



**Cursos Integrados
em Vigilância em Saúde**

Curso —

**Análises de séries temporais
aplicadas à vigilância em saúde**

UNIVERSIDADE FEDERAL DE SANTA CATARINA

Reitor Irineu Manoel de Souza

Vice-Reitora Joana Célia dos Passos

Pró-Reitora de Pós-graduação Werner Kraus

Pró-Reitor de Pesquisa e Inovação Jacques Mick

Pró-Reitor de Extensão Olga Regina Zigelli Garcia

CENTRO DE CIÊNCIAS DA SAÚDE

Diretor Fabrício de Souza Neves

Vice-Diretora Rodrigo Otávio Moretti Pires

DEPARTAMENTO DE SAÚDE PÚBLICA

Chefe do Departamento Sheila Rúbia Lindner

Subchefe do Departamento Maria Cristina Marino Calvo

INSTITUTO TODOS PELA SAÚDE (ITPS)

Diretor-presidente Gerson Oliveira Penna

ASSOCIAÇÃO BRASILEIRA DE SAÚDE COLETIVA (ABRASCO)

Presidente Rômulo Paes de Sousa

EQUIPE DE PRODUÇÃO

João Henrique de Araujo Moraes

Denis de Oliveira Rodrigues

Laís Picinini Freitas

Wagner de Souza Tassinari

Oswaldo Gonçalves Cruz

Alexandra Crispim Boing

Antonio Fernando Boing



Curso

**Análises de séries temporais
aplicadas à vigilância em saúde**

**Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)**

Análises de séries temporais aplicadas à
vigilância em saúde [livro eletrônico] / João
Henrique de Araujo Morais ... [et al.] ;
organização Alexandra Crispim da Silva Boing ,
Antonio Fernando Boing. -- 1. ed. --
Florianópolis, SC : Associação Brasileira de
Saúde Coletiva, 2025.
PDF

Outros autores: Denis de Oliveira Rodrigues, Laís
Picinini Freitas, Wagner de Souza Tassinari, Oswaldo
Gonçalves Cruz
Bibliografia
ISBN 978-85-85740-12-2

1. Análise de séries temporais 2. Medicina
preventiva 3. Promoção da saúde 4. Saúde pública
5. Vigilância em saúde I. Moraes, João Henrique de
Araujo. II. Rodrigues, Denis de Oliveira.
III. Freitas, Laís Picinini. IV. Tassinari, Wagner de
Souza. V. Cruz, Oswaldo Gonçalves. VI. Boing,
Alexandra Crispim da Silva. VII. Boing, Antonio
Fernando.

25-308622.0

CDD-362.1068

Índices para catálogo sistemático:

1. Vigilância em saúde pública : Bem-estar social
362.1068

Maria Alice Ferreira - Bibliotecária - CRB-8/7964

Sumário

Análises de séries temporais aplicadas à vigilância em saúde	06
Módulo 1 - Introdução a séries temporais	07
O que são séries temporais?	07
Objetivos na análise de séries temporais.....	10
Tipos de séries temporais.....	11
Módulo 2 - Conceitos Fundamentais.....	15
Processo estocástico	15
Notação e nomenclatura	17
Estacionariedade.....	17
Pressuposto de independência e dependência serial	20
Tipos de dependência serial	22
Autocorrelação	22
Função de autocorrelação (ACF).....	25
Função de autocorrelação parcial (PACF).....	27
Componentes de uma série temporal	28
Tendência	28
Sazonalidade.....	31
Ciclo	33
Aleatoriedade ou ruído branco	34
Decomposição de séries temporais	34
Composição de séries temporais: modelo aditivo e modelo multiplicativo	37
Modelo Aditivo.....	37
Modelo Multiplicativo	39
Módulo 3 - Prática em R	40
Breve análise descritiva da série	46
Analizando seus componentes	48
Módulo 4 - Transformações e suavizações	59
Box-Cox.....	61
Diferenciação.....	63
Médias móveis	65
Kernel	68
Loess	70
Splines	71
Prática em R: Transformações e suavizações	72
Suavizações	80
Kernel	83
Lowess	84
Splines	86
Módulo 5 - Introdução à modelagem de séries temporais	88
Modelo Box & Jenkins - o modelo ARIMA	89
Termo AR (autorregressivo).....	91
Termo MA (médias móveis).....	93
Termo I (diferenciação)	95
Ajustando nosso primeiro modelo	97
Ajuste automático de modelos ARIMA	104
E a sazonalidade?	106
Nosso primeiro modelo SARIMA.....	107
Módulo 6 - Diagnóstico do modelo e avaliação da qualidade do ajuste	110
Análise de resíduos	113
Modelos de previsão	116
Módulo 7 - Modelando séries temporais com GAM	119
Definição de um modelo GAM	127
Tipos de Splines.....	128
Módulo 8 - GAM em Séries Temporais.....	130

Análises de séries temporais aplicadas à vigilância em saúde

Na rotina da vigilância em saúde é importante entender como certos problemas de saúde são distribuídos ao longo do tempo. Por meio de análises de **dados distribuídos no tempo**, que chamaremos de **séries temporais**, podemos observar, por exemplo, que algumas doenças respiratórias são mais comuns em meses de inverno, enquanto algumas doenças transmitidas por mosquitos costumam aumentar durante o verão. Tais análises, ditas análises temporais, também permitem compreender de forma relativamente simples a tendência dos problemas de saúde ao longo de um período de tempo. Por exemplo, a vigilância pode identificar uma tendência de hospitalizações por gastroenterite causadas por um potencial aumento de casos de Hepatite A, acendendo um alerta para identificar possíveis problemas de abastecimento de água. As análises temporais também podem ser utilizadas para ajudar a planejar o futuro, tornando-se essenciais para a tomada de decisões.

