:

$$X_t = F^{-1}_{\text{GEV}}(U_t, \xi). \quad (3.21)$$

Deste modo, X_t mantém a dependência temporal da série original. Esta dependência é usualmente denominada de dependência de cópula.

Capítulo 4

Estudo computacional

Tal como definido no início do trabalho, o principal objetivo deste estudo consiste numa análise crítica dos métodos[1] clássicos de deteção de mudanças de estrutura em séries temporais, com especial atenção aos métodos de segmentação, com vista à sua melhoria ou, eventualmente, ao desenvolvimento de uma nova abordagem. Esta análise assume particular relevância, dado que os métodos discutidos na Secção 3 foram concebidos para observações independentes e/ou com dependência fraca, e tendem a funcionar melhor sob a suposição de normalidade dos dados. Nesta análise computacional, avaliamos o desempenho destes métodos clássicos em cenários que violam tais suposições, nomeadamente na presença de falhas de normalidade e de diferentes graus de autocorrelação. Adicionalmente, analisamos o seu desempenho na presença de *outliers*, considerando diferentes dimensões da série temporal, bem como diferentes magnitudes de mudança.

4.1 Escolha da janela de tolerância

A estimação de pontos de mudança em séries temporais, mesmo em cenários simulados e controlados, está sujeita a variabilidade estatística. Tal variabilidade decorre principalmente da aleatoriedade intrínseca dos dados e da fraca capacidade dos métodos em detetar mudanças de pequena magnitude. Alguns autores recorrem, por esse motivo, à utilização de janelas de tolerância Ma et al. (2020), classificando como verdadeiros pontos de mudança aqueles que se encontram dentro de um determinado raio em torno do ponto de mudança real.

Neste trabalho, validamos empiricamente a metodologia do raio ótimo através de resultados de simulação. Para esse efeito, simulámos 2000 séries temporais com estrutura do tipo AR(1), com parâmetro $\phi = 0.2$, em que os resíduos do modelo seguem uma distribuição normal com média zero e variância unitária. Considerou-se um tamanho amostral $n = 200$ e uma única mudança estrutural acentuada ($\mu_1 = 1$ e $\mu_2 = 4$), ver secção 4.2.1, localizada nos instantes $k = 50$, $k = 100$ e $k = 150$, correspondentes, respetivamente, a mudanças situadas em 25%, 50% e 75% do total da série.

Para cada cenário, estudámos o erro na estimação do ponto de mudança k, denotado por $\hat{k}$. Nos casos em que os algoritmos detetam múltiplas quebras, registámos o número dessas ocorrências, representado por L. O erro de localização do ponto de

[1]Para simplificar a apresentação das tabelas, adotamos as seguintes abreviações: PELT = PL, BinSeg = BS e o algoritmo de Bai–Perron = BP.

mudança é definido como:

$$e_i = \left| \hat{k}_i - k \right|, \tag{4.1}$$

onde $\hat{k}_i$, $i = 1, \ldots, 2000$, representa o ponto de mudança estimado na i-ésima simulação.

Admitindo que $\gamma = 1, \ldots, n$ representa o raio da janela de tolerância, a taxa de acertos (denotada por H_k) associada a cada valor de γ é definida por:

$$H_k = \frac{\#\{i : e_i \leq \gamma\}}{m}, \tag{4.2}$$

onde $m = 2000 - L$ corresponde ao número de simulações em que foi detetada apenas uma quebra estrutural.

As Tabelas 4.1 e 4.2 ilustram a construção do raio ótimo γ, com base nas taxas de acerto associadas aos pontos de mudança $k = 50$, $k = 100$ e $k = 150$, sendo este critério posteriormente utilizado no desenvolvimento do estudo de simulação para mudanças de nível (média) e de tendência.

Tabela 4.1: Definição de raio ótimo-Mudança na média

Tolerância γ $n = 200$, $\phi = 0.2$	BS			PL		
	$k = 50$	$k = 100$	$k = 150$	$k = 50$	$k = 100$	$k = 150$
0	65.53%	65.31%	64.08%	65.65%	66.17%	65.26%
1	83.85%	84.63%	82.94%	83.07%	85.44%	83.71%
2	91.43%	92.52%	91.59%	91.33%	93.13%	91.89%
3	95.30%	96.17%	95.74%	95.38%	96.61%	95.62%
4	97.53%	97.97%	97.66%	97.52%	97.93%	97.68%
5	98.65%	98.90%	98.71%	98.84%	98.76%	98.59%

Tabela 4.2: Definição de raio ótimo-Mudança na média e tendência

Tolerância γ $n = 200$, $\phi = 0.2$	BP		
	$k = 50$	$k = 100$	$k = 150$
0	59.85%	60.25%	59.20%
1	77.00%	77.60%	75.45%
2	84.65%	85.45%	83.50%
3	89.30%	90.15%	88.55%
4	92.80%	93.10%	91.10%
5	94.75%	95.20%	93.25%

Estabelecemos como janela de tolerância o menor raio γ que assegura uma taxa de acerto próximo de 90%, o que nos levou a adotar tolerâncias de ± 3 para as análises principais, embora os algoritmos de segmentação, como o PL e BS chegam aos 90% com ± 2, ver figura 4.1. Outro fator importante constatado é que, a posição do ponto de quebra não tem muita influencia nas estimativas dos métodos. Ressaltamos ainda que esse é um cenário ideal, adotado unicamente para definir uma regra empírica na escolha da janela de tolerância, outros contextos — como mudanças mais subtis e condições de dependência mais forte — também são considerados neste estudo, especialmente na avaliação do comportamento dos métodos frente a esses desafios.

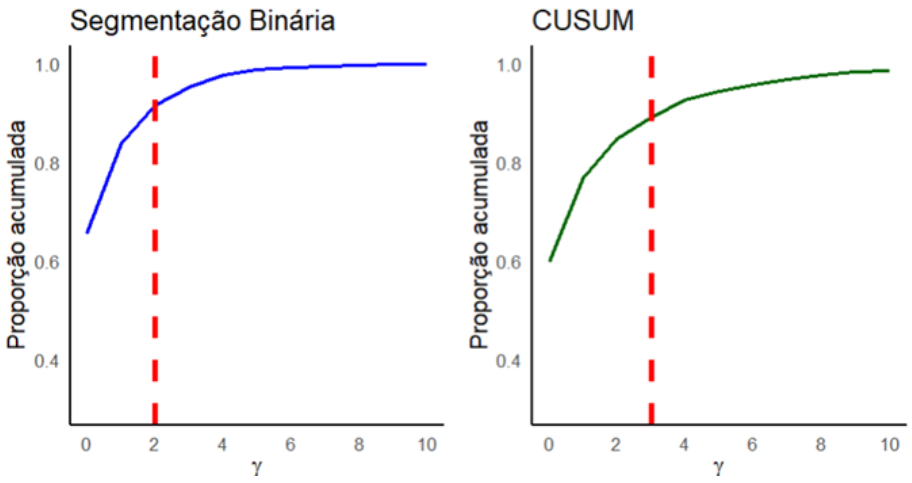


Figura 4.1: Proporção acumulada de taxas de acerto até o raio γ

4.2 Mudanças na média da série

Como já referido na secção 2.3.1, uma mudança na média da série temporal ocorre quando se verifica uma alteração — de pequena magnitude ou de magnitude mais elevada — no valor médio da série. Nesta secção, propomos estudar estes cenários em três contextos distintos: séries com resíduos normalmente distribuídos, séries com resíduos t-Student e séries afetadas pela presença de *outliers*.

4.2.1 Resíduos com distribuição normal

Antes de avançarmos para a análise de testes em séries temporais com estrutura de dependência, iniciamos com simulações de processos simples, gerados a partir de uma distribuição normal. Esse é exatamente o ambiente inicial projetado para os métodos clássicos, já que a maioria deles assumem independência ou dependência fraca, e tende a apresentar melhor desempenho em séries normais ou aproximadamente normais .

Consideremos o processo

$$X_t = \mu_i + \varepsilon_t, \quad t \in T, \tag{4.3}$$

em que μ_i denota a média no i-ésimo segmento e ϵ_t é um ruído branco normalmente distribuído, $\varepsilon_t \sim \mathcal{N}(0, 1)$.

Consideramos processos com uma mudança na média de μ_1 para μ_2, onde a magnitude de salto será dado por δ. Simulamos 2000 séries sem autocorrelação, de dimensões $n = \{200, 500, 1000\}$, com mudanças fracas, $\delta = |\mu_2 - \mu_1| \leq 1\sigma$, moderadas, $1.5\sigma \leq \delta \geq 2\sigma$, e fortes, $\delta \geq 3\sigma$, no ponto de mudança $k = 50$, onde σ é desvio padrão de ε_t

Tabela 4.3: Taxa de acertos do algoritmo de segmentação binária em cenários IID por penalizações AIC, BIC e MBIC.

	$n = 200$			$n = 500$			$n = 1000$		
δ	AIC	BIC	MBIC	AIC	BIC	MBIC	AIC	BIC	MBIC
0.5σ	36.95	21.55	6.55	36.20	23.60	5.35	36.30	20.45	4.70
1σ	73.10	70.55	70.00	73.65	72.30	69.55	73.25	69.90	71.25
1.5σ	98.30	97.25	96.50	97.65	96.90	97.65	97.55	96.80	97.20
2σ	99.60	99.25	99.40	99.70	99.35	99.55	99.75	99.40	99.45
3σ	99.85	99.85	99.80	100.00	99.85	99.75	99.95	99.85	99.80
4σ	100.00	100.00	99.95	100.00	100.00	100.00	100.00	100.00	100.00

Os resultados da Tabela 4.3, obtidos com o algoritmo `BS` mostram que, no caso em que não existe autocorrelação (resíduos IID), a taxa de deteção é bastante considerável em mudanças de magnitude mais elevadas, com valores acima de 95%, cenário este que muda em mudanças de pequena magnitude, onde apresenta-se taxas mais baixas. De salientar ainda que o tamanho da amostra ($n = 200, 500, 1000$) não altera substancialmente os resultados neste cenário IID.

A tabela 4.44.5 , mostram os resultados das simulações para outros métodos.

Tabela 4.4: PL

	$n = 200$			$n = 500$			$n = 1000$		
δ	AIC	BIC	MBIC	AIC	BIC	MBIC	AIC	BIC	MBIC
0.5σ	39.50	27.35	10.90	38.80	33.15	13.75	40.20	31.65	14.80
1.5σ	87.95	90.00	89.75	88.75	90.45	91.30	88.55	89.15	90.50
3σ	99.90	99.80	99.80	100.00	99.95	99.75	99.80	99.95	99.80

Tabela 4.5: BP

δ	$n = 200$	$n = 500$	$n = 1000$
0.5σ	29.5	33.5	30.1
1.5σ	90.5	92.5	93.1
3σ	100.0	100.0	100.0

Os resultados tanto para o método `PL`, como para `BP`, são parecidos com os obtidos pelo método `BS`, com taxas altas próximos dos 90% já em $\delta = 1.5$, embora a dificuldade continua em situação de mudanças fracas com taxas muito baixas de acerto.

Vamos considerar agora, cenários autocorelacionados, em que os ε_t da equação 4.3 seguem uma estrutura autoregressiva AR(1) de média nula e $\sigma = 1$.

A tabela 4.6, mostra os resultados para 2000 séries simuladas, de tamanho $n = 100$ e fixamos o ponto de mudança $k = 50$, mas fazendo variar o coeficiente do modelo AR(1), de um dependência fraca para uma forte.

Tabela 4.6: Taxas de acerto para $\phi = 0.2$,$\phi = 0.5$ e $\phi = 0.8$, usando os métodos `BS`, `BP` e `PL`

	$\phi = 0.2$			$\phi = 0.5$			$\phi = 0.8$		
δ	BS	PL	BP	BS	PL	BP	BS	PL	BP
0.5σ	17.60	18.30	17.5	10.75	22.05	10.0	6.10	48.05	11.5
1.5σ	83.85	81.05	82.0	59.75	62.10	64.5	23.20	58.60	26.0
3σ	99.50	99.00	98.5	93.60	93.00	94.0	58.90	78.40	64.5

Conforme se pode observar na tabela 4.6 autocorrelação em níveis de dependências mais forte ($\phi = 0.8$) diminui a taxa de acerto dos métodos e agrava-se quando comparado com cenário IID.

O gráfico da figura 4.2 mostra o desempenho do algoritmo à medida que alteramos o valor do parâmetro ϕ, mantendo a série com a mesma estrutura, e para diferentes magnitudes de mudanças.

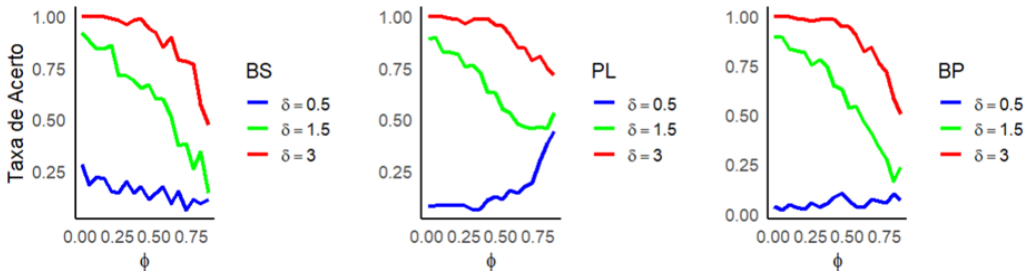


Figura 4.2: Variação da taxa de acertos em função do parâmetro ϕ, para os métodos `BS`, `BP`, e `PL`, considerando mudanças de pequenas magnitudes a maguinitudes mais elevadas

Observa-se que, à medida que o valor de ϕ aumenta, a taxa de acerto diminui em todos os cenários. A autocorrelação introduz uma estrutura de dependência que dificulta a deteção da mudança de nível, reduzindo a eficácia do algoritmo. O método `PL`, no caso das mudanças fraca e autocorrelação forte a taxa tende a aumentar, um comportamento não visto nos métodos `BS` e `BP`. A razão pode estar relacionada com o fato do método `PL` ser um método de otmizacão global, o que o torna potencialmente menos sensível a deteção de quebras espúrias[2] que aumentam na presença de autocorrelação e mudanças de pequena magnitude, ao contrário dos métodos `BS` e `BP` que com divisões sucessivas amplia efeito da dependência dos resíduos, ver 3.1.1 e 3.2.

[2]No contexto de deteção, uma quebra espúria refere-se a quando o método identifica uma mudança que não corresponde a uma alteração real. Ou seja, o método "julga" que houve uma quebra, mas esta aparente quebra resulta apenas de flutuações do ruído, da presença de autocorrelação ou de sobreajustamento do modelo.

As Tabelas 4.7, 4.8 e 4.9 apresentam as análises do sobreajustamento[3] dos métodos `BS`, `PL`, considerando diferentes escolhas de penalização: `AIC`, `BIC` e `MBIC`.