Neste curso você vai conhecer as séries temporais, seus objetivos, suas classificações e como podem ser analisadas no contexto da vigilância em saúde.

Ao final deste curso, você será capaz de:

1. entender os principais conceitos de séries temporais;
2. entender e realizar transformações, suavizações e testes de autocorrelação em séries temporais;
4. realizar modelagem estatística de séries temporais.

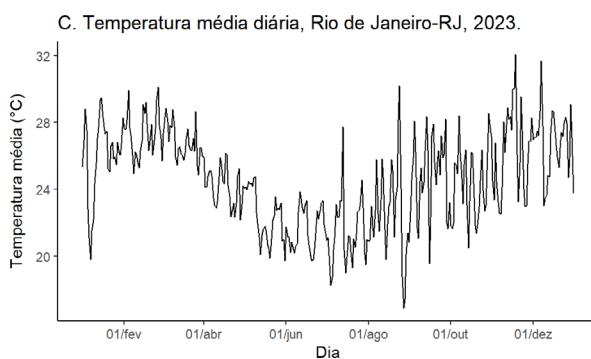
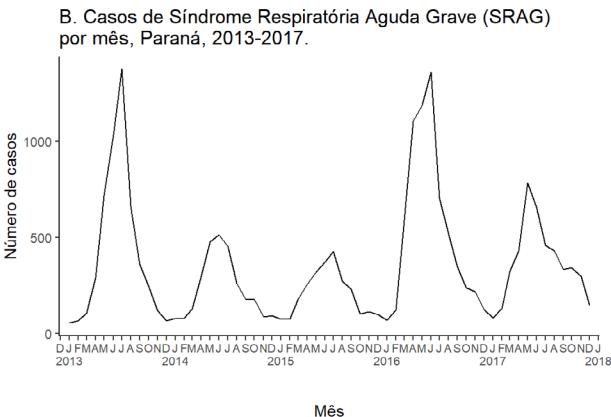
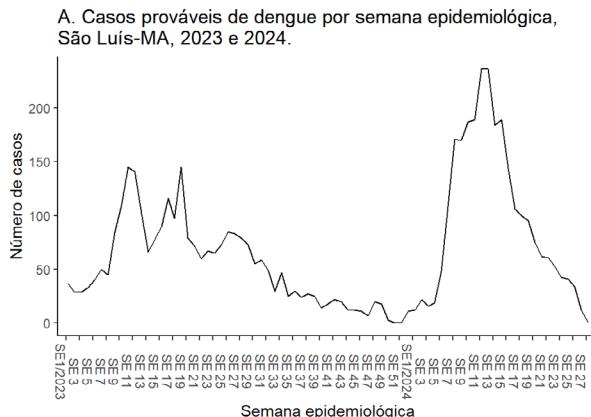
Módulo 1 - Introdução a séries temporais

O que são séries temporais?

Séries temporais podem ser definidas como um conjunto de valores organizados em ordem cronológica, ou seja, com informações sobre **o momento** em que esses valores foram observados. Geralmente, essa informação temporal segue uma **frequência**, ou **unidade de tempo**, podendo ser mensurações diárias, semanais, mensais, anuais, entre outras. Estamos acostumados a ver séries temporais cotidianamente: ao acompanharmos a variação do preço dos alimentos nos jornais, o aumento da temperatura ao longo das décadas, ou o número de casos de dengue ao longo das semanas.

A maneira mais comum de representarmos graficamente uma série temporal é por meio de um gráfico de linhas, onde o eixo X representa a variável temporal, e o eixo Y, a variável de interesse. Observemos na **Figura 1** quatro exemplos de séries temporais que têm aplicação na vigilância em saúde pública. Perceba que cada série trata de uma variável de interesse diferente (A: número de casos prováveis de dengue, B: número de casos de Síndrome Respiratória Aguda Grave, C: temperatura média, e D: número de desastres ambientais). Elas estão em diferentes unidades de tempo (A: por semana epidemiológica, B: por mês, C: por dia, e D: por ano) e representam diferentes locais (A: município de São Luís-MA, B: estado do Paraná, C: município do Rio de Janeiro-RJ, D: Brasil).

Figura 1: Exemplos de séries temporais.



Agora, vamos interpretar as séries temporais dispostas na Figura 1.

A Figura 1A mostra o número de casos prováveis de dengue por semana epidemiológica em São Luís, Maranhão, desde o início de 2023 até a semana 27 de 2024. Em ambos os anos, podemos observar aumento acentuado nos casos por volta da semana epidemiológica 5, atingindo picos por volta da semana 15, seguidos por diminuições graduais. Esse comportamento temporal de aumento expressivo em determinado período do ano pode estar relacionado a condições climáticas ou, ainda, ao impacto de medidas de controle da dengue. Também podemos avaliar que a curva de casos de 2023 foi mais “achatada”, com mais de um pico e casos sendo observados por maior período de tempo comparado com 2024. Em 2024, a curva apresentou comportamento mais “explosivo”, ou seja, o número de casos aumentou mais acentuada e rapidamente, formando apenas um pico, que decresce também rapidamente. Uma curva de casos mais “achatada” pode significar uma persistência da transmissão da doença ao longo do tempo, enquanto uma curva mais “explosiva” pode significar um surto ou epidemia. Essas informações são importantes pois têm implicações diretas nas escolhas das medidas de prevenção e enfrentamento mais adequadas ao cenário.