Tabela 4.7: Estimativa de Sobreajustamento (`AIC`)

	BS			PL		
ϕ	$\delta = 0.5\sigma$	$\delta = 1.5\sigma$	$\delta = 3\sigma$	$\delta = 0.5\sigma$	$\delta = 1.5\sigma$	$\delta = 3\sigma$
0.2	0.7335	0.1850	0.0050	11.1500	10.8800	11.1135
0.5	0.8760	0.4200	0.0675	16.7665	16.6725	16.4700
0.8	0.9400	0.7930	0.4070	21.0820	20.9985	21.1880

Tabela 4.8: Estimativa de Sobreajustamento (`BIC`)

	BS			PL		
ϕ	$\delta = 0.5\sigma$	$\delta = 1.5\sigma$	$\delta = 3\sigma$	$\delta = 0.5\sigma$	$\delta = 1.5\sigma$	$\delta = 3\sigma$
0.2	0.4755	0.1850	0.0050	0.8375	0.5885	0.4495
0.5	0.7445	0.4195	0.0675	4.1775	4.0675	3.9315
0.8	0.9390	0.7920	0.4070	10.4385	10.4960	10.4130

Tabela 4.9: Estimativa de Sobreajustamento (`MBIC`)

	BS			PL		
ϕ	$\delta = 0.5\sigma$	$\delta = 1.5\sigma$	$\delta = 3\sigma$	$\delta = 0.5\sigma$	$\delta = 1.5\sigma$	$\delta = 3\sigma$
0.2	0.1695	0.1860	0.0050	0.1980	0.2015	0.0245
0.5	0.4815	0.4180	0.0675	1.1785	1.2720	0.9910
0.8	0.9180	0.7920	0.4080	6.7635	6.7750	6.7045

A análise das tabelas mostra que o `AIC` é a penalização que piora mais os resultados dos métodos, particularmente no método `PL`, onde, em cada uma de 2000 simulações, observa-se uma média de quase 21 pontos fora do raio ótimo em cenários extremos. Em contraste, com a penalização `MBIC`, essa média reduz-se para aproximadamente 7 pontos. Já o método `BS`, cujo comportamento é semelhante ao do BP na tabela (4.6), apresenta em média cerca de 1 ponto fora do raio ótimo, comportamento consistente em quase todas as penalizações consideradas.

4.2.2 Resíduos com distribuição $t - Student$

Até ao momento, as análises basearam-se em situações em que os resíduos seguem uma distribuição normal. As tabelas seguintes apresentam cenários que se afastam das hipóteses clássicas de normalidade dos dados, considerando, em particular, a distribuição *t-Student*, que possui caudas mais pesadas do que a normal. Se $X \sim t_{(\nu)}$, então $\xi = \nu^{-1}$ representa o peso da cauda da distribuição t. A Figura 4.3 ilustra a

[3] O sobreajustamento ocorre quando o método identifica mais pontos de mudança do que realmente existem.

variação do peso da cauda direita da distribuição $t - student$ em função do número de graus de liberdade. À medida que $\nu \to 0$, ou para valores pequenos, a cauda da distribuição t, é mais pesada do que a normal.

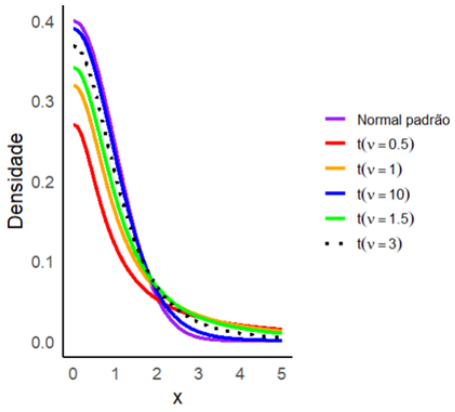


Figura 4.3: Variação de ν da distribuição t vs a normal

A tabela 4.11, contém dados de 2000 séries simuladas, com resíduos t_ν com $\nu = \{1.5,\ 2.5,\ 5\}$, $k = 50$ e $n = 200$.

Tabela 4.10: Taxa de acertos em cenários IID, considerando graus de liberdades 1.5, 2.5 e 5 da distribuição t

	$\nu = 1.5$			$\nu = 2.5$			$\nu = 5$		
δ	BS	PL	BP	BS	PL	BP	BS	PL	BP
0.5σ	4.35	41.55	1	8.30	18.35	6	9.50	8.90	9
1.5σ	23.45	63.85	10	61.40	64.40	66	80.40	78.50	80
3σ	56.70	94.00	54	91.85	96.15	92	98.80	98.90	98

Note que, quanto menor o número de graus de liberdade ν, mais pesadas se tornam as caudas da distribuição e menor é o poder de deteção dos métodos `BS` e `BP`. Já o método `PL` apresenta taxas mais elevadas em cenários de mudanças de pequena magnitude e caudas mais pesadas, aproximando-se, contudo, do comportamento dos restantes métodos à medida que aumenta o número de graus de liberdade e a magnitude das mudanças. À medida que ν aumenta, os resultados convergem para os obtidos no caso de resíduos normais.

Os gráficos da figura4.4 apresentam o desempenho dos algoritmos para valores de ν no intervalo [1, 10].

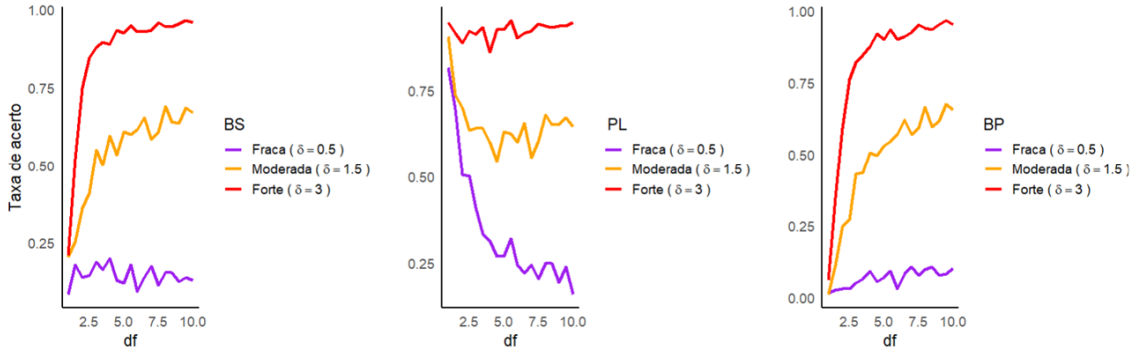


Figura 4.4: Variação do peso da Cauda

As tabelas 4.11, 4.12,4.13, mostram o comportamento dos métodos em cenários autocorrelacionados.

Tabela 4.11: Taxa de acertos-Distribuição t ($\phi = 0.2$)

	$\nu = 1.5$			$\nu = 2.5$			$\nu = 5$		
δ	BS	PL	BP	BS	PL	BP	BS	PL	BP
0.5σ	3.9	48.0	1	10.5	25.7	12	14.2	17.6	16
1.5σ	22.4	71.4	16	61.9	69.4	68	80.3	79.6	82
3σ	60.1	93.3	55	90.7	96.0	92	98.3	98.9	98

Tabela 4.12: Taxa de acertos-Distribuição t ($\phi = 0.5$)

	$\nu = 1.5$			$\nu = 2.5$			$\nu = 5$		
δ	BS	PL	BP	BS	PL	BP	BS	PL	BP
0.5σ	5.6	61.1	5	12.7	38.4	17	20.7	32.5	24
1.5σ	23.6	79.4	26	61.8	75.7	69	78.2	79.3	82
3σ	58.3	95.3	54	89.2	97.0	92	98.0	98.7	98

Tabela 4.13: Taxa de acertos-Distribuição t ($\phi = 0.8$)

	$\nu = 1.5$			$\nu = 2.5$			$\nu = 5$		
δ	BS	PL	BP	BS	PL	BP	BS	PL	BP
0.5σ	8.5	78.3	9	15.7	61.2	23	23.7	52.5	24
1.5σ	23.5	88.5	24	47.7	83.0	55	57.6	83.4	68
3σ	47.3	97.5	43	67.6	97.5	72	76.6	99.5	78

Tal como no caso da distribuição normal, o aumento da autocorrelação reduz o poder de deteção, mas esse impacto é agravado quando há falha da normalidade. O método `PL` tende a apresentar melhor desempenho em cenários mais extremos, mas

é o método que apresenta mais estimativas de mudanças fora do raio ótimo definido na secção 4.1, sobretudo na penalização `AIC`.

4.2.3 Efeito dos Outliers

Nesta secção, analisamos o impacto da presença de *outliers* na deteção de mudanças de estrutura em séries temporais, com ênfase nas mudanças na média. O objetivo é avaliar como a **posição** e a **intensidade** dos *outliers* afetam a deteção correta do ponto de mudança.

A categorização definida na Tabela 4.14 baseia-se na proximidade temporal dos *outliers* relativamente ao ponto real de mudança k. Assim, consideram-se *outliers próximos* aqueles que ocorrem na vizinhança imediata do ponto de mudança (± 10 unidades em torno de k). Os *outliers intermédios* correspondem a ocorrências situadas a uma distância moderada, entre 50 e 100 unidades antes ou depois de k. Finalmente, os *outliers distantes* referem-se a valores posicionados nas extremidades da série, isto é, nos primeiros 10% ou nos últimos 10% das observações.

Tabela 4.14: Categorias de posicionamento dos *outliers* em relação ao ponto real de mudança k.

Categoria	**Posição**
Próximos	$t_{\text{outlier}} \in [k-10, k+10]$
Intermédios	$t_{\text{outlier}} \in [k-100, k-50] \ \cup \ [k+50, k+100]$
Distantes	$t_{\text{outlier}} \in [1,\ 0.1\,n] \ \cup \ [0.9\,n,\ n]$

Além da posição, a intensidade dos *outliers* também é um fator crucial. Para quantificar esta intensidade, utilizamos o desvio padrão (σ) da série, expressando a magnitude dos *outliers* como múltiplos de σ, dada pela expressão .

$$x_{\text{outlier}} = \mu \pm c.\sigma, \quad c \in \mathbb{N}^+,$$

sendo μ a média da série e σ o desvio padrão.

Usando o cenário descrito na Secção 4.2, simulámos 2000 séries temporais e, através de um teste estatístico para a significância dos outliers, avaliámos para quais valores de c o ponto foi classificado como outlier, conforme descrito na tabela 4.15. Esses resultados serviram de base para a construção da Tabela 4.16, que apresenta o impacto da posição e da intensidade dos outliers na taxa de falsos positivos do algoritmo de segmentação.

Tabela 4.15: Valores-p do teste de Grubbs baseados em 2000 replicas, para diversos valores de c

c	**Teste de *Grubbs* (valor-p)**
1	0.98080
2	0.97844
3	0.71459
4	0.01868
5	0.00021
6	0.00000
7	0.00000
8	0.00000
9	0.00000
10	0.00000

Para a nossa metodologia, e de acordo com o teste de *Grubbs* Grubbs (1950), um ponto é considerado *outlier* quando $c \geq 4$, valor a partir do qual o valor-p do teste é estatisticamente significativo. Com base nesse critério, calculámos as taxas de falsos positivos (FP), definidas como o número de vezes que o algoritmo identifica *outliers* como pontos de mudança.

A Tabela 4.16 apresenta os resultados obtidos para diferentes posições e intensidades dos *outliers*, obtidos através de algoritmos de segmentação binária

Tabela 4.16: Taxas de FP (%) em 2000 réplicas (BS)

Posição	$\bar{X} + 4\sigma$	$\bar{X} + 6\sigma$	$\bar{X} + 8\sigma$	$\bar{X} + 10\sigma$	$\bar{X} + 12\sigma$
Próximos	22.30	28.60	30.55	58.90	82.90
Intermédios	0.10	0.65	3.80	8.55	19.85
Distantes	8.70	33.15	65.20	81.25	91.00

Tabela 4.17: Taxas de FP (%) em 2000 réplicas (PL)

Posição	$\bar{X} + 4\sigma$	$\bar{X} + 6\sigma$	$\bar{X} + 8\sigma$	$\bar{X} + 10\sigma$	$\bar{X} + 12\sigma$
Próximos	100	100	100	100	100
Intermédios	100	100	100	100	100
Distantes	100	100	100	100	100

Estas simulações foram realizadas em contextos bastante favoráveis, consideramos $n = 500$, $k = 250$, $\phi = 0.2$ e $\delta = 4\sigma$, uma mudança elevada e uma estrutura de dependência fraca ($\phi = 0.2$), situações nas quais se espera que os métodos não apresentem dificuldades e que a taxa de falsos positivos (FP) seja nula.

Para o método `BS` a proximidade dos *outliers* ao ponto de mudança aumenta de forma significativa as taxas de FP é ainda superior com o aumento da intensidade, onde chega a alcançar aproximadamente 83% para $c = 12$. Os *outliers* distantes, embora não próximos de k, podem induzir elevadas taxas de FP, chega a alcançar aproximadamente 65% ainda em $c = 8$ e 91% em $c = 12$. Os intermédios são os menos problemáticos, de acordo com os resultados do algoritmo, mas ainda assim tendem a distorcer a segmentação para intensidades elevadas, com aproximadamente 20% para $c = 12$.

Na presença de *outliers*, o método `PL` perde totalmente a sua eficiência, apresentando uma taxa de falsos positivos (FP) de 100%. Como era de esperar, os *outliers* tendem a distorcer a capacidade de deteção correta do ponto de mudança. Observa-se ainda que os métodos de divisão sucessiva (BS e `BP`) podem apresentar vantagens face aos métodos de otimização global, como o `PL`, neste tipo de cenário.

4.3 Mudanças na tendência da série

Ao contrário das mudanças na média, que afetam apenas o valor médio da série, as mudanças de tendência envolvem variações no padrão de crescimento ou decrescimento ao longo do tempo, conforme o modelo inicial proposto na secção 2.3.2.

Inicialmente, consideramos o cenário IID, sem autocorrelação, onde se espera um desempenho eficiente do método. Os resultados para séries de dimensão $n = 200$, 2000 réplicas e ordenadas na origem fixos em todos os cenários, estão apresentados na Tabela 4.18.

Tabela 4.18: Mudanças na tendência–Cenário IID

Cenário de inclinação	**Taxa de acerto (%)**
$\beta_1 = -0.3$ e $\beta_2 = 0.1$	99.00
$\beta_1 = 0$ e $\beta_2 = 0.1$	91.00
$\beta_1 = 0$ e $\beta_2 = 0.3$	99.00
$\beta_1 = 0$ e $\beta_2 = 0.5$	99.00
$\beta_1 = 0.1$ e $\beta_2 = -0.1$	98.00

O nosso interesse é avaliar se o método **BP** consegue detetar mudanças graduais no comportamento tendencial da série, mesmo na presença de autocorrelação nos resíduos, neste contexto simulamos um processo AR(1) com tendência em que os os $\varepsilon_t \sim N(0, 1)$.

Os resultados apresentados na Tabela 4.19 foram obtidos com base em 2000 réplicas de séries de dimensão $n = 200$, onde, os pontos de mudança na tendência da série temporal foram estimados utilizando o método de mínimos quadrados seg-

Tabela 4.19: Mudanças na tendência–Cenário dependentes

$\beta_{0,1} = \mathbf{0}$ e $\beta_{0,2} = \mathbf{1}$	**Taxa de acerto (%) por ϕ**		
	$\phi = 0.2$	$\phi = 0.5$	$\phi = 0.8$
$\beta_1 = 0$ e $\beta_2 = 0.1$	95.0	62.0	7.5
$\beta_1 = 0$ e $\beta_2 = 0.3$	96.0	68.5	14.5
$\beta_1 = 0$ e $\beta_2 = 0.5$	98.0	62.0	13.5
$\beta_1 = 0.1$ e $\beta_2 = -0.1$	95.0	68.0	15.5

mentados, implementado pela função `breakpoints()`[4] da biblioteca `strucchange` e fixamos os valores dos $\beta_{0,1}$ e $\beta_{0,2}$ e variamos os coeficientes de inclinação β_1 e β_2, para diferentes valores do coeficiente de autocorrelação ϕ.

O método apresenta um bom desempenho em cenários sem autocorrelação ou com dependência fraca, com uma média de 96% de acerto nos diferentes cenários de variação dos coeficientes β, conforme a tabela 4.19. Contudo, a sua eficácia diminui rapidamente à medida que a autocorrelação aumenta, mesmo quando a mudança na inclinação é significativa, registando-se uma taxa média de acerto em apenas 12,75% para $\phi = 0,8$.

4.4 Mudanças na forma da distribuição

Nesta secção, avaliamos a capacidade do método proposto em Kojadinovic and Naveau (2017) para identificar mudanças no parâmetro de forma (ξ) da distribuição `GEV`. Inicialmente, foram feitas um conjunto de simulações para se ter uma ideia da janela de tolerância (γ), conforme descrito na Secção 4.1.

Para esta análise, consideramos em contexto i.i.d, um cenário que evidencia uma clara mudança no peso da cauda, caracterizado por uma mudança de $\xi = -0.2$ para $\xi = 0.2$, o que representa a passagem de uma distribuição com cauda leve para outra com cauda pesada. Neste contexto, espera-se uma taxa de deteção do ponto exato onde ocorre a mudança no parâmetro de forma seja próxima de 100%, a qual corresponderia a um valor de $\gamma = 0$. A Tabela 4.20 apresenta os resultados obtidos nas 2000 simulações, considerando $n = 200$ e $k = n/2 = 100$.

Note-se que, mesmo em contexto i.i.d. e em condições em que a mudança é claramente visível, a taxa de deteção do ponto exato da mudança é praticamente nula e, mesmo para $\gamma = 10$, que corresponde ao intervalo $[90, 110]$, a taxa mantém-se abaixo de 50%.

Esta metodologia revela-se mais eficiente na simples identificação de que ocorreu uma mudança na distribuição, apresentando dificuldades na deteção exata do local

[4] A função `breakpoints` da biblioteca `strucchange` implementa o algoritmo de Bai e Perron, discutido na secção 3.2. Esta técnica, baseada em programação dinâmica, minimiza a Soma dos Quadrados dos Resíduos (RSS) para encontrar o conjunto de quebras ideal, utilizando o critério *Bayesian Information Criterion* (BIC) para selecionar o número de quebras. Anteriormente, aplicámos o método a mudanças na média, usando a terminologia `BP`-(Bai e Perron), mas, a seguir, focar-nos-emos na mudança de tendência, que constitui uma das principais funcionalidades deste algoritmo.

Tabela 4.20: Definição de raio ótimo — mudança no parâmetro forma.

γ	PMW	GPMW
0	0.0140	0.0115
1	0.0450	0.0455
2	0.0760	0.0770
3	0.1025	0.1115
4	0.1320	0.1445
5	0.1780	0.1835
6	0.2190	0.2260
7	0.2740	0.2735
8	0.3405	0.3325
9	0.4135	0.3915
10	0.4635	0.4395

da alteração. Por este motivo, no contexto autocorrelacionado, abordou-se apenas o nível empírico e a potência do teste, com o objetivo de avaliar o comportamento dos métodos PWM e GPWM dentro da estatística CUSUM na deteção de mudanças no parâmetro de forma ξ, considerando diferentes intensidades de dependência serial.

Foram consideradas séries temporais de tamanho $n = 200$, com três níveis de dependência serial: fraca, $\phi = 0.2$, moderada, $\phi = 0.5$ e forte, $\phi = 0.8$. Foi também considerando o caso $\phi = 0$, que corresponde a assumir independência, e para o qual esta metodologia foi desenvolvida.

Tal como apresentado na Secção 2.3.3, e considerando apenas o caso em que temos uma mudança no parâmetro de forma. Para cada configuração, foram realizadas 2000 repetições de Monte Carlo. A Tabela 4.21 apresenta o nível empírico dos testes PWM e GPWM para os diferentes cenários de dependência, i.e., a proporção de rejeições de H_0 quando não há mudança de ξ.

Tabela 4.21: Nível empírico dos testes PWM e GPWM para diferentes níveis de dependência, ao nível de significância de 5%.

ϕ	$\xi = -\mathbf{0.2}$		$\xi = \mathbf{0}$		$\xi = \mathbf{0.2}$	
	PWM	GPWM	PWM	GPWM	PWM	GPWM
0.0	0.0485	0.0175	0.0345	0.0280	0.0260	0.0355
0.2	0.0430	0.0145	0.0380	0.0210	0.0330	0.0290
0.5	0.0850	0.0345	0.0720	0.0440	0.0660	0.0625
0.8	0.3630	0.2005	0.3310	0.2650	0.3700	0.3240

De acordo com os resultados da Tabela 4.21, o método parece não funcionar bem a 5%, já que não se observa padrões de valores perto de 0.05 mesmo em condições de dependência fraca e à medida que aumenta o peso da cauda e a dependência serial, os métodos tendem a distorcer os resultados, resultando na perda de controlo sobre o nível de significância empírico. Ou seja o teste relata mudança como estatisticamente significativas, mesmo na ausência de uma alteração real.

Para análise da potência do teste que corresponde à proporção de rejeições de H_0 quando há mudança de ξ, consideramos os cenários descritos na Tabela 4.22 .

Tabela 4.22: Potência dos testes PWM e GPWM para diferentes níveis de dependência com mudança de ξ, ao nível de significância de 5%.

ϕ	ξ_{antes}	ξ_{depois}	**PWM**	**GPWM**
0.0	0	0.2	0.1520	0.1510
0.2	0	0.2	0.1575	0.1500
0.5	0	0.2	0.2140	0.1935
0.8	0	0.2	0.5355	0.4552
0.0	0.2	0.6	0.3125	0.478
0.2	0.2	0.6	0.3180	0.500
0.5	0.2	0.6	0.4410	0.527
0.8	0.2	0.6	0.7770	0.650
0.0	-0.3	0	0.2375	0.240
0.2	-0.3	0	0.2370	0.243
0.5	-0.3	0	0.3390	0.328
0.8	-0.3	0	0.7115	0.674
0.0	-0.2	0.2	0.5905	0.4950
0.2	-0.2	0.2	0.6210	0.5155
0.5	-0.2	0.2	0.7060	0.6270
0.8	-0.2	0.2	0.9100	0.8565
0.0	-0.1	-0.3	0.2585	0.0930
0.2	-0.1	-0.3	0.2740	0.0805
0.5	-0.1	-0.3	0.4060	0.1580
0.8	-0.1	-0.3	0.7320	0.4405

Pela Tabela 4.22, constata-se que à medida que ϕ aumenta (mais dependência serial), os testes tornam-se mais potentes, sobretudo o PWM, que responde de forma mais acentuada à estrutura de dependência. Com dados independentes, os testes têm comportamento semelhante e potência limitada, mas sob dependência moderada ou forte, o PWM é claramente o mais eficaz para detetar mudanças no parâmetro de forma ξ da GEV. Por outro lado, observamos que o teste PWM é sensível ao "engrossar" da cauda mesmo com o aumento de dependência serial. As mudanças estruturais de cauda leve (limitada) para cauda pesada são as mais fáceis de detetar pelos testes. No caso em que as caudas ficam mais leves ($\xi_{\text{antes}} = -0.1$; $\xi_{\text{depois}} = -0.3$), os métodos têm mais dificuldade em detetar estas mudanças, com especial destaque para o GPWM, com potências de teste bastante inferiores às do PWM.

Capítulo 5

Conclusões e trabalho futuro

A presente dissertação analisou de forma crítica os métodos clássicos de deteção de mudanças de estrutura em séries temporais, avaliando o desempenho destes face à violação dos seus pressupostos. As simulações de Monte Carlo validaram as abordagens propostas e destacaram os limites de aplicabilidade das mesmas em cenários mais complexos.

Os resultados indicam que a forte dependência entre as observações reduz a capacidade de deteção, fazendo com que alguns métodos detetem pontos de mudança onde não existem. Entre os métodos de divisão sucessiva, como a segmentação binária (`BS`) e o algoritmo de Bai Perron `BP`, apresentam bom desempenho em cenários i.i.d., mas perdem potência em contextos autocorrelacionados, especialmente na análise de mudanças na tendência. O algoritmo de otimização global `PELT` (`PL`) tende a acertar nos pontos de quebra na média , mas apresenta muitas quebras espúrias e é extremamente sensível a valores atípicos (*outliers*). As taxas de acerto nas mudança de nível são relativamente baixas quando a magnitude da variação é pequena, independentemente do método utilizado. Resultados semelhantes são obtidos em dados não normais, sendo que quanto maior o afastamento da normalidade, menor é a taxa de acerto.

Por fim, o tamanho da amostra não se mostra um fator determinante para a eficiência dos métodos; nas situações analisadas, as diferenças de desempenho entre tamanhos amostrais foram pequenas.

No âmbito dos testes CUSUM, adaptados ao contexto de mudança no parâmetro de forma, ξ, da distribuição GEV, o método *Probability-Weighted Moments* `PWM` mostrou-se mais potente para detetar mudanças no parâmetro ξ, especialmente para caudas pesadas, enquanto que *Generalized Probability Weighted Moments* `GPWM` apresenta menor potência, sobretudo em mudanças para caudas leves. Contudo, ambos revelam limitações na determinação precisa do instante da mudança e tendem a detetar falsos alarmes sob forte dependência serial ou em distribuições com caudas mais pesadas.

A obtenção de previsões precisas em séries temporais que contêm mudanças de estruturas requer a integração de técnicas de estimação robusta e metodologias adaptativas à cauda, capazes de lidar com dependência serial e *outliers*, bem como a deteção de mudanças de estrutura (que podem ser por vezes pouco percetíveis). Com base nos resultados do teste PWM na deteção de alterações no parâmetro de forma da distribuição GEV, pretende-se generalizar o teste a cenários com dependência

serial e desenvolver medidas que permitam estimar com maior precisão o ponto de quebra. Os resultados empíricos obtidos reforçam esta necessidade e motiva linhas futuras de investigação que devem focar o estudo e desenvolvimento de métodos mais recentes e robustos para lidar com todos estes desafios.

Bibliografia

Andrews, D. W. (1993). Tests for parameter instability and structural change with unknown change point. *Econometrica: Journal of the Econometric Society*, pages 821–856.

Andrews, D. W. and Ploberger, W. (1994). Optimal tests when a nuisance parameter is present only under the alternative. *Econometrica: Journal of the Econometric Society*, pages 1383–1414.

Andrews, D. W. K. (1991). Heteroskedasticity and autocorrelation consistent covariance matrix estimation. *Econometrica*, 59(3):817–858.

Aue, A. and Horváth, L. (2013). Structural breaks in time series. *Journal of Time Series Analysis*, 34(1):1–16.

Bai, J. (1997). Estimation of a change point in multiple regression models. *Review of Economics and Statistics*, 79(4):551–563.

Bai, J. and Perron, P. (2003a). Computation and analysis of multiple structural change models. *Journal of applied econometrics*, 18(1):1–22.

Bai, J. and Perron, P. (2003b). Critical values for multiple structural change tests. *The Econometrics Journal*, 6(1):72–78.

Barry, D. and Hartigan, J. A. (1993). A bayesian analysis for change point problems. *Journal of the American Statistical Association*, 88(421):309–319.

Basseville, M., Nikiforov, I. V., et al. (1993). *Detection of abrupt changes: theory and application*, volume 104. Prentice hall Englewood Cliffs.

Beirlant, J., Goegebeur, Y., Segers, J., and Teugels, J. L. (2006). *Statistics of extremes: theory and applications*. John Wiley & Sons.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

Box, G. E. and Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 26(2):211–243.

Brown, R. L., Durbin, J., and Evans, J. M. (1975). Techniques for testing the constancy of regression relationships over time. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 37(2):149–163.

Bücher, A. and Zhou, C. (2021). A horse race between the block maxima method and the peak–over–threshold approach. *Statistical Science*, 36(3):360–378.

Casini, A. and Perron, P. (2018). Structural breaks in time series. *arXiv preprint arXiv:1805.03807*.

Chen, J. and Gupta, A. K. (1997). Testing and locating variance changepoints with application to stock prices. *Journal of the American Statistical association*, 92(438):739–747.

Chen, J., Gupta, A. K., and Gupta, A. (2000). *Parametric statistical change point analysis*, volume 192. Springer.

Chow, G. C. (1960). Tests of equality between sets of coefficients in two linear regressions. *Econometrica: Journal of the Econometric Society*, pages 591–605.

Chu, C.-S. J., Hornik, K., and Kuan, C.-M. (1995). The moving-estimates test for parameter stability. *Econometric Theory*, 11(4):699–720.

Cordeiro, C. and Neves, M. M. (2014). Forecast intervals with boot.expos. In *Proceedings of the International Symposium on Forecasting*, Roterdão, Países Baixos.

Cordeiro, C. and Neves, M. M. (2019). Modelação e previsão de extremos em séries temporais (boot.expos). In *XXIII Congresso da Sociedade Portuguesa de Estatística*, Lisboa, Portugal.

Cryer, J. D. (1986). *Time series analysis*, volume 286. Duxbury Press Boston.

Cryer, J. D. and Chan, K.-S. (2008). *Time series analysis: with applications in R*. Springer.

Diebolt, J., Guillou, A., and Rached, I. (2007). Approximation of the distribution of excesses through a generalized probability-weighted moments method. *Journal of Statistical Planning and Inference*, 137(3):841–857.

Embrechts, P., Klüppelberg, C., and Mikosch, T. (1997). *Modelling Extremal Events for Insurance and Finance*. Stochastic Modelling and Applied Probability. Springer.

Fearnhead, P. and Rigaill, G. (2019). Changepoint detection in the presence of outliers. *Journal of the American Statistical Association*, 114(525):169–183.

Fryzlewicz, P. (2014). Wild binary segmentation for multiple change-point detection.

Greenwood, J. A., Landwehr, J. M., Matalas, N. C., and Wallis, J. R. (1979). Probability weighted moments: definition and relation to parameters of several distributions expressable in inverse form. *Water resources research*, 15(5):1049–1054.

Grubbs, F. E. (1950). Sample criteria for testing outlying observations. *The Annals of Mathematical Statistics*, 21(1):27–58.

Hamilton, J. D. (2020). *Time series analysis*. Princeton university press.

Hinkley, D. V. (1970). Inference about the change-point in a sequence of random variables.

Horváth, L. (1993). The maximum likelihood method for testing changes in the parameters of normal observations. *The Annals of statistics*, pages 671–680.

Horváth, L. and Rice, G. (2024). *Change Point Analysis for Time Series*. Springer Nature, Cham.

Hosking, J. R. M., Wallis, J. R., and Wood, E. F. (1985). Estimation of the generalized extreme-value distribution by the method of probability-weighted moments. *Technometrics*, 27(3):251–261.

Jenkins, G. M. and Box, G. E. (1976). Time series analysis: forecasting and control. *(No Title)*.

Jenkinson, A. F. (1955). The frequency distribution of the annual maximum (or minimum) values of meteorological elements. *Quarterly Journal of the Royal meteorological society*, 81(348):158–171.