A Figura 1B mostra o número mensal de casos de Síndrome Respiratória Aguda Grave (SRAG) no Paraná de 2013 a 2018. O monitoramento de casos de SRAG é crucial para a detecção precoce de surtos de doenças respiratórias. Neste exemplo, podemos identificar os meses do ano de maior e menor magnitude de casos. Essa informação permite a tomada de decisões para preparar o serviço de saúde para absorver casos de SRAG nos períodos de alta atividade, por exemplo. Além disso, também podemos identificar mudanças no padrão esperado, como por exemplo picos incomumemente altos como os observados em 2013 e 2016. Um aumento de casos inusual pode gerar um alerta para uma possível introdução de novo vírus ou cepa.

Já a Figura 1C mostra a variação da temperatura média diária na cidade do Rio de Janeiro ao longo de 2023. Observamos que, em geral, a temperatura média varia entre 20 e 30 °C durante o ano. Podemos observar uma tendência de diminuição da temperatura média à medida que o ano avança para os meses de inverno, voltando a subir depois de agosto. Dados de temperatura têm se tornado cada vez mais importantes na vigilância em saúde, visto que o aumento da temperatura pode elevar a incidência de doenças transmitidas por vetores, como a dengue, e o calor extremo pode elevar o risco de morte em pessoas com doenças cardiovasculares.

Finalmente, a Figura 1D representa o número de desastres ambientais registrados no Brasil por ano no período de 2000 a 2023. Há tendência crescente no número de desastres ao longo dos anos, com aumento acentuado especialmente após 2019. Esse padrão pode estar associado a mudanças climáticas, aumento da urbanização desordenada, ou a outros fatores ambientais e até mesmo econômicos e políticos. No contexto da vigilância em saúde, esses dados são fundamentais para planejar respostas rápidas a eventos que podem impactar gravemente a saúde pública.

Você percebeu a riqueza de informações que são possíveis de extrair de uma série temporal? Esses exemplos ilustram como diferentes tipos de dados temporais são essenciais para a tomada de decisões na vigilância em saúde, permitindo o monitoramento contínuo e a resposta rápida a ameaças emergentes. Na sequência do curso, mais exemplos serão apresentados.

Objetivos na análise de séries temporais

Podemos ter diferentes objetivos ao analisar séries temporais no contexto da vigilância em saúde. Frequentemente, adotamos um objetivo **descritivo**, buscando identificar se as séries apresentam uma tendência ao longo do tempo ou se seguem algum padrão que se repete em determinados períodos. Além disso, podemos investigar se mudanças em uma série temporal, como o aumento da cobertura vacinal, impactam outra série, como o número de casos de sarampo. Acompanhe o Quadro 1 a seguir que apresenta um breve resumo dos principais objetivos da análise de séries temporais na vigilância em saúde.

Quadro 1. Principais objetivos da análise de séries temporais na vigilância em saúde e seus exemplos práticos.

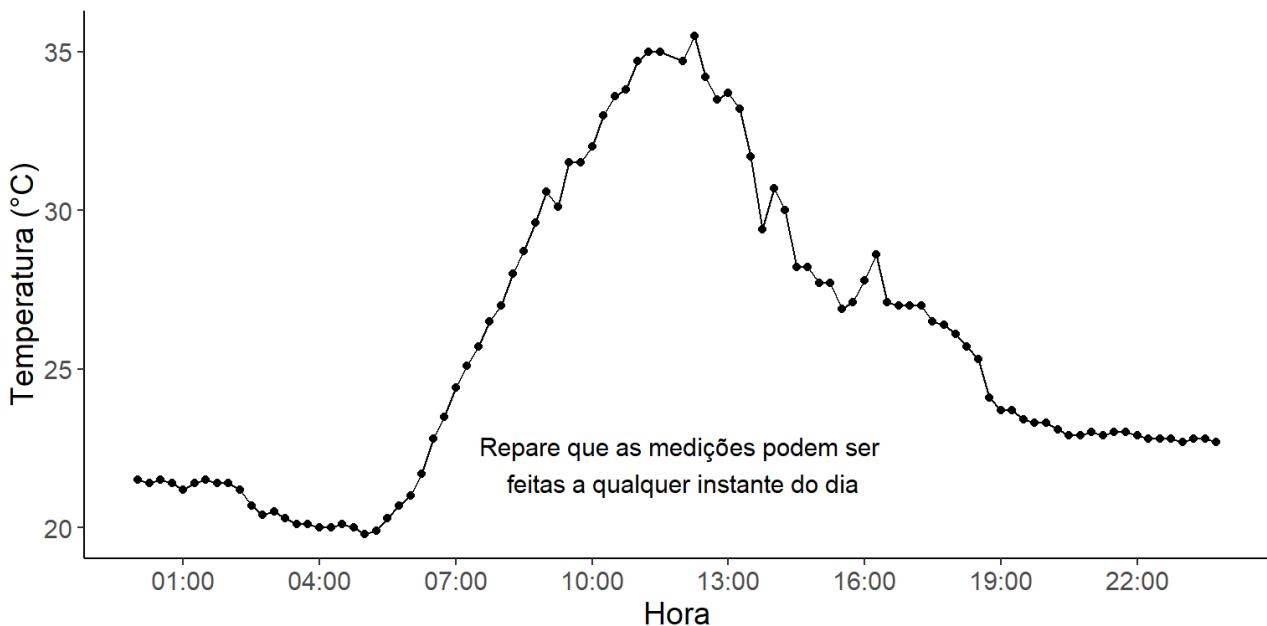
Objetivo	Exemplo
Descrição: Verificar a distribuição da variável no tempo, buscando a existência de tendências ou padrões que se repetem em determinados períodos. Representações gráficas, como gráficos de linhas, histogramas e boxplots, são úteis para uma análise exploratória descritiva.	Identificar uma tendência na série de diagnósticos de AIDS; verificar se há padrões que se repetem na série de casos de dengue, fornecendo subsídios para planejamento de intervenções.
Gerar hipóteses: Investigar possíveis associações temporais entre variáveis.	O aumento na série de cobertura vacinal está associado a uma queda na série de casos de sarampo? Essa observação pode levantar hipóteses sobre o impacto da cobertura vacinal, que devem ser investigadas com estudos adicionais.
Classificação: Identificar padrões comuns entre diferentes séries.	Identificar padrões temporais comuns entre diferentes séries, como a correlação entre a série de leishmaniose tegumentar e a visceral.
Avaliação de intervenção: Avaliar o impacto em uma série de medidas de controle a uma série de eventos de interesse.	Avaliar o impacto de medidas de controle, como campanhas de vacinação, durante uma epidemia de uma doença específica.
Monitoramento: Detectar variações incomuns no comportamento de séries temporais em tempo real, utilizando técnicas como a detecção de anomalias.	Identificar uma mudança na tendência dos casos de Síndrome Respiratória Aguda Grave, o que pode sinalizar ajustes nos sistemas de saúde como a necessidade de aumento do número de leitos de internação.
Predição (forecast): Estimar o comportamento futuro de uma determinada série. Pode ser de curto ou longo prazo.	Predição do número de casos esperados em uma epidemia, a partir das estimativas de novos casos da doença.
Atualização (nowcast[*]): Estimar o valor atual de uma série temporal, corrigindo atrasos nas notificações dos sistemas de informação.	Modelos de nowcasting que permitem uma interpretação mais precisa do cenário atual e possibilitam respostas mais ágeis, como no caso do atraso das notificações de doenças infecciosas.

* O termo **nowcasting** aqui empregado corresponde à abordagem sob o olhar da epidemiologia e dos sistemas de informação, no sentido de levar em consideração o atraso existente na rotina de inserção desses dados e tentar corrigi-los. O sentido é diferente, portanto, da definição de nowcasting sob a ótica meteorológica, que se refere a uma previsão no futuro de curto prazo.

Tipos de séries temporais

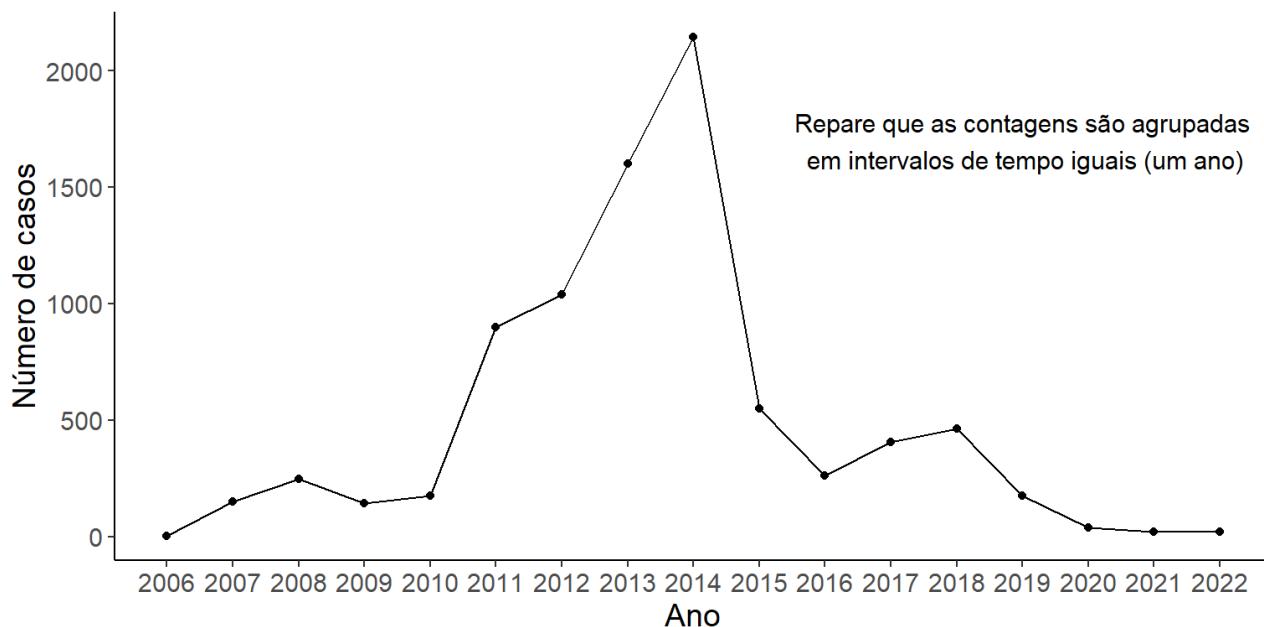
As séries temporais podem ser classificadas em diferentes tipos de acordo com a natureza do que se está medindo e a forma como os dados são coletados. As séries **contínuas** representam medições que podem ser feitas em qualquer instante de tempo. Elas são úteis para monitorar variáveis que mudam continuamente, como, por exemplo, variáveis climáticas. Nas últimas décadas, tem sido acompanhado o impacto das mudanças climáticas na saúde humana. Um desses acompanhamentos é da temperatura em uma estação meteorológica, onde sensores registram valores a cada momento (Figura 2).

Figura 2: Exemplo de série temporal contínua: Temperatura (°C) medida em 04/10/2023 na estação meteorológica de São Cristóvão, Rio de Janeiro-RJ.