Killick, R. and Eckley, I. A. (2014). changepoint: An r package for changepoint analysis. *Journal of statistical software*, 58:1–19.

Killick, R., Eckley, I. A., Ewans, K., and Jonathan, P. (2010). Detection of changes in variance of oceanographic time-series using changepoint analysis. *Ocean Engineering*, 37(13):1120–1126.

Killick, R., Fearnhead, P., and Eckley, I. A. (2012). Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598.

Kleiber, A. (2002). An {R} package for testing for structural change in linear regression models. *An {R} Package for Testing for Structural*, 7(2).

Kojadinovic, I. and Naveau, P. (2017). Detecting distributional changes in samples of independent block maxima using probability weighted moments. *Extremes*, 20:417–450.

Landwehr, J. M., Matalas, N., and Wallis, J. (1979). Probability weighted moments compared with some traditional techniques in estimating gumbel parameters and quantiles. *Water resources research*, 15(5):1055–1064.

Leadbetter, M. R., Lindgren, G., and Rootzén, H. (1983). *Extremes and Related Properties of Random Sequences and Processes*. Springer Series in Statistics. Springer.

Lumley, T. and Heagerty, P. J. (1999). Weighted estimating equations and correlation structure. *Journal of the American Statistical Association*, 94(446):430–440.

Ma, L., Grant, A. J., and Sofronov, G. (2020). Multiple change point detection and validation in autoregressive time series data. *Statistical Papers*, 61:1507–1528.

Montgomery, D. C., Jennings, C. L., and Kulahci, M. (2008). *Introduction to Time Series Analysis and Forecasting*. Wiley-Blackwell, Hoboken, NJ, 1 edition.

Nelson, C. R. and Plosser, C. R. (1982). Trends and random walks in macroeconmic time series: some evidence and implications. *Journal of monetary economics*, 10(2):139–162.

Page, E. S. (1954). Continuous inspection schemes. *Biometrika*, 41(1/2):100–115.

Page, E. S. (1955). A test for a change in a parameter occurring at an unknown point. *Biometrika*, 42(3/4):523–527.

Salman, Y., Hoayek, A., and Batton-Hubert, M. (2024). New algorithm for weak changes detection with application to welding electrical signals.

Scott, A. J. and Knott, M. (1974). A cluster analysis method for grouping means in the analysis of variance. *Biometrics*, pages 507–512.

Scotto, M. G., Weiss, C. H., and Gouveia, S. (2015). Clustering of time series by extremal dependence. *REVSTAT Statistical Journal*, 13(3):197–216.

Sen, A. and Srivastava, M. S. (1975). On tests for detecting change in mean. *The Annals of statistics*, pages 98–108.

Shao, X. and Zhang, X. (2010). Testing for change points in time series. *Journal of the American Statistical Association*, 105(491):1228–1240.

Truong, C., Oudre, L., and Vayatis, N. (2020). Selective review of offline change point detection methods. *Signal Processing*, 167:107299.

Von Mises, R. (1936). Wahrscheinlichkeit statistik und wahrheit: Einführung in die neue wahrscheinlichkeitslehre und ihre anwendung.

Weisberg, S. (2001). Yeo-johnson power transformations. *Department of Applied Statistics, University of Minnesota. Retrieved June*, 1:2003.

Yao, Y.-C. and Davis, R. A. (1986). The asymptotic behavior of the likelihood ratio statistic for testing a shift in mean in a sequence of independent normal variates. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 339–353.

Zeileis, A., Leisch, F., Hornik, K., and Kleiber, C. (2002a). strucchange: An r package for testing for structural change in linear regression models. *Journal of statistical software*, 7:1–38.

Zeileis, A., Leisch, F., Hornik, K., and Kleiber, C. (2002b). strucchange: An r package for testing for structural change in linear regression models. *Journal of statistical software*, 7:1–38.

Zhang, N. R. and Siegmund, D. O. (2007). A modified bayes information criterion with applications to the analysis of comparative genomic hybridization data. *Biometrics*, 63(1):22–32.