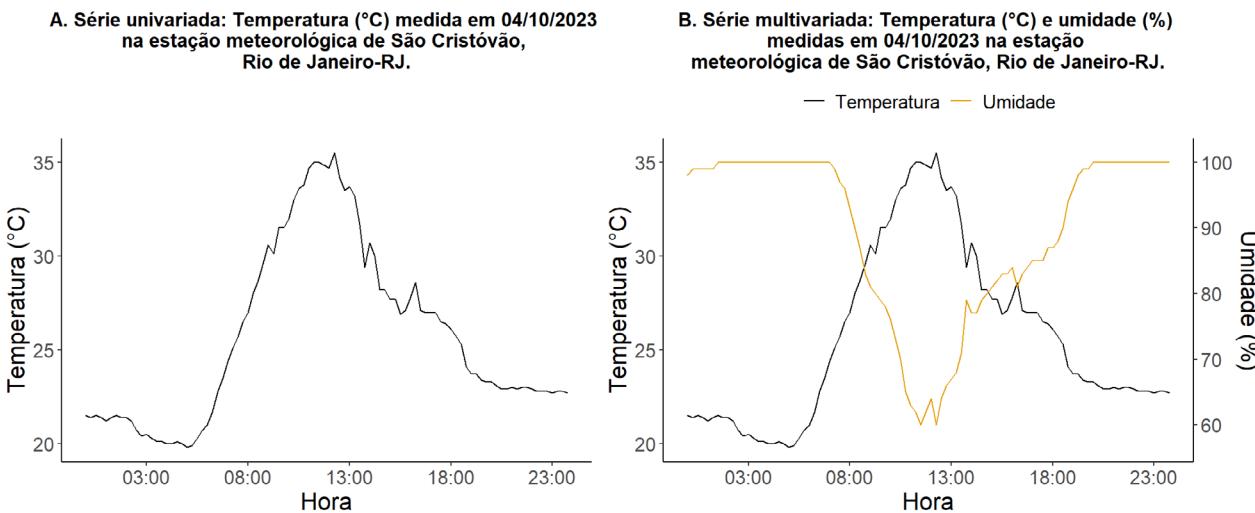
Já as séries **discretas** representam observações obtidas a partir da contagem de ocorrências de um fenômeno agregadas em intervalos de tempo definidos e separados em partes regulares, como dias, semanas, meses ou anos. Um exemplo clássico é o número de casos confirmados de uma doença a cada ano em uma localidade, como podemos ver na Figura 3.

Figura 3: Exemplo de série temporal discreta: Número de casos confirmados de Coqueluche por ano, Estado de São Paulo, 2006-2023.



Na rotina do profissional da vigilância em saúde, muitas vezes, a análise é realizada utilizando variáveis diferentes para a mesma localidade e para o mesmo período de tempo. Nesse caso, as séries temporais são classificadas como séries multivariadas, pois possuem mais de uma variável para o mesmo conjunto de observações no tempo. A temperatura aferida na estação meteorológica de São Cristóvão no Rio de Janeiro pode ser classificada como uma série **univariada**, por conter apenas uma variável (Figura 4A). Entretanto, a mesma estação meteorológica coleta outras informações, como a umidade do ar. Caso utilizemos tanto a temperatura quanto a umidade coletadas nesta mesma estação meteorológica no mesmo período de tempo, podemos chamar essa série de multivariada (Figura 4B).

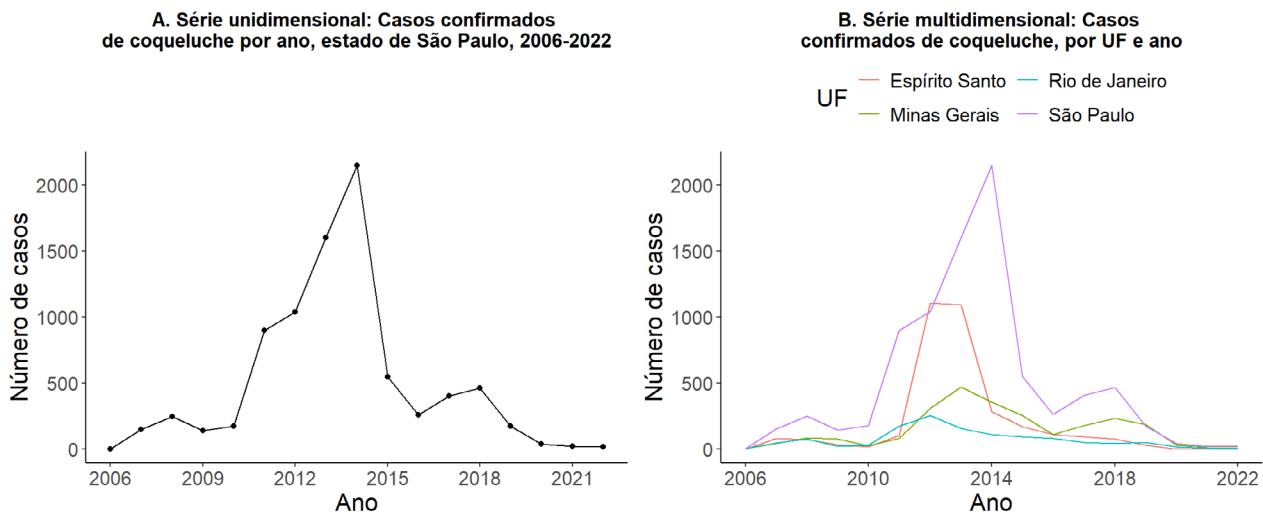
**Figura 4: Exemplo de série temporal univariada (A)
e de série temporal multivariada (B).**



Quando a análise precisa ser realizada para a mesma variável, mas com diferentes dimensões (além da dimensão de tempo), como, por exemplo, a localização do fenômeno, utilizamos as séries **multidimensionais**. Essas séries permitem monitorar a contagem de casos de uma mesma doença no mesmo intervalo de tempo, mas em diferentes regiões ou municípios. Com isso, podemos comparar o comportamento da doença em diversas localidades e avaliar como fatores locais podem influenciar as variações observadas.