Anexo

Script Software R Cram

Bibliotecas Principais

```
packages <- c("changepoint","evd", "gridExtra", "CPAT","nortest","
    robseg",
              "dplyr", "MLmetrics", "strucchange", "ggplot2", "
                 fUnitRoots",
              "tidyverse", "lubridate", "pROC", "forecast", "
                 tseries", "stR","bcpa","FinTS","npcp", "urca","
                 npcp")

lapply(packages, library, character.only = TRUE)

# An lises preliminares
set.seed(125)
phi1<-0.4
n=50
innov<-rt(n,1.5)
innov1 <- rexp(n, rate = 1)

# Simular a s rie
yt<-arima.sim(n = n, model = list(ar = phi1))
ts.plot(yt)
shapiro.test(yt) # normal
lillie.test(yt)
qqnorm(yt, xlab="QuantisTeoricos", ylab="QuantisEmpricos")
qqline(yt, col=2)

yt<-arima.sim(n = n, model = list(ar = phi1), innov = innov) # Yt ~
     t, 2 graus de liberdade
ts.plot(yt)
hist(yt)
shapiro.test(yt) # t
lillie.test(yt)
qqnorm(yt, xlab="QuantisTeoricos", ylab="QuantisEmpricos")
qqline(yt, col=2)

yt<-arima.sim(n = n, model = list(ar = phi1), innov = innov1) # Yt
    ~ Exp(1)
ts.plot(yt)
hist(yt)
shapiro.test(yt) # exponencial
lillie.test(yt)
qqnorm(yt, xlab="QuantisTeoricos", ylab="QuantisEmpricos")
qqline(yt, col=2)

#################################################################
data("Nile")
```

```r
df <- data.frame(
  year = time(Nile),
  flow = as.numeric(Nile)
)

ggplot(df, aes(x = year, y = flow)) +
  geom_line(color = "blue", size = 1) +
  labs(title = "S rie Temporal do Rio Nilo",
       x = "Ano",
       y = "Fluxo") +
  theme_minimal() +
  theme(panel.grid = element_blank())

length(Nile)
result <- cpt.mean(Nile, method = "BinSeg", penalty = "BIC", Q=1)

cpts(result)
plot(result, main = "")
############################################################
#Gr ficos
set.seed(42)

n <- 500
time <- 1:n

# --------- 1) AR(1) estacion rio (phi = 0.6) ----------
phi1 <- 0.6
eps <- rnorm(n)
x <- numeric(n)

for (t in 2:n) {
  x[t] <- phi1 * x[t-1] + eps[t]
}

# Criar data frame
df <- data.frame(time = time, value = x)

# Plot com ggplot2 sem grade e com eixos
g1<-ggplot(df, aes(x = time, y = value)) +
  geom_line(size=1, col="red") +
  geom_hline(yintercept = 0, color = "black", linetype = "dashed",
     size=1) +
  labs(
    title = "Processo AR(1) estacion rio (   = 0.6)",
    x = "t",
    y = "y"
  ) +
  theme_classic() +  # remove grade e mant m eixos
  theme(
    axis.line = element_line(),   # enfatiza os eixos
    panel.grid = element_blank()  # garante que n o h  grid
  )

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


set.seed(123)

n <- 600
```

```
time <- 1:n
eps <- rnorm(n)

y <- numeric(n)

# Definir os segmentos
seg1 <- 1:(n/3)
seg2 <- (n/3 + 1):(2*n/3)
seg3 <- (2*n/3 + 1):n

# Par metros para cada segmento
phi1 <- 0.3; mu1 <- 0
phi2 <- 0.3; mu2 <- 1
phi3 <- 0.3; mu3 <- -1

# Simular segmento 1
for (t in seg1[-1]) {
  y[t] <- mu1 + phi1 * (y[t-1] - mu1) + eps[t]
}

# Simular segmento 2
y[seg2[1]] <- mu2
for (t in seg2[-1]) {
  y[t] <- mu2 + phi2 * (y[t-1] - mu2) + eps[t]
}

# Simular segmento 3
y[seg3[1]] <- mu3
for (t in seg3[-1]) {
  y[t] <- mu3 + phi3 * (y[t-1] - mu3) + eps[t]
}

# Criar dataframe para ggplot
df <- data.frame(time = time, value = y)

# Plot com ggplot2
g2<-ggplot(df, aes(x = time, y = value)) +
  geom_line(size=1,col="blue") +
  # linhas verticais marcando os regimes
  geom_vline(xintercept = c(n/3, 2*n/3), linetype = "dashed") +
  # linhas horizontais dos n veis m dios
  geom_hline(yintercept = mu1, color = "orange", linetype = "dashed
      ",size=1) +
  geom_hline(yintercept = mu2, color = "green", linetype = "dashed"
      ,size=1) +
  geom_hline(yintercept = mu3, color = "black", linetype = "dashed"
      ,size=1) +
  labs(
    title = "",
    x = "t",
    y = "y"
  ) +
  theme_classic() +
  theme(
    axis.line = element_line(),
    panel.grid = element_blank()
  )
#juntar os dois gr ficos
library(patchwork)
```

```
g1+g2


set.seed(123)

n <- 500
time <- 1:n

# --------- AR(1) com vari ncia  n  o constante ----------
phi1 <- 0.6
beta <- 0.02          # pequena tend ncia linear
eps <- rnorm(n)
x_het <- numeric(n)

for (t in 2:n) {
  # Ru do com vari ncia crescente
  sigma_t <- 0.5 + 0.005 * t       # vari ncia aumenta com o tempo
  x_het[t] <- phi1 * x_het[t-1] + beta + eps[t] * sigma_t
}

# Criar data frame
df_het <- data.frame(time = time, value = x_het)

# Plot com ggplot2
g5 <- ggplot(df_het, aes(x = time, y = value)) +
  geom_line(size = 1, col = "purple") +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed",
     size = 1) +
  labs(
    title = "Processo AR(1) com vari ncia  n  o constante",
    x = "Tempo",
    y = "Valor"
  ) +
  theme_classic() +
  theme(
    axis.line = element_line(),
    panel.grid = element_blank()
  )

# Mostrar gr fico
print(g5)

################33
library(ggplot2)

set.seed(42)

n <- 500
time <- 1:n

# --------- AR(1) com tend ncia ----------
phi1 <- 0.6
beta <- 0.05         # inclina  o da tend ncia linear
eps <- rnorm(n, mean = 0, sd = 5)
x_trend <- numeric(n)

for (t in 2:n) {
  x_trend[t] <- phi1 * x_trend[t-1] + beta * t + eps[t]
}
```

```r
# Criar data frame
df_trend <- data.frame(time = time, value = x_trend)

# Plot com ggplot2
g4 <- ggplot(df_trend, aes(x = time, y = value)) +
  geom_line(size = 1, col = "red") +
  labs(
    title = "I",
    x = "t",
    y = "y"
  ) +
  theme_classic() +
  theme(
    axis.line = element_line(),
    panel.grid = element_blank()
  )

# Mostrar gr fico
print(g4)
##############333
library(ggplot2)

set.seed(42)

n <- 500
time <- 1:n

# --------- AR(1) com vari ncia n o constante ----------
phi1 <- 0.6
beta <- 0.02          # pequena tend ncia linear
eps <- rnorm(n)
x_het <- numeric(n)

for (t in 2:n) {
  # Ru do com vari ncia crescente
  sigma_t <- 0.5 + 0.005 * t       # vari ncia aumenta com o tempo
  x_het[t] <- phi1 * x_het[t-1] + beta + eps[t] * sigma_t
}

# Criar data frame
df_het <- data.frame(time = time, value = x_het)

# Plot com ggplot2
g5 <- ggplot(df_het, aes(x = time, y = value)) +
  geom_line(size = 1, col = "blue") +

  labs(
    title = "II",
    x = "t",
    y = "y"
  ) +
  theme_classic() +
  theme(
    axis.line = element_line(),
    panel.grid = element_blank()
  )

# Mostrar gr fico
```

```r
print(g5)
library(patchwork)
g4+g5

##########################################3
library(ggplot2)

set.seed(123)

n <- 200
change_point <- 100

# ---- AR(1) ----
phi <- 0.7
ar1 <- arima.sim(n = n, list(ar = phi))
ar1[change_point:n] <- ar1[change_point:n] + 3
df_ar <- data.frame(time = 1:n, value = ar1, tipo = "AR(1)")

# ---- ARMA(1,1) ----
phi <- 0.7
theta <- 0.5
arma11 <- arima.sim(n = n, list(ar = phi, ma = theta))
arma11[change_point:n] <- arma11[change_point:n] + 3
df_arma <- data.frame(time = 1:n, value = arma11, tipo = "ARMA(1,1)
    ")

# ---- ARIMA(1,1,1) ----
phi <- 0.7
theta <- 0.5
arima111 <- arima.sim(n = n, list(order = c(1,1,1), ar = phi, ma =
    theta))
arima111[change_point:n] <- arima111[change_point:n] + 3
df_arima <- data.frame(time = 1:n, value = arima111, tipo = "ARIMA
    (1,1,1)")

# ---- Combinar e plotar ----
df <- rbind(df_ar, df_arma, df_arima)

ggplot(df, aes(x = time, y = value, color = tipo)) +
  geom_line(size = 1) +
  geom_vline(xintercept = change_point, linetype = "dashed", color
      = "red") +
  labs(
    title = "Compara  o da detec  o de mudan a de m dia: AR
        (1), ARMA(1,1), ARIMA(1,1,1)",
    x = "Tempo",
    y = "Valor",
    color = "Processo"
  ) +
  theme_classic()

##########################################3
# Defini  o dos par metros do AR(1)
library(forecast)   # para autoplot
library(ggplot2)

# Par metros do AR(1)
phi <- 0.6
n <- 500
```

```r
# Simula  o
set.seed(42)
x <- arima.sim(model = list(ar = phi), n = n,sd=2)+3

# FAC com autoplot
g1<-autoplot(acf(x, plot = FALSE),lwd=1,col="blue") +
  ggtitle("Fun  o de Autocorrela  o (FAC) - AR(1) simulado")+
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  theme(
    axis.line = element_line(),
    panel.grid = element_blank()
  )
# FACP com autoplot
g2<-autoplot(pacf(x, plot = FALSE),lwd=1.5,col="orange")+
  ggtitle("Fun  o de Autocorrela  o Parcial (FACP) - AR(1)
     simulado")+
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  geom_segment(aes(xend = lag, yend = 0), size = 3) +
  theme(
    axis.line = element_line(),
    panel.grid = element_blank()
  )
library(patchwork)
g1+g2

# Simular MA(1)
set.seed(42)
ma1 <- arima.sim(n = n, list(ma = 0.7), sd=2)+3

# Simular MA(3)
set.seed(42)
ma3 <- arima.sim(n = n, list(ma = c(0.5, -0.3, 0.4)),sd=2)+5

# Plot das s ries temporais
g1<-autoplot(acf(ma1, plot = FALSE),lwd=1.5,col="blue")+
  ggtitle("ACF- MA(1)")+
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  geom_segment(aes(xend = lag, yend = 0), size = 3) +
  theme(
    axis.line = element_line(),
    panel.grid = element_blank()
  )
# FACP com autoplot
g2<-autoplot(acf(ma3, plot = FALSE),lwd=1.5,col="orange")+
  ggtitle("ACF- MA(3)")+
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  geom_segment(aes(xend = lag, yend = 0), size = 3) +
  theme(
    axis.line = element_line(),
    panel.grid = element_blank()
  )
g1+g2

#####################3333
```

```r
library(forecast)
library(ggplot2)
library(patchwork) # para juntar os gr ficos

n <- 200

# Simular ARMA(1,1) com AR=0.6, MA=0.5
set.seed(42)
arma11 <- arima.sim(n = n, list(ar = 0.6, ma = 0.5),sd=2)+4

# Simular ARMA(2,2) com AR=c(0.5,-0.3), MA=c(0.4,0.2)
set.seed(42)
arma22 <- arima.sim(n = n, list(ar = c(0.5, -0.3), ma = c(0.4, 0.2)
    ),sd=2)+2

# FAC ARMA(1,1)
g1 <- autoplot(acf(arma11, plot = FALSE), lwd=1.5, col="blue") +
  ggtitle("ACF - ARMA(1,1)") +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  geom_segment(aes(xend = lag, yend = 0), size = 3) +
  theme(axis.line = element_line(), panel.grid = element_blank())

# FAC ARMA(2,2)
g2 <- autoplot(acf(arma22, plot = FALSE), lwd=1.5, col="orange") +
  ggtitle("ACF - ARMA(2,2)") +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  geom_segment(aes(xend = lag, yend = 0), size = 3) +
  theme(axis.line = element_line(), panel.grid = element_blank())

# Juntar gr ficos
g1 + g2


########################################3333
library(forecast)
library(ggplot2)
library(patchwork) # para juntar os gr ficos

n <- 200

# Simular ARMA(1,1) com AR=0.6, MA=0.5
set.seed(42)
arma11 <- arima.sim(n = n, list(ar = 0.6, ma = 0.5),sd=2)+2

# Simular ARMA(2,2) com AR=c(0.5,-0.3), MA=c(0.4,0.2)
set.seed(42)
arma22 <- arima.sim(n = n, list(ar = c(0.5, -0.3), ma = c(0.4, 0.2)
    ),sd=2)+3

# FAC ARMA(1,1)
g3 <- autoplot(pacf(arma11, plot = FALSE), lwd=1.5, col="blue") +
  ggtitle("PACF - ARMA(1,1)") +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  geom_segment(aes(xend = lag, yend = 0), size = 3) +
  theme(axis.line = element_line(), panel.grid = element_blank())
```

```r
# FAC ARMA(2,2)
g4 <- autoplot(pacf(arma22, plot = FALSE), lwd=1.5, col="orange") +
  ggtitle("PACF - ARMA(2,2)") +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  geom_segment(aes(xend = lag, yend = 0), size = 3) +
  theme(axis.line = element_line(), panel.grid = element_blank())

# Juntar gr ficos
par(mfrow=c(2,2))
g1+g2+g3+g4

####################333333
# par metros
n <- 100
lambda <- 50

# gera s rie temporal
t <- 1:n
y <- c(rnorm(lambda, mean = 10, sd = 1),   # antes da mudan a
       rnorm(n - lambda, mean = 13, sd = 1)) # depois da mudan a

# cria dataframe
df <- data.frame(
  tempo = t,
  valor = y
)

# plota
ggplot(df, aes(x = tempo, y = valor)) +
  geom_line(color = "black", linewidth = 1.5) +
  geom_vline(xintercept = lambda, linetype = "dashed", color = "red
     ",size=1) +
  labs(
    title = " k = 50",
    x = "t",
    y = "y"
  ) +
  theme_minimal()+
  theme(
    panel.grid = element_blank(),     # remove a grelha
    axis.line = element_line(color = "black", linewidth = 0.8), #
       adiciona linhas dos eixos
    axis.ticks = element_line(color = "black")                 #
       adiciona ticks
  )
 #secc   o: 2.3
#
   ##############################################################################

set.seed(123)

# par metros
n <- 100
lambda <- 50

# gera s rie temporal com mudan a de tend ncia
t <- 1:n
y <- c( 2 + 0.3 * (1:lambda) + rnorm(lambda, sd = 5),
```

```
        7 + 0.6 * (1:(n - lambda)) + rnorm(n - lambda, sd = 5) )

# cria dataframe
df <- data.frame(
  tempo = t,
  valor = y
)

# estima breakpoints
bp <- breakpoints(y ~ t)
fitted_vals <- fitted(bp)

df$fitted <- fitted_vals
library(ggplot2)
ggplot(df, aes(x = tempo)) +
  geom_line(aes(y = valor), color = "black", linewidth = 1.2) +
  geom_line(aes(y = fitted), color = "blue", linewidth = 1.2) +
  geom_vline(xintercept = bp$breakpoints, linetype = "dashed",
     color = "red",size=1) +
  labs(
    title = "k=50",
    x = "t",
    y = "y"
  ) +
  theme_minimal()+
  theme(
    panel.grid = element_blank(),     # remove a grelha
    axis.line = element_line(color = "black", linewidth = 0.8), #
       adiciona linhas dos eixos
    axis.ticks = element_line(color = "black")                  #
       adiciona ticks
  )

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



library(evd)

set.seed(123)
n <- 100

# --- Cen rio I: xi constante = 0.2 ---
x1 <- rgev(n, loc = 0, scale = 1, shape = 0.2)
df1 <- data.frame(
  t = 1:n,
  valor = x1
)

# --- Cen rio II: xi muda em t = 50 ---
x2 <- c(
  rgev(50, loc = 0, scale = 1, shape = 0.2),
  rgev(50, loc = 0, scale = 1, shape = 0.5)
)
df2 <- data.frame(
  t = 1:n,
  valor = x2
)
```

```r
# --- Gr fico 1: Cen rio I ---
g1 <- ggplot(df1, aes(x = t, y = valor)) +
  geom_line(color = "black",size=1) +
  labs(
    title = "I",
    x = "t",
    y = "y"
  ) +
  theme_minimal(base_size = 13) +
  theme(
    panel.grid = element_blank(),
    axis.line = element_line(color = "black")
  )

# --- Gr fico 2: Cen rio II ---
g2 <- ggplot(df2, aes(x = t, y = valor)) +
  geom_line(color = "blue",size=1) +
  geom_vline(xintercept = 50, color = "red", linetype = "dashed",
     linewidth = 1) +
  labs(
    title = "II",
    x = "t",
    y = "y"
  ) +
  theme_minimal(base_size = 13) +
  theme(
    panel.grid = element_blank(),
    axis.line = element_line(color = "black")
  )
# --- Exibe gr ficos lado ao lado
library(gridExtra)
grid.arrange(g1, g2, ncol = 2)

####################3

library(patchwork) # para combinar gr ficos

# --- Fun   o CUSUM ---
cusum_stat <- function(y) {
  n <- length(y)
  stats <- numeric(n - 1)
  for (lambda in 1:(n - 1)) {
    mean1 <- mean(y[1:lambda])
    mean2 <- mean(y[(lambda + 1):n])
    stats[lambda] <- sqrt((lambda * (n - lambda)) / n) * abs(mean1
       - mean2)
  }
  return(stats)
}

# --- Simula   o de dados ---
set.seed(123)
y <- c(rnorm(50, mean = 0), rnorm(50, mean = 3))
cusum_vals <- cusum_stat(y)

df_y <- data.frame(t = 1:length(y), valor = y)
df_cusum <- data.frame(lambda = 1:(length(y)-1), C = cusum_vals)
```

```r
# --- Gr fico 1: S rie temporal ---
g1 <- ggplot(df_y, aes(x = t, y = valor)) +
  geom_line(color = "blue", linewidth = 1) +

  labs(title = "S rie temporal", x = "X", y = "y") +
  theme_minimal(base_size = 13) +
  theme(
    panel.grid = element_blank(),
    axis.line = element_line(color = "black")
  )

# --- Gr fico 2: Estat stica CUSUM ---
g2 <- ggplot(df_cusum, aes(x = lambda, y = C)) +
  geom_line(color = "darkred", linewidth = 1) +
  geom_vline(xintercept = which.max(cusum_vals), color = "red",
     linetype = "dashed") +
  labs(title = "Estat stica CUSUM", x = "  ", y = expression(C[
     lambda])) +
  theme_minimal(base_size = 13) +
  theme(
    panel.grid = element_blank(),
    axis.line = element_line(color = "black")
  )

# --- Lado a lado ---
g1 + g2
#
    ##############################################################################


set.seed(123)
# Par metros
n <- 200                # Tamanho da s rie
n_series <- 2000        # N mero de s ries
phi1 <- 0.2             # Coeficiente AR(1)
mu1 <- 1; mu2 <- 4      # M dias antes/depois
sd1 <- 1; sd2 <- 1      # Desvios padr o antes/depois
m_real <- 50            # Ponto verdadeiro de mudan a
penalt_val <- "BIC"
method1 <- "BinSeg"

#Armazenar vectores
matrx <- matrix(NA, nrow = n_series, ncol = n)
cpt.vet <- vector("list", n_series)
epsilon <- numeric(n_series)
num_detectados <- numeric(n_series)

# Simula  o com mudan a na m dia e na vari ncia
for (i in 1:n_series) {
  x1 <- arima.sim(n = m_real, model = list(ar = phi1), sd = sd1) +
     mu1
  x2 <- arima.sim(n = n - m_real, model = list(ar = phi1), sd = sd2
     ) + mu2
  y <- c(x1, x2)
  matrx[i, ] <- y

  # Detec  o de mudan a com BinSeg
  change_t <- cpt.meanvar(y, method = method1, penalty = penalt_val
     )
```

```
  detectados <- cpts(change_t)
  cpt.vet[[i]] <- detectados
  num_detectados[i] <- length(detectados)

  if (length(detectados) == 1) {
    epsilon[i] <- detectados - m_real
  } else {
    epsilon[i] <- NA
  }
}

epsilon_validos <- na.omit(epsilon)
n_validos <- length(epsilon_validos)

# taxa de acerto para janelas 0 a 5
Hs <- sapply(0:5, function(k) {
  sum(abs(epsilon_validos) <= k) / n_validos
})
Hs

# --- Gr fico 1: BinSeg ---
df <- data.frame(erro = sort(abs(epsilon_validos)))
df$proporcao <- ecdf(abs(epsilon_validos))(df$erro)

g1 <- ggplot(df, aes(x = erro, y = proporcao)) +
  geom_line(color = "blue", size = 1) +
  geom_vline(xintercept = 2, linetype = "dashed", color = "red",
     size = 1.5) +
  annotate("text", x = 4.5, y = 0.3,
           label = "",
           hjust = 0, vjust = 0,
           size = 4.2, fontface = "italic", color = "black") +
  labs(
    title = "Segmenta  o Bin ria",
    x = "|  |",
    y = "Propor  o acumulada"
  ) +
  scale_x_continuous(breaks = seq(0, 10, 2), limits = c(0, 10)) +
 theme_minimal(base_size = 13) +
  theme(
    panel.grid = element_blank(),
    axis.line = element_line(color = "black")
  )

# --- Fun  o CUSUM ---
cusum_stat <- function(y) {
  n <- length(y)
  stats <- numeric(n - 1)
  for (lambda in 1:(n - 1)) {
    mean1 <- mean(y[1:lambda])
    mean2 <- mean(y[(lambda + 1):n])
    stats[lambda] <- sqrt((lambda * (n - lambda)) / n) * abs(mean1
       - mean2)
  }
  return(stats)
}

# --- DETEC  O COM CUSUM ---
epsilon_cusum <- numeric(n_series)
```

```
for (i in 1:n_series) {
  y <- matrx[i, ]
  stats <- cusum_stat(y)
  lambda_est <- which.max(stats)  # ponto estimado de mudan a

  if (length(lambda_est) == 1) {
    epsilon_cusum[i] <- lambda_est - m_real
  } else {
    epsilon_cusum[i] <- NA
  }
}

epsilon_cusum_validos <- na.omit(epsilon_cusum)
n_validos_cusum <- length(epsilon_cusum_validos)

# taxa de acerto para janelas  0  a  5
Hs_cusum <- sapply(0:5, function(k) {
  sum(abs(epsilon_cusum_validos) <= k) / n_validos_cusum
})
Hs_cusum

# --- Gr fico 2: CUSUM ---
df_cusum <- data.frame(erro = sort(abs(epsilon_cusum_validos)))
df_cusum$proporcao <- ecdf(abs(epsilon_cusum_validos))(df_cusum$
    erro)

g2 <- ggplot(df_cusum, aes(x = erro, y = proporcao)) +
  geom_line(color = "darkgreen", size = 1) +
  geom_vline(xintercept = 3, linetype = "dashed", color = "red",
      size = 1.5) +
  annotate("text", x = 4.5, y = 0.3,
           label = "",
           hjust = 0, vjust = 0,
           size = 4.2, fontface = "italic", color = "black") +
  labs(
    title = "CUSUM",
    x = "|  |",
    y = "Propor  o acumulada"
  ) +
  scale_x_continuous(breaks = seq(0, 10, 2), limits = c(0, 10)) +
  theme_minimal(base_size = 13) +
  theme(
    panel.grid = element_blank(),
    axis.line = element_line(color = "black")
  )

# --- Compara  o Lado a Lado ---
g1 + g2 + plot_annotation(title = "Compara  o BinSeg vs CUSUM")

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


# Par metros da simula  o
set.seed(123)
m_real <- 50
n <- 200
n_serie <- 2000
```

```r
penalty1 <- "MBIC"
phis <- seq(0.01, 0.9, by = 0.05)
methods <- c("PELT")
intervalo_tolerancia <- 3
deltas <- c(0.5, 1.5, 3)

# Labels com
rotulos_delta <- c(expression(delta == 0.5),
                   expression(delta == 1.5),
                   expression(delta == 3))

# Resultados
resultados <- data.frame()
set.seed(123)

for (d in 1:length(deltas)) {
  delta <- deltas[d]
  for (phi in phis) {
    for (met in methods) {
      acertos <- 0
      for (i in 1:n_serie) {
        x1 <- arima.sim(n = m_real, model = list(ar = phi))
        x2 <- arima.sim(n = n - m_real, model = list(ar = phi)) +
           delta
        y <- c(x1, x2)
        cpt <- cpt.mean(y, penalty = penalty1, method = met, test.
           stat = "Normal")
        cps <- cpts(cpt)
        if (any(cps %in% (m_real - intervalo_tolerancia):(m_real +
           intervalo_tolerancia))) {
          acertos <- acertos + 1
        }
      }
      taxa <- acertos / n_serie
      resultados <- rbind(resultados, data.frame(delta = factor(
         deltas[d]), phi = phi, metodo = met, taxa_acerto = taxa))
    }
  }
}

# Gr fico com    na legenda
ggplot(resultados, aes(x = phi, y = taxa_acerto, color = delta)) +
  geom_line(size = 1.2) +
  labs(title = "",
       x = expression(phi),
       y = "Taxa de Acerto",
       color = "PL") +
  scale_color_manual(values = c("blue", "green", "red"), labels =
     rotulos_delta) +
  theme_minimal(base_size = 14)+
theme(
    panel.grid = element_blank(),
    axis.line = element_line(color = "black")
  )

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

library(dplyr)
library(tidyr)
```

```r
# Criar data frame com os valores
x <- seq(0, 5, length=1000)
df <- data.frame(
  x = x,
  t_05 = dt(x, df=0.5),
  t_1 = dt(x, df=1),
  t_15 = dt(x, df=1.5),
  t_3 = dt(x, df=3),
  t_10 = dt(x, df=10),
  normal = dnorm(x)
)

# Transformar para formato longo
df_long <- df %>%
  pivot_longer(cols = -x, names_to = "Distribuicao", values_to = "
     densidade")

# Mapear nomes para legendas com express o
labels_expr <- c(
  t_05 = expression(t(nu==0.5)),
  t_1 = expression(t(nu==1)),
  t_15 = expression(t(nu==1.5)),
  t_3 = expression(t(nu==3)),
  t_10 = expression(t(nu==10)),
  normal = "Normal padr o"
)

# Criar gr fico
ggplot(df_long, aes(x = x, y = densidade, color = Distribuicao,
    linetype = Distribuicao)) +
  geom_line(size = 1.3) +
  scale_color_manual(values = c("purple", "red", "orange", "blue",
     "green", "black"),
                     labels = labels_expr) +
  scale_linetype_manual(values = c(1,1,1,1,1,3),
                        labels = labels_expr) +
  labs(
    title = "Distribui  es t-Student vs Normal padr o (x >= 0)",
    x = "x",
    y = "Densidade",
    color = NULL,
    linetype = NULL
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "right",   # Colocar a legenda    direita
    legend.direction = "vertical", # Vertical
    legend.title = element_text(size = 12),
    legend.text = element_text(size = 10)
  )+
  theme(
    panel.grid = element_blank(),
    axis.line = element_line(color = "black")
  )

##################3
```

```r
# Par metros
m_real <- 50
n <- 200
n_serie <- 2000
phi <- 0.4
deltas <- c(0.5, 1.5, 3)  # fraca, moderada, forte
dfs <- seq(1, 10, by = 0.5)
tolerancia <- 3
methods <- "BinSeg"
names(methods) <- c("BinSeg")

set.seed(123)

# Resultados
resultados <- data.frame()

# Simula  o
for (delta in deltas) {
  for (df in dfs) {
    for (met_nome in names(methods)) {
      metodo <- methods[met_nome]
      penalty <- "BIC"
      acertos <- 0

      for (i in 1:n_serie) {
        # Inova  es t-Student
        innov1 <- rt(m_real, df = df)
        innov2 <- rt(n - m_real, df = df)

        # S rie AR(1) com mudan a de n vel delta
        x1 <- arima.sim(n = m_real, model = list(ar = phi), innov =
            innov1)
        x2 <- arima.sim(n = n - m_real, model = list(ar = phi),
           innov = innov2) + delta
        y <- c(x1, x2)

        # Detec  o de mudan a de m dia
        cpt <- cpt.mean(y, penalty = penalty, method = metodo)
        cps <- cpts(cpt)

        if (any(cps %in% (m_real - tolerancia):(m_real + tolerancia
           ))) {
          acertos <- acertos + 1
        }
      }

      taxa <- acertos / n_serie
      resultados <- rbind(resultados, data.frame(delta = delta, df
         = df, taxa_acerto = taxa))
    }
  }
}

# Transformar delta em fator para legendas
resultados$delta <- factor(resultados$delta, levels = deltas)

# Labels para legenda com    e intensidade
labels_delta <- c(expression("Fraca ("~delta==0.5~")"),
                  expression("Moderada ("~delta==1.5~")"),
```

```
                    expression("Forte ("~delta==3~")"))

# Plot
g1<-ggplot(resultados, aes(x = df, y = taxa_acerto, color = delta,
    group = delta)) +
  geom_line(size = 1.2) +

  scale_color_manual(
    values = c("purple", "orange", "red"),
    labels = labels_delta
  ) +
  labs(
    title = "",
    x = "df",
    y = "Taxa de acerto",
    color = "BS"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "right",
    legend.direction = "vertical",
    panel.grid = element_blank(),
    axis.line = element_line(color = "black")
  )

##################3


# Par metros
m_real <- 50
n <- 200
n_serie <- 2000
phi <- 0.4
deltas <- c(0.5, 1.5, 3)  # fraca, moderada, forte
dfs <- seq(1, 10, by = 0.5)
tolerancia <- 3
methods <- "PELT"
names(methods) <- c("BinSeg")

set.seed(123)

# Resultados
resultados <- data.frame()

# Simula  o
for (delta in deltas) {
  for (df in dfs) {
    for (met_nome in names(methods)) {
      metodo <- methods[met_nome]
      penalty <- "BIC"
      acertos <- 0

      for (i in 1:n_serie) {
        # Inova  es t-Student
        innov1 <- rt(m_real, df = df)
        innov2 <- rt(n - m_real, df = df)

        # S rie AR(1) com mudan a de n vel delta
```

```
        x1 <- arima.sim(n = m_real, model = list(ar = phi), innov =
            innov1)
        x2 <- arima.sim(n = n - m_real, model = list(ar = phi),
           innov = innov2) + delta
        y <- c(x1, x2)

        # Detec  o de mudan a de m dia
        cpt <- cpt.mean(y, penalty = penalty, method = metodo)
        cps <- cpts(cpt)

        if (any(cps %in% (m_real - tolerancia):(m_real + tolerancia
           ))) {
          acertos <- acertos + 1
        }
      }

      taxa <- acertos / n_serie
      resultados <- rbind(resultados, data.frame(delta = delta, df
         = df, taxa_acerto = taxa))
    }
  }
}

# Transformar delta em fator para legendas
resultados$delta <- factor(resultados$delta, levels = deltas)

# Labels para legenda com    e intensidade
labels_delta <- c(expression("Fraca ("~delta==0.5~")"),
                  expression("Moderada ("~delta==1.5~")"),
                  expression("Forte ("~delta==3~")"))

# Plot
g2<-ggplot(resultados, aes(x = df, y = taxa_acerto, color = delta,
    group = delta)) +
  geom_line(size = 1.2) +

  scale_color_manual(
    values = c("purple", "orange", "red"),
    labels = labels_delta
  ) +
  labs(
    title = "",
    x = "df",
    y = "",
    color = " PL"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "right",
    legend.direction = "vertical",
    panel.grid = element_blank(),
    axis.line = element_line(color = "black")
  )
######################3


# Par metros
m_real <- 50
n <- 200
```

```r
n_serie <- 200
phi <- 0.4
deltas <- c(0.5, 1.5, 3)  # fraca, moderada, forte
dfs <- seq(1, 10, by = 0.5)
tolerancia <- 3

set.seed(123)

# Resultados
resultados <- data.frame()

# Simula  o
for (delta in deltas) {
  for (df in dfs) {
    acertos <- 0

    for (i in 1:n_serie) {
      # Inova  es t-Student
      innov1 <- rt(m_real, df = df)
      innov2 <- rt(n - m_real, df = df)

      # S rie AR(1) com mudan a de n vel delta
      x1 <- arima.sim(n = m_real, model = list(ar = phi), innov =
          innov1)
      x2 <- arima.sim(n = n - m_real, model = list(ar = phi), innov
           = innov2) + delta
      y <- c(x1, x2)

      # Detec  o de mudan a (for ando 1 quebra)
      fm <- breakpoints(y ~ 1, breaks = 1, h = 10)
      cps <- fm$breakpoints

      if (any(cps %in% (m_real - tolerancia):(m_real + tolerancia))
          ) {
        acertos <- acertos + 1
      }
    }

    taxa <- acertos / n_serie
    resultados <- rbind(resultados, data.frame(delta = delta, df =
        df, taxa_acerto = taxa))
  }
}

# Transformar delta em fator para legendas
resultados$delta <- factor(resultados$delta, levels = deltas)

# Labels para legenda com    e intensidade
labels_delta <- c(expression("Fraca ("~delta==0.5~")"),
                  expression("Moderada ("~delta==1.5~")"),
                  expression("Forte ("~delta==3~")"))

# Plot
g3 <- ggplot(resultados, aes(x = df, y = taxa_acerto, color = delta
    , group = delta)) +
  geom_line(size = 1.2) +
  scale_color_manual(values = c("purple", "orange", "red"),
                     labels = labels_delta) +
  labs(
```

```
    title = "",
    x = "df",
    y = "",
    color = "BP"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "right",
    legend.direction = "vertical",
    panel.grid = element_blank(),
    axis.line = element_line(color = "black")
  )
g1+ g2 + g3
#########################################################################
library(changepoint)
library(ggplot2)

# Par metros da simula  o
set.seed(123)
m_real <- 50
n <- 200
n_serie <- 2000
penalty1 <- "MBIC"
phis <- seq(0.01, 0.9, by = 0.05)
methods <- c("BinSeg")
intervalo_tolerancia <- 3
deltas <- c(0.5, 1.5, 3)

# Labels com
rotulos_delta <- c(expression(delta == 0.5),
                   expression(delta == 1.5),
                   expression(delta == 3))

# Resultados
resultados <- data.frame()
set.seed(123)

for (d in 1:length(deltas)) {
  delta <- deltas[d]
  for (phi in phis) {
    for (met in methods) {
      acertos <- 0
      for (i in 1:n_serie) {
        x1 <- arima.sim(n = m_real, model = list(ar = phi))
        x2 <- arima.sim(n = n - m_real, model = list(ar = phi)) +
          delta
        y <- c(x1, x2)
        cpt <- cpt.mean(y, penalty = penalty1, method = met, test.
          stat = "Normal")
        cps <- cpts(cpt)
        if (any(cps %in% (m_real - intervalo_tolerancia):(m_real +
          intervalo_tolerancia))) {
          acertos <- acertos + 1
        }
      }
      taxa <- acertos / n_serie
      resultados <- rbind(resultados, data.frame(delta = factor(
        deltas[d]), phi = phi, metodo = met, taxa_acerto = taxa))
    }
```

```
  }
}

# Gr fico com    na legenda
g1<-ggplot(resultados, aes(x = phi, y = taxa_acerto, color = delta)
    ) +
  geom_line(size = 1.2) +
  labs(title = "",
       x = expression(phi),
       y = "Taxa de Acerto",
       color = "BS") +
  scale_color_manual(values = c("blue", "green", "red"), labels =
     rotulos_delta) +
  theme_minimal(base_size = 14)+
  theme(
    panel.grid = element_blank(),
    axis.line = element_line(color = "black")
  )
#################
# Par metros da simula  o
set.seed(123)