Na Figura 5A temos a série temporal de casos confirmados de coqueluche no Estado de São Paulo, de 2006 a 2022, representando uma única dimensão (tempo). Observa-se aumento significativo de casos a partir de 2010, atingindo o pico em 2014, seguido de uma redução gradual até 2022. Já a Figura 5B mostra uma série multidimensional, com os casos confirmados de coqueluche por Unidade Federativa (UF) (Espírito Santo, Minas Gerais, Rio de Janeiro, São Paulo) ao longo do mesmo período da Figura 5A (2006-2022). Na Figura 5B é possível comparar as trajetórias da coqueluche entre diferentes estados. São Paulo e Minas Gerais se destacam com um pico acentuado de casos em 2014, enquanto os outros estados apresentam curvas mais moderadas e estáveis.

Figura 5: Exemplos de séries temporais unidimensional (A) e multidimensional (B).



Agora que já entendemos o que são as séries temporais e como podem ser classificadas, é importante destacar que, para extrair o máximo de informação desses dados, precisamos compreender alguns conceitos de estatística. Sabemos que a palavra “estatística” pode soar intimidadora para quem não está acostumado, mas aqui vamos tratar de conceitos simples, focados em ajudar no dia a dia da vigilância em saúde. A boa notícia é que muitos dos métodos estatísticos já estão implementados em ferramentas como o R, o que facilita muito o processo de análise. Vamos lá?

Módulo 2 - Conceitos Fundamentais

Processo estocástico

Quando falamos em séries temporais no contexto da vigilância em saúde, estamos lidando com dados que mudam ao longo do tempo, como o número de casos de uma doença ou a temperatura. Esses dados são influenciados por diversos fatores, muitos dos quais são imprevisíveis ou incertos. Para lidar com essa incerteza, usamos um conceito da estatística chamado **processo estocástico**.

Um processo estocástico pode ser entendido como um sistema que evolui ao longo do tempo de acordo com leis probabilísticas. Por exemplo, imagine que você está observando a variação de um fenômeno, como o número de casos de dengue, ao longo dos meses. Cada vez que você mede esse número, é como se estivesse tirando uma foto de uma realidade que pode mudar a cada instante. Essa variação pode ser influenciada por inúmeros fatores, como clima, ações de controle e até eventos inesperados e não podemos prever com grande exatidão o que vai acontecer a cada dia. Por isso, dizemos que essa variação no número de casos segue um “processo estocástico”, pois depende de elementos aleatórios, ou seja, que não conseguimos controlar totalmente.

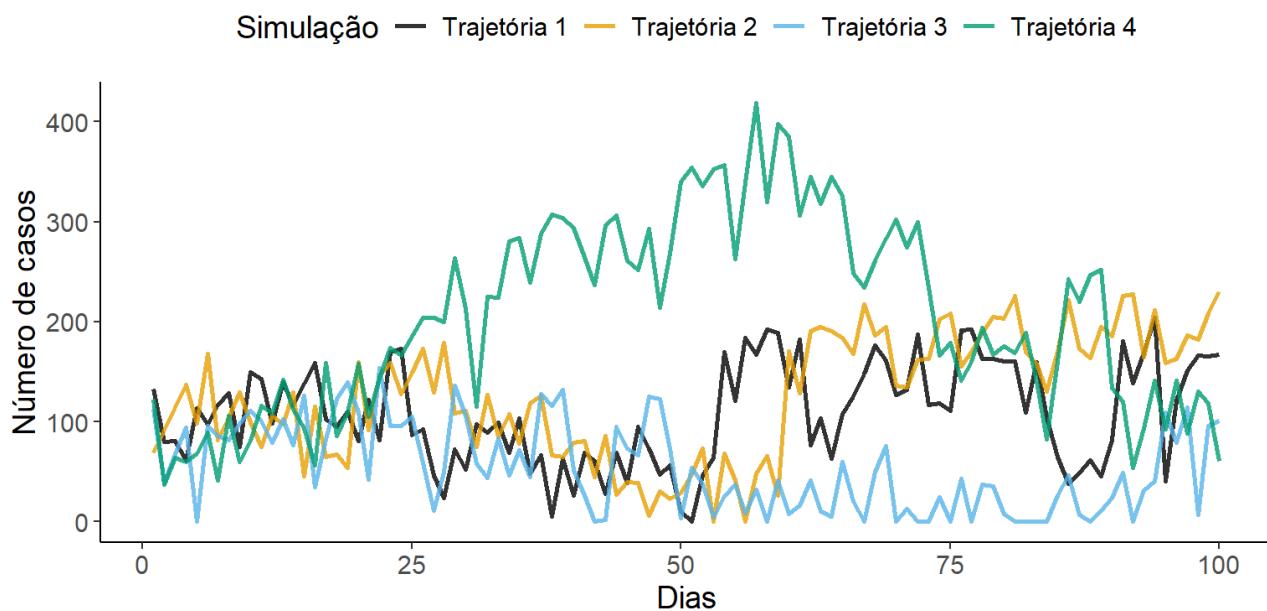
Podemos pensar em um processo estocástico de duas formas:

- Várias possíveis evoluções do mesmo fenômeno. Por exemplo, dependendo das condições, o número de casos de uma doença pode seguir diferentes trajetórias ao longo do tempo;
- Cada ponto no tempo representa um valor que poderia ter sido diferente, de acordo com certas probabilidades. Assim, o valor que observamos hoje é apenas uma das muitas possibilidades que poderiam ter acontecido.

Quando analisamos uma série temporal, estamos observando uma dessas possíveis trajetórias, que chamamos de “realização” de um processo estocástico.

Para ajudar a entender esse conceito, em vez de olhar apenas uma série temporal de casos de dengue, podemos simular várias possíveis trajetórias que poderiam ocorrer, cada uma representando uma evolução diferente (Figura 6). Isso é importante, pois como citamos anteriormente, existem muitas incertezas que podem influenciar a evolução de uma doença. Essas simulações nos ajudam a entender as diferentes formas que uma epidemia pode seguir e nos permitem nos preparar para diferentes cenários.

Figura 6: Simulação de múltiplas trajetórias: Casos diários de uma doença.



Com essas simulações, podemos mostrar que, em um cenário real de vigilância, mesmo que o número de casos varie de forma imprevisível no dia a dia, conseguimos criar cenários possíveis. Essas múltiplas trajetórias ajudam as equipes de saúde pública a se preparam para diferentes cenários futuros, como um aumento rápido de casos, uma estabilização ou uma queda mais lenta.

Na sequência, continuaremos conhecendo conceitos da estrutura das séries temporais. Vamos lá?

Notação e nomenclatura

As séries temporais podem ser matematicamente representadas por funções do tipo:

$$Z_t = f(\text{tempo}, a)$$

na qual:

- Z_t é definido, por convenção, como o valor da variável observável Z no tempo t , e
- a é a componente aleatória.

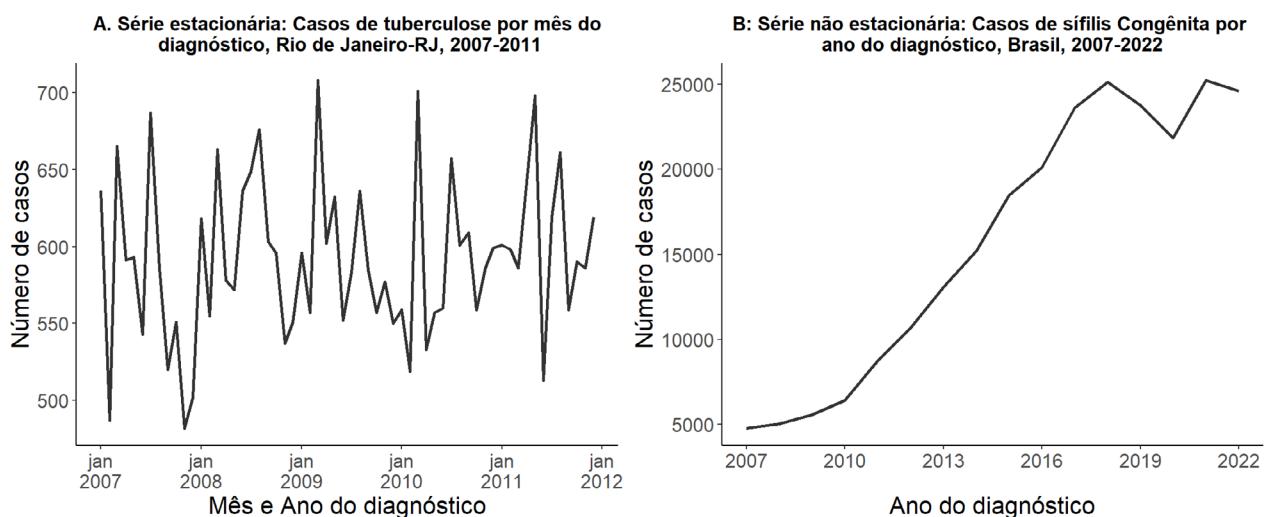
Estacionariedade

Como mencionado anteriormente, as técnicas estatísticas ajudam a interpretar a incerteza nas análises de séries temporais. Um ponto importante é que a maioria dessas técnicas supõe que a série temporal analisada possui características estáveis ao longo do tempo. Essas são as principais características de uma série **estacionária**. Uma série temporal é considerada estacionária quando seus valores flutuam aleatoriamente ao redor de uma média constante, e a variação em torno dessa média (variância) também permanece constante ao longo do tempo. Isso reflete um comportamento equilibrado e estável da série.

Na prática, muitas séries temporais que encontramos apresentam algum tipo de não estacionariedade, como, por exemplo, uma tendência crescente ou decrescente ao longo do tempo. A não estacionariedade pode inviabilizar a aplicação de técnicas estatísticas, exigindo que transformemos os dados para estabilizar a média e a variância.

Na Figura 7A, vemos o número de casos de tuberculose ao longo dos anos, de 2007 a 2011, no Município Rio de Janeiro. A série temporal flutua ao redor de uma média constante, com variações, mas sem um padrão claro de tendência crescente ou decrescente a longo prazo. Ao contrário da série de tuberculose, observamos os casos de sífilis congênita no Brasil, de 2007 a 2022 na Figura 7B. Esta série apresenta um padrão de crescimento ao longo do tempo, com um aumento contínuo nos números de casos ano a ano.

Figura 7: Exemplo de série estacionária (A) e série não estacionária (B).