m_real <- 50
n <- 200
n_serie <- 200
penalty1 <- "MBIC"
phis <- seq(0.01, 0.9, by = 0.05)
methods <- c("PELT")
intervalo_tolerancia <- 3
deltas <- c(0.5, 1.5, 3)

# Labels com
rotulos_delta <- c(expression(delta == 0.5),
                   expression(delta == 1.5),
                   expression(delta == 3))

# Resultados
resultados <- data.frame()
set.seed(123)

for (d in 1:length(deltas)) {
  delta <- deltas[d]
  for (phi in phis) {
    for (met in methods) {
      acertos <- 0
      for (i in 1:n_serie) {
        x1 <- arima.sim(n = m_real, model = list(ar = phi))
        x2 <- arima.sim(n = n - m_real, model = list(ar = phi)) +
           delta
        y <- c(x1, x2)
        cpt <- cpt.mean(y, penalty = penalty1, method = met, test.
           stat = "Normal")
        cps <- cpts(cpt)
        if (any(cps %in% (m_real - intervalo_tolerancia):(m_real +
           intervalo_tolerancia))) {
          acertos <- acertos + 1
        }
      }
      taxa <- acertos / n_serie
```

```
      resultados <- rbind(resultados, data.frame(delta = factor(
          deltas[d]), phi = phi, metodo = met, taxa_acerto = taxa))
    }
  }
}

# Gr fico com    na legenda
g2<-ggplot(resultados, aes(x = phi, y = taxa_acerto, color = delta)
    ) +
  geom_line(size = 1.2) +
  labs(title = "",
       x = expression(phi),
       y = "",
       color = "PL") +
  scale_color_manual(values = c("blue", "green", "red"), labels =
      rotulos_delta) +
  theme_minimal(base_size = 14)+
  theme(
    panel.grid = element_blank(),
    axis.line = element_line(color = "black")
  )
###########library(changepoint)

# Par metros da simula  o
set.seed(123)
m_real <- 50
n <- 200
n_serie <- 2000
penalty1 <- "MBIC"
phis <- seq(0.01, 0.9, by = 0.05)
methods <- c("AMOC")
intervalo_tolerancia <- 3
deltas <- c(0.5, 1.5, 3)

# Labels com
rotulos_delta <- c(expression(delta == 0.5),
                   expression(delta == 1.5),
                   expression(delta == 3))

# Resultados
resultados <- data.frame()
set.seed(123)

for (d in 1:length(deltas)) {
  delta <- deltas[d]
  for (phi in phis) {
    for (met in methods) {
      acertos <- 0
      for (i in 1:n_serie) {
        x1 <- arima.sim(n = m_real, model = list(ar = phi))
        x2 <- arima.sim(n = n - m_real, model = list(ar = phi)) +
            delta
        y <- c(x1, x2)
        cpt <- cpt.mean(y, penalty = penalty1, method = met, test.
            stat = "Normal")
        cps <- cpts(cpt)
        if (any(cps %in% (m_real - intervalo_tolerancia):(m_real +
            intervalo_tolerancia))) {
          acertos <- acertos + 1
```

```
        }
      }
      taxa <- acertos / n_serie
      resultados <- rbind(resultados, data.frame(delta = factor(
          deltas[d]), phi = phi, metodo = met, taxa_acerto = taxa))
    }
  }
}

# Gr fico com    na legenda
g3<-ggplot(resultados, aes(x = phi, y = taxa_acerto, color = delta)
    ) +
  geom_line(size = 1.2) +
  labs(title = "",
       x = expression(phi),
       y = "Taxa de Acerto",
       color = "AMOC") +
  scale_color_manual(values = c("blue", "green", "red"), labels =
     rotulos_delta) +
  theme_minimal(base_size = 14)+
  theme(
    panel.grid = element_blank(),
    axis.line = element_line(color = "black")
  )
######################

# Par metros da simula  o
set.seed(123)
m_real <- 50
n <- 200
n_serie <- 2000
penalty1 <- "BIC"
phis <- seq(0.01, 0.9, by = 0.05)
methods <- c("SegNeigh")
intervalo_tolerancia <- 3
deltas <- c(0.5, 1.5, 3)

# Labels com
rotulos_delta <- c(expression(delta == 0.5),
                   expression(delta == 1.5),
                   expression(delta == 3))

# Resultados
resultados <- data.frame()
set.seed(123)

for (d in 1:length(deltas)) {
  delta <- deltas[d]
  for (phi in phis) {
    for (met in methods) {
      acertos <- 0
      for (i in 1:n_serie) {
        x1 <- arima.sim(n = m_real, model = list(ar = phi))
        x2 <- arima.sim(n = n - m_real, model = list(ar = phi)) +
          delta
        y <- c(x1, x2)
        cpt <- cpt.mean(y, penalty = penalty1, method = met, test.
          stat = "Normal")
        cps <- cpts(cpt)
```

```
        if (any(cps %in% (m_real - intervalo_tolerancia):(m_real +
            intervalo_tolerancia))) {
          acertos <- acertos + 1
        }
      }
      taxa <- acertos / n_serie
      resultados <- rbind(resultados, data.frame(delta = factor(
          deltas[d]), phi = phi, metodo = met, taxa_acerto = taxa))
    }
  }
}

# Gr fico com    na legenda
g5<-ggplot(resultados, aes(x = phi, y = taxa_acerto, color = delta)
    ) +
  geom_line(size = 1.2) +
  labs(title = "",
       x = expression(phi),
       y = "Taxa de Acerto",
       color = "SegNeigh") +
  scale_color_manual(values = c("blue", "green", "red"), labels =
      rotulos_delta) +
  theme_minimal(base_size = 14)+
  theme(
    panel.grid = element_blank(),
    axis.line = element_line(color = "black")
  )

##########3

# Par metros da simula  o
set.seed(123)
m_real <- 50
n <- 200
n_serie <- 2000
phis <- seq(0.01, 0.9, by = 0.05)
methods <- c("breakpoints") # s  ilustrativo, pois breakpoints
    a fun  o usada
intervalo_tolerancia <- 3
deltas <- c(0.5, 1.5, 3)

# Labels com
rotulos_delta <- c(expression(delta == 0.5),
                   expression(delta == 1.5),
                   expression(delta == 3))

# Resultados
resultados <- data.frame()

for (d in 1:length(deltas)) {
  delta <- deltas[d]
  for (phi in phis) {
    acertos <- 0
    for (i in 1:n_serie) {
      # duas partes AR(1) com shift na m dia
      x1 <- arima.sim(n = m_real, model = list(ar = phi))
      x2 <- arima.sim(n = n - m_real, model = list(ar = phi)) +
          delta
      y <- c(x1, x2)
```

```
      # Ajuste de breakpoints (1 quebra)
      bp <- breakpoints(y ~ 1, breaks = 1)
      cps <- bp$breakpoints

      # Contagem de acertos (dentro do intervalo de toler ncia)
      if (any(cps %in% (m_real - intervalo_tolerancia):(m_real +
         intervalo_tolerancia))) {
        acertos <- acertos + 1
      }
    }
    taxa <- acertos / n_serie
    resultados <- rbind(resultados,
                        data.frame(delta = factor(deltas[d]),
                                   phi = phi,
                                   metodo = "breakpoints",
                                   taxa_acerto = taxa))
  }
}

# Gr fico com    na legenda
g6 <- ggplot(resultados, aes(x = phi, y = taxa_acerto, color =
    delta)) +
  geom_line(size = 1.2) +
  labs(title = "",
       x = expression(phi),
       y = "",
       color = "BP") +
  scale_color_manual(values = c("blue", "green", "red"), labels =
     rotulos_delta) +
  theme_minimal(base_size = 14) +
  theme(
    panel.grid = element_blank(),
    axis.line = element_line(color = "black")
  )


g1+g2+g6
################################333
#dependencia
m_real <- 50
n <- 200
n_serie <- 2000
df <- 5  # graus de liberdade da t-student

penalties <- c("AIC", "BIC", "MBIC")
mu2_vec <- c(0.5, 1.5, 3)
phi_vec <- c(0.2, 0.5, 0.8)  # valores do phi a testar
mean1 <- 0   # m dia antes da mudan a
tol <- 3     # toler ncia para considerar acerto

set.seed(123)

# Array 3D para armazenar taxa de acerto: phi x mu2 x penalty
taxas_acerto <- array(NA,
                      dim = c(length(phi_vec), length(mu2_vec),
                         length(penalties)),
                      dimnames = list(paste0("phi=", phi_vec),
                                      paste0("mu2=", mu2_vec),
```

```
                                        penalties))

for (f in seq_along(phi_vec)) {
  phi <- phi_vec[f]

  for (m in seq_along(mu2_vec)) {
    mean2 <- mu2_vec[m] + mean1   # m dia ap s mudan a

    for (p in seq_along(penalties)) {
      penalty1 <- penalties[p]
      cp_detected <- vector("list", n_serie)

      for (i in 1:n_serie) {
        innov1 <- rt(m_real, df = df)
        innov2 <- rt(n - m_real, df = df)

        x1 <- arima.sim(n = m_real, model = list(ar = phi), innov =
            innov1) + mean1
        x2 <- arima.sim(n = n - m_real, model = list(ar = phi),
           innov = innov2) + mean2
        y <- c(x1, x2)

        cpt <- cpt.mean(y, penalty = penalty1, method = "BinSeg")
        cp_detected[[i]] <- cpts(cpt)
      }

      acertos <- sum(sapply(cp_detected, function(x) any(x %in% (m_
         real - tol):(m_real + tol))))
      taxas_acerto[f, m, p] <- acertos / n_serie
    }
  }
}

print(round(taxas_acerto * 100, 2))
#################################################################


## Res duos normais e independentes
# Processos estacion rios em segmentos
library(changepoint)

n <- 200          # tamanho total da s rie
breakp <- 50      # ponto real da mudan a
mean1 <- 0
mean2 <- 1
penalty1 <- "AIC"
n_sim <- 2000
tol <- 3

detected_cpts <- vector("list", n_sim)

set.seed(123)

for (i in 1:n_sim) {
  yt <- c(
    rnorm(breakp, mean = mean1, sd = 1),
    rnorm(n - breakp, mean = mean2, sd = 1)
  )
  y <- ts(yt)
```

```
  cp <- cpt.mean(y, method = "PELT", penalty = penalty1)
  detected_cpts[[i]] <- cpts(cp)
}

acertos <- sum(sapply(detected_cpts, function(x) any(x %in% 47:53))
    )
tax_acert <- acertos / n_sim
print(paste("Taxa de acerto:", round(tax_acert, 4)*100, "%"))



set.seed(123)

# Par metros
n <- 200
n_series <- 2000
phi1 <- 0.2
mu1 <- 1
mu2 <- 4
sd1 <- 1
sd2 <- 2
m_real <- 50
method1 <- "BinSeg"
penalties <- c("AIC", "BIC", "MBIC")
tol_max <- 5  # janelas  0  a  5

# Matriz para armazenar taxas de acerto
taxas_acerto <- matrix(NA, nrow = tol_max + 1, ncol = length(
    penalties),
                       dimnames = list(paste0("  ", 0:tol_max),
                          penalties))

for (p in seq_along(penalties)) {
  pen <- penalties[p]
  epsilon <- numeric(n_series)

  for (i in 1:n_series) {
    # Simula  o da s rie com mudan a de m dia e vari ncia
    x1 <- arima.sim(n = m_real, model = list(ar = phi1), sd = sd1)
       + mu1
    x2 <- arima.sim(n = n - m_real, model = list(ar = phi1), sd =
       sd2) + mu2
    y <- c(x1, x2)

    # Detec  o de mudan a
    cp <- cpt.meanvar(y, method = method1, penalty = pen)
    detectados <- cpts(cp)

    if (length(detectados) == 1) {
      epsilon[i] <- detectados - m_real
    } else {
      epsilon[i] <- NA
    }
  }

  epsilon_validos <- na.omit(epsilon)
  n_validos <- length(epsilon_validos)
```

```
  # Calcular taxas de acerto para janelas 0 a 5
  taxas_acerto[, p] <- sapply(0:tol_max, function(k) {
    sum(abs(epsilon_validos) <= k) / n_validos * 100
  })
}

# Transformar em dat para visualiza o
df_taxas <- data.frame(Janela = rownames(taxas_acerto), taxas_
    acerto)
print(df_taxas)
#
    ##############################################################################

library(changepoint)

# Par metros
n <- 200
breakp <- 50
mean1 <- 0
mu2_vec <- c(0.5, 1.5, 3)   # magnitude da mudan a
penalties <- "MBIC"
n_sim <- 2000
tol <- 3
phi_vec <- c(0, 0.2, 0.5, 0.8)   # coeficientes AR(1)
METHOD <- c("BinSeg", "PELT")    # m todos a testar

# Array 4D: phi x mu2 x m todo x penalty
taxas_acerto <- array(NA,
                      dim = c(length(phi_vec), length(mu2_vec),
                         length(METHOD), length(penalties)),
                      dimnames = list(
                        paste0("phi=", phi_vec),
                        paste0("mu2=", mu2_vec),
                        METHOD,
                        penalties
                      ))

set.seed(123)

for (j in seq_along(phi_vec)) {
  phi_val <- phi_vec[j]

  for (m in seq_along(mu2_vec)) {
    mean2 <- mu2_vec[m]

    for (meth in seq_along(METHOD)) {
      method_val <- METHOD[meth]

      for (p in seq_along(penalties)) {
        penalty1 <- penalties[p]
        cpt.vet <- vector("list", n_sim)

        for (i in 1:n_sim) {
          # simula AR(1) normal
          Z <- arima.sim(model = list(ar = phi_val), n = n)
          yt <- c(Z[1:breakp] + mean1,
                  Z[(breakp+1):n] + mean2)
```

```
          cp <- cpt.mean(yt, method = method_val, penalty =
             penalty1, Q = 1)
          cpt.vet[[i]] <- cpts(cp)
        }

        acertos <- sum(sapply(cpt.vet, function(x) any(x %in% (
           breakp - tol):(breakp + tol))))
        taxas_acerto[j, m, meth, p] <- acertos / n_sim
      }
    }
  }
}

# Visualizar resultados em porcentagem
print(round(taxas_acerto * 100, 2))
#
    ##############################################################################

# Overthingt



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



library(changepoint)

# Par metros
n <- 200
breakp <- 50
mean1 <- 0
mu2_vec <- c(0.5, 1.5, 3)   # magnitude da mudan a
penalties <- "MBIC"
n_sim <- 2000
tol <- 3
phi <- c(0.2, 0.5, 0.8)    # graus de liberdade
METHOD <- c("BinSeg", "PELT")   # M todos a testar

set.seed(123)

# Array 4D: df x mu2 x m todo x penalty
taxas_acerto <- array(NA,
                      dim = c(length(df_vec), length(mu2_vec),
                         length(METHOD), length(penalties)),
                      dimnames = list(
                        paste0("df=", df_vec),
                        paste0("mu2=", mu2_vec),
                        METHOD,
                        penalties
                      ))

for (j in seq_along(df_vec)) {
  df_t <- df_vec[j]

  for (m in seq_along(mu2_vec)) {
    mean2 <- mu2_vec[m]
```

```
    for (meth in seq_along(METHOD)) {
      method_val <- METHOD[meth]

      for (p in seq_along(penalties)) {
        penalty1 <- penalties[p]
        cpt.vet <- vector("list", n_sim)

        for (i in 1:n_sim) {
          yt <- c(
            rt(breakp, df = df_t) + mean1,
            rt(n - breakp, df = df_t) + mean2
          )

          cp <- cpt.mean(yt, method = method_val, penalty =
             penalty1, Q = 1)
          cpt.vet[[i]] <- cpts(cp)
        }

        acertos <- sum(sapply(cpt.vet, function(x) any(x %in% (
           breakp - tol):(breakp + tol))))
        taxas_acerto[j, m, meth, p] <- acertos / n_sim
      }
    }
  }
}

# Visualizar resultados em porcentagem
print(round(taxas_acerto * 100, 2))
#
    ##########################################################################333

library(strucchange)

# Par metros
n <- 200
breakp <- 50
mean1 <- 0
mu2_vec <- c(0.5, 1.5, 3)    # magnitude da mudan a
n_sim <- 2000
tol <- 3
df_vec <- c(1.5, 2.5, 5)     # graus de liberdade
set.seed(123)

# Array 3D: df x mu2 x m todo (s   BK aqui)
taxas_acerto <- array(NA,
                      dim = c(length(df_vec), length(mu2_vec), 1),
                      dimnames = list(
                        paste0("df=", df_vec),
                        paste0("mu2=", mu2_vec),
                        "BK"
                      ))

for (j in seq_along(df_vec)) {
  df_t <- df_vec[j]

  for (m in seq_along(mu2_vec)) {
    mean2 <- mu2_vec[m]
    cpt.vet <- vector("list", n_sim)
```

```
    for (i in 1:n_sim) {
      yt <- c(
        rt(breakp, df = df_t) + mean1,
        rt(n - breakp, df = df_t) + mean2
      )

      # Modelo de regress o simples para detectar mudan a na
         m dia
      bp <- breakpoints(yt ~ 1, h = 10)   # h = tamanho m nimo do
         segmento
      cpt.vet[[i]] <- bp$breakpoints
    }

    acertos <- sum(sapply(cpt.vet, function(x) any(x %in% (breakp -
        tol):(breakp + tol))))
    taxas_acerto[j, m, "BK"] <- acertos / n_sim
  }
}

# Resultados em %
print(round(taxas_acerto * 100, 2))

#################################################################33

library(changepoint)

# Par metros
n <- 200
breakp <- 50
mean1 <- 0
mu2_vec <- c(0.5, 1.5, 3)
penalties <- "BIC"
n_sim <- 2000
tol <- 3
df_vec <- c(1.5, 2.5, 5)
METHOD <- c("BinSeg", "PELT")
phi <- 0.8  # autocorrela  o AR(1)

set.seed(123)

# Array 4D: df x mu2 x m todo x penalty
taxas_acerto <- array(NA,
                      dim = c(length(df_vec), length(mu2_vec),
                         length(METHOD), length(penalties)),
                      dimnames = list(
                        paste0("df=", df_vec),
                        paste0("mu2=", mu2_vec),
                        METHOD,
                        penalties
                      ))

for (j in seq_along(df_vec)) {
  df_t <- df_vec[j]

  for (m in seq_along(mu2_vec)) {
    mean2 <- mu2_vec[m]

    for (meth in seq_along(METHOD)) {
      method_val <- METHOD[meth]
```

```
      for (p in seq_along(penalties)) {
        penalty1 <- penalties[p]
        cpt.vet <- vector("list", n_sim)

        for (i in 1:n_sim) {
          # gera inova  es t-student
          eps1 <- rt(breakp, df = df_t)
          eps2 <- rt(n - breakp, df = df_t)

          # gera s rie AR(1) manualmente
          yt1 <- numeric(breakp)
          yt2 <- numeric(n - breakp)

          yt1[1] <- eps1[1] + mean1
          for (t in 2:breakp) {
            yt1[t] <- phi * yt1[t-1] + eps1[t] + mean1
          }

          yt2[1] <- eps2[1] + mean2
          for (t in 2:(n - breakp)) {
            yt2[t] <- phi * yt2[t-1] + eps2[t] + mean2
          }

          yt <- c(yt1, yt2)

          # detec  o de mudan a
          cp <- cpt.mean(yt, method = method_val, penalty =
             penalty1, Q = 1)
          cpt.vet[[i]] <- cpts(cp)
        }

        acertos <- sum(sapply(cpt.vet, function(x) any(x %in% (
           breakp - tol):(breakp + tol))))
        taxas_acerto[j, m, meth, p] <- acertos / n_sim
      }
    }
  }
}

# Resultados em porcentagem
print(round(taxas_acerto * 100, 2))
############################3
library(strucchange)

# Par metros
n <- 200
breakp <- 50
mean1 <- 0
mu2_vec <- c(0.5, 1.5, 3)
n_sim <- 2000
tol <- 3
df_vec <- c(1.5, 2.5, 5)
phi <- 0.8   # autocorrela  o AR(1)

set.seed(123)


taxas_acerto <- array(NA,
```

```
                        dim = c(length(df_vec), length(mu2_vec)),
                        dimnames = list(
                          paste0("df=", df_vec),
                          paste0("mu2=", mu2_vec)
                        ))

for (j in seq_along(df_vec)) {
  df_t <- df_vec[j]

  for (m in seq_along(mu2_vec)) {
    mean2 <- mu2_vec[m]

    cpt.vet <- vector("list", n_sim)

    for (i in 1:n_sim) {
      # gera inova  es t-student
      eps1 <- rt(breakp, df = df_t)
      eps2 <- rt(n - breakp, df = df_t)

      # gera s rie AR(1) manualmente
      yt1 <- numeric(breakp)
      yt2 <- numeric(n - breakp)

      yt1[1] <- eps1[1] + mean1
      for (t in 2:breakp) {
        yt1[t] <- phi * yt1[t-1] + eps1[t] + mean1
      }

      yt2[1] <- eps2[1] + mean2
      for (t in 2:(n - breakp)) {
        yt2[t] <- phi * yt2[t-1] + eps2[t] + mean2
      }

      yt <- c(yt1, yt2)

      # detec  o de mudan a usando strucchange
      bp <- breakpoints(yt ~ 1, breaks = 1)
      cpt.vet[[i]] <- bp$breakpoints
    }

    acertos <- sum(sapply(cpt.vet, function(x) any(x %in% (breakp -
        tol):(breakp + tol))))
    taxas_acerto[j, m] <- acertos / n_sim
  }
}

# Resultados em porcentagem
print(round(taxas_acerto * 100, 2))

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

# Par metros
n <- 200
breakp <- 50
mean1 <- 0
mu2_vec <- c(0.5, 1.5, 3)
penalties <- "BIC"
n_sim <- 2000
tol <- 3
```

```r
df_vec <- c(1.5, 2.5, 5)
METHOD <- c("BinSeg", "PELT")
phi <- 0.8  # autocorrela  o AR(1)

set.seed(123)

# Array para guardar apenas falsos positivos
taxas_fp <- array(NA,
                  dim = c(length(df_vec), length(mu2_vec), length(
                     METHOD), length(penalties)),
                  dimnames = list(
                    paste0("df=", df_vec),
                    paste0("mu2=", mu2_vec),
                    METHOD,
                    penalties
                  ))

for (j in seq_along(df_vec)) {
  df_t <- df_vec[j]

  for (m in seq_along(mu2_vec)) {
    mean2 <- mu2_vec[m]

    for (meth in seq_along(METHOD)) {
      method_val <- METHOD[meth]

      for (p in seq_along(penalties)) {
        penalty1 <- penalties[p]
        cpt.vet <- vector("list", n_sim)

        for (i in 1:n_sim) {
          # gera inova  es t-student
          eps1 <- rt(breakp, df = df_t)
          eps2 <- rt(n - breakp, df = df_t)

          # gera s rie AR(1) manualmente
          yt1 <- numeric(breakp)
          yt2 <- numeric(n - breakp)

          yt1[1] <- eps1[1] + mean1
          for (t in 2:breakp) {
            yt1[t] <- phi * yt1[t-1] + eps1[t] + mean1
          }

          yt2[1] <- eps2[1] + mean2
          for (t in 2:(n - breakp)) {
            yt2[t] <- phi * yt2[t-1] + eps2[t] + mean2
          }

          yt <- c(yt1, yt2)
          cp <- cpt.mean(yt, method = method_val, penalty =
             penalty1, Q = 5)
          cpt.vet[[i]] <- cpts(cp)
        }

        falsos_pos <- sum(sapply(cpt.vet, function(x) any(!(x %in%
           (breakp - tol):(breakp + tol)))))

        taxas_fp[j, m, meth, p] <- falsos_pos / n_sim
```

```
      }
    }
  }
}

print(round(taxas_fp * 100, 2))
############################################################

# ---------------------- Par metros ----------------------
n <- 200                # comprimento da s rie
breakp <- 50            # ponto de mudan a verdadeiro (k)
mean1 <- 0              # m dia antes da mudan a
mu2_vec <- c(0.5, 1.5, 3)  # magnitudes da mudan a (mu2)
penalties <- "MBIC"     # penaliza  o
n_sim <- 2000        # n mero de simula  es
tol <- 3                # raio de toler ncia (breakp +/- tol)
phi_vec <- c(0.2, 0.5, 0.8)  # coeficientes AR(1)
METHOD <- c("BinSeg", "PELT")   # m todos a testar

# ---------------------- Inicializar array ----------------------
# Este array armazenar  a PROPOR  O de vezes que ocorreu o "
   overfit"
overfit <- array(NA,
                 dim = c(length(phi_vec), length(mu2_vec), length(
                    METHOD), length(penalties)),
                 dimnames = list(
                   paste0("phi=", phi_vec),
                   paste0("mu2=", mu2_vec),
                   METHOD,
                   penalties
                 ))

# ---------------------- Simula  o (CORRIGIDA)
   ----------------------
set.seed(123)

for (j in seq_along(phi_vec)) {
  phi_val <- phi_vec[j]

  for (m in seq_along(mu2_vec)) {
    mean2 <- mu2_vec[m]

    for (meth in seq_along(METHOD)) {
      method_val <- METHOD[meth]

      for (p in seq_along(penalties)) {
        penalty1 <- penalties[p]

        # VETOR TEMPOR RIO: Armazena o resultado de cada uma das n
           _sim itera  es
        overfit_results <- numeric(n_sim)

        for (i in 1:n_sim) {
          # 1. Simular a s rie temporal (AR(1) + Mudan a de
             M dia)
          Z <- arima.sim(model = list(ar = phi_val), n = n)
          yt <- c(Z[1:breakp] + mean1,
                  Z[(breakp+1):n] + mean2)
```

```
            # 2. Detec  o de ponto de mudan a
            # Usamos Q = 1 para for ar a detec  o de um  nico
               ponto de mudan a
            cp <- cpt.mean(yt, method = method_val, penalty =
               penalty1, Q = 1)
            est_cpts <- cpts(cp)

            # 3. Contar o n mero de "overfit" na simula  o atual (
               deve ser 0 ou 1)
            # A condi  o de "overfit"  : o CP estimado est  fora
               do intervalo [breakp - tol, breakp + tol]

            # Tratamento para quando n o h  CP detectado (est_cpts
                NA):
            # sum(..., na.rm=TRUE) garantir  que ele conte 0 se est_
               cpts for NA.
            is_overfit <- sum(est_cpts < (breakp - tol) | est_cpts >
               (breakp + tol), na.rm = TRUE)

            # Armazenar o resultado da itera  o 'i'
            overfit_results[i] <- is_overfit
          }

          # 4. C lculo final: Armazenar
          # n mero total de simula  es)
          overfit[j, m, meth, p] <- sum(overfit_results) / n_sim
        }
      }
    }
}

############################################################
# Mudan as na Forma da Distribui  o

require(evd)
library(npcp)
n <- 100
k <- 50 ## the true change-point
## Change in the shape parameter of a GEV
x <- rgev(k,loc=0,scale=1,shape=0)
y <- rgev(k,loc=0,scale=1,shape=0.8)
cp <- cpBlockMax(c(x,y), r=10,method = "pwm")
cp$pvalues[3]
max(cp$stats.shape)
## Estimated change-point
which(cp$stats.shape == max(cp$stats.shape))
## Change in the scale parameter of a GEV

cpCopula
x <- rgev(k,loc=0,scale=0.5,shape=0)
y <- rgev(k,loc=0,scale=1,shape=0)
cp <- cpBlockMax(c(x,y))
cp
## Estimated change-point
which(cp$stats.scale == max(cp$stats.scale))
## Change in the location parameter of a GEV
x <- rgev(k,loc=0,scale=1,shape=0)
y <- rgev(k,loc=0.5,scale=1,shape=0)
cp <- cpBlockMax(c(x,y))
```

```
cp
## Estimated change-point
which(cp$stats.loc == max(cp$stats.loc))

######## C digo simples################

library(evd)
set.seed(123)
n <- 1000
shape1 <- 0.1
shape2 <- 0.6
phi <- 0.2
k <- 500
z <- numeric(n)
z[1] <- rnorm(1)
for (t in 2:n) {
  z[t] <- phi * z[t - 1] + rnorm(1)
}
U <- pnorm(z)

x1 <- qgev(U[1:k], loc = 0, scale = 1, shape = shape1)
x2 <- qgev(U[(k+1):n], loc = 0, scale = 1, shape = shape2)


y <- c(x1, x2)

library(npcp)
which(cp$stats.loc == max(cp$stats.shape))

#
    ##########################################################################33


# ---------- 1. Fun   o para simular GEV com depend ncia AR(1)
    ----------
sim_gev_ar1 <- function(n, phi, xi, mu = 0, sigma = 1) {
  # Passo 1: simula AR(1) normal padr o
  Z <- arima.sim(model = list(ar = phi), n = n)
  # Passo 2: transforma para uniforme (CDF normal)
  U <- pnorm(Z)
  # Passo 3: aplica fun   o quantil da GEV
  X <- qgev(U, loc = mu, scale = sigma, shape = xi)
  return(X)
}

# ---------- 2. Simular cen rio com mudan a em xi ----------
sim_gev_ar1_change <- function(n, phi, xi1, xi2, lambda, mu = 0,
    sigma = 1) {
  Z <- arima.sim(model = list(ar = phi), n = n)
  U <- pnorm(Z)
  U1 <- U[1:lambda]
  U2 <- U[(lambda+1):n]
  X1 <- qgev(U1, loc = mu, scale = sigma, shape = xi1)
  X2 <- qgev(U2, loc = mu, scale = sigma, shape = xi2)
  return(c(X1, X2))
}

# ---------- 3. Fun   o para rodar o teste cpBlockMax ----------
rodar_teste <- function(x, metodo = "gpwm", r = 10) {
```

```
  x <- as.numeric(x)  # garantir que    vetor num rico
  res <- cpBlockMax(x, method = metodo, r = r)
  return(res$pvalues[3]) # p-valor para mudan a no shape
}

# ---------- 4. Simula  o para n vel emp rico e poder
   ----------
set.seed(123)
n <- 200
Nsim <- 2000
phi_vals <- c(0.2, 0.5, 0.8)
alpha <- 0.05
xi_const <- 0.2

nivel_emp <- numeric(length(phi_vals))
poder <- numeric(length(phi_vals))

for (i in seq_along(phi_vals)) {
  # N vel emp rico (sem mudan a)
  rejeicoes <- 0
  for (j in 1:Nsim) {
    X <- sim_gev_ar1(n, phi = phi_vals[i], xi = xi_const)
    pval <- rodar_teste(X, metodo = "gpwm", r = 10)
    if (pval < alpha) rejeicoes <- rejeicoes + 1
  }
  nivel_emp[i] <- rejeicoes / Nsim

  # Poder (com mudan a em xi)
  rejeicoes <- 0
  for (j in 1:Nsim) {
    X <- sim_gev_ar1_change(n, phi = phi_vals[i], xi1 = 0.2, xi2 =
       0.4, lambda = n/2)
    pval <- rodar_teste(X, metodo = "gpwm", r = 10)
    if (pval < alpha) rejeicoes <- rejeicoes + 1
  }
  poder[i] <- rejeicoes / Nsim
}

# ---------- 5. Resultados ----------
data.frame(phi = phi_vals, NivelEmpirico = nivel_emp, Poder = poder
   )
##############################################################
############# Estimar pontos de localiza  o
   ###########################################


# ---------- 1. Fun  o para simular GEV com depend ncia AR(1)
   ----------
sim_gev_ar1 <- function(n, phi, xi, mu = 0, sigma = 1) {
  # Passo 1: simula AR(1) normal padr o
  Z <- arima.sim(model = list(ar = phi), n = n)
  # Passo 2: transforma para uniforme (CDF normal)
  U <- pnorm(Z)
  # Passo 3: aplica fun  o quantil da GEV
  X <- qgev(U, loc = mu, scale = sigma, shape = xi)
  return(X)
}

# ---------- 2. Simular cen rio com mudan a em xi ----------
```

```r
sim_gev_ar1_change <- function(n, phi, xi1, xi2, lambda, mu = 0,
    sigma = 1) {
  Z <- arima.sim(model = list(ar = phi), n = n)
  U <- pnorm(Z)
  lambda <- as.integer(lambda)
  if (lambda < 1 || lambda >= n) stop("lambda deve estar entre 1 e
     n-1")
  U1 <- U[1:lambda]
  U2 <- U[(lambda+1):n]
  X1 <- qgev(U1, loc = mu, scale = sigma, shape = xi1)
  X2 <- qgev(U2, loc = mu, scale = sigma, shape = xi2)
  return(c(X1, X2))
}

# ---------- 3. Fun  o para rodar o teste cpBlockMax (robusta)
    ----------
rodar_teste <- function(x, metodo = "gpwm", r = 10) {
  x <- as.numeric(x)  # garantir que    vetor num rico

  res <- tryCatch(
    cpBlockMax(x, method = metodo, r = r),
    error = function(e) return(NULL)
  )

  if (is.null(res)) {
    # em caso de erro, devolve NA
    return(list(pval = NA_real_, cpt = NA_integer_))
  }

  # p-valor para mudan a no shape (assumindo que est  na terceira
      posi  o)
  pval <- NA_real_
  if (!is.null(res$pvalues) && length(res$pvalues) >= 3) pval <-
     res$pvalues[3]

  # Estimador do ponto de mudan a
  cpt <- NA_integer_
  if ("stats.shape" %in% names(res) && length(res$stats.shape) > 0)
      {
    cpt <- as.integer(which.max(res$stats.shape))
  } else if ("cpts" %in% names(res) && length(res$cpts) > 0) {
    cpt <- as.integer(res$cpts[1])
  }

  return(list(pval = pval, cpt = cpt))
}

# ---------- 4. Simula  o para n vel emp rico, poder e
    precis o ----------
set.seed(123)
n <- 200
Nsim <- 2000
phi_vals <- c(0.2, 0.5, 0.8)
alpha <- 0.05
xi_const <- 0

# par metros para cpBlockMax em cada parte
r_level <- 1   # para n vel emp rico
r_power <- 1  # para poder
```

```r
nivel_emp <- numeric(length(phi_vals))
poder <- numeric(length(phi_vals))
precisao <- numeric(length(phi_vals))

for (i in seq_along(phi_vals)) {
  phi <- phi_vals[i]

  # ---------- N vel emp rico (sem mudan a) ----------
  rejeicoes <- 0
  for (j in 1:Nsim) {
    X <- sim_gev_ar1(n, phi = phi, xi = xi_const)
    teste <- rodar_teste(X, metodo = "gpwm", r = r_level)
    if (!is.na(teste$pval) && teste$pval < alpha) rejeicoes <-
       rejeicoes + 1
  }
  nivel_emp[i] <- rejeicoes / Nsim

  # ---------- Poder e precis o (com mudan a em xi) ----------
  rejeicoes <- 0
  acertos_localizacao <- 0
  lambda <- as.integer(n/2)
  raio <- 3  # crit rio  3

  for (j in 1:Nsim) {
    # mudan a de xi: ex.: 0.2 -> 0.4
    X <- sim_gev_ar1_change(n, phi = phi, xi1 = -0.2, xi2 = 0.2,
       lambda = lambda)
    teste <- rodar_teste(X, metodo = "gpwm", r = r_power)

    if (!is.na(teste$pval) && teste$pval < alpha) {
      rejeicoes <- rejeicoes + 1
      if (!is.na(teste$cpt) && abs(teste$cpt - lambda) <= raio) {
        acertos_localizacao <- acertos_localizacao + 1
      }
    }
  }

  poder[i] <- rejeicoes / Nsim
  precisao[i] <- acertos_localizacao / Nsim
}


# ---------- 6. Resultados ----------
res_df <- data.frame(phi = phi_vals,
     NivelEmpirico = nivel_emp,
    Poder = poder,
  Acerto = precisao)

print(res_df)
```