O modelo mais simples de uma série temporal estacionária pode ser descrito da seguinte forma:

$$Z_t = \mu + a_t$$

na qual:

- μ é a média do processo temporal, que é constante ao longo do tempo,
- a_t é o componente aleatório, chamado de **ruído branco** em análises de séries temporais, que representa as flutuações imprevisíveis em torno da média.

Podemos, portanto, denominar de ruído branco uma sequência de variáveis aleatórias independentes e identicamente distribuídas, com média zero e variância constante, sendo essencial para modelar a incerteza presente em qualquer série temporal estacionária.

A estacionariedade de uma série temporal pode ser classificada em dois tipos:

- **primeira ordem:** A média é constante ao longo de todo o período analisado. Isso significa que os valores flutuam aleatoriamente em torno de uma média fixa, sem apresentar uma tendência de aumento ou diminuição.
- **segunda ordem:** Além da média constante, a variância também permanece constante ao longo do tempo. Esse tipo de estacionariedade é mais rigoroso e garante que a série temporal não apenas oscile ao redor de uma média fixa, mas que a magnitude dessas oscilações (variância) e a relação entre valores em diferentes pontos no tempo (autocovariância) permaneçam estáveis.

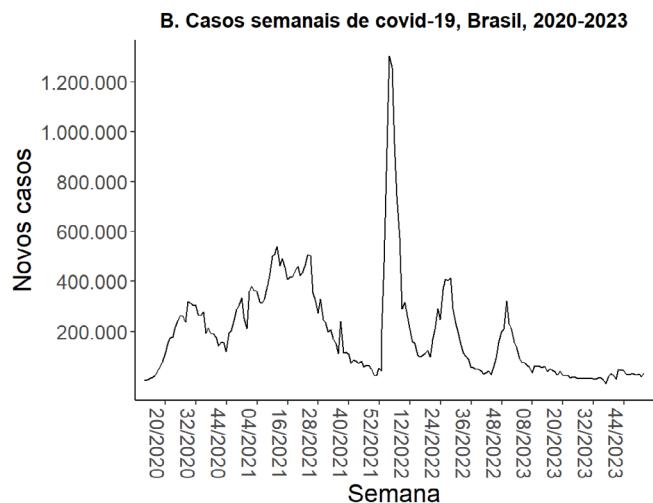
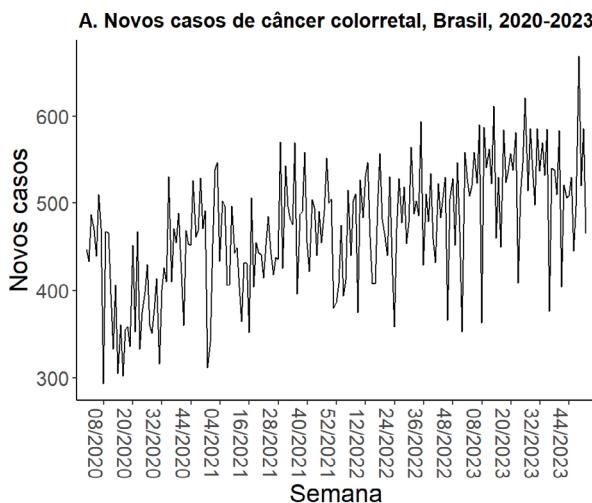
Vimos que uma série temporal estacionária tem média e variância constantes ao longo do tempo, além de um componente aleatório representado pelo ruído branco. No entanto, em muitas análises estatísticas, assumimos que uma observação não influencia as observações subsequentes. Vamos entender esse pressuposto e ver mais alguns exemplos sobre no próximo capítulo.

Pressuposto de independência e dependência serial

Quando trabalhamos com modelos e técnicas estatísticas, como modelos lineares generalizados, é comum termos como pressuposto a **independência** dos dados. Isso significa que assumimos que uma determinada observação não influencia as observações seguintes. Ao analisar, por exemplo, a incidência de casos de câncer de colorretal em uma determinada população, não esperamos que o fato de um indivíduo ser diagnosticado com o câncer influencie um outro indivíduo não relacionado a ele. Ou até mesmo, não esperamos que o número de casos novos em um determinado mês dependa do número de casos nos meses anteriores. Ou seja, podemos dizer que esse determinado agravo **não** possui dependência serial.

Contudo, para outras doenças e agravos - especialmente agravos contagiosos - esperamos que essa dependência exista. Podemos citar o caso da covid-19: quando um indivíduo é diagnosticado com a doença, há uma alta probabilidade que outros indivíduos próximos dele possam se tornar novos casos também. Em termos de série temporal, podemos afirmar que o número de casos de covid-19 em uma determinada semana **depende fortemente** do número de casos da semana anterior e das semanas anteriores. Portanto, a série de covid-19 é uma série que possui **dependência serial**. Agora, vamos analisar graficamente esses dois exemplos: o caso de uma série com independência (câncer colorretal) e uma série com dependência serial (covid-19).

Figura 8: Dependência serial: comparação entre séries semanais de casos de câncer colorretal (A) e covid-19 (B) no Brasil. 2020-2023.



Vemos que na Figura 8A não parece haver uma dependência forte do número de novos casos de câncer em uma determinada semana em relação às anteriores. O inverso é observado para a covid-19 (Figura 8B): há uma dependência do número de casos entre as semanas, de forma com que vemos explosões eventuais no número de casos. Podemos dizer, portanto, que para a covid-19 há indícios de forte dependência serial. Vamos ver mais propriedades sobre esse fenômeno de dependência e como quantificá-la.

Tipos de dependência serial

Quando não há autocorrelação, ou seja, quando os valores da série temporal são independentes uns dos outros, dizemos que a série é **independente**. Por outro lado, quando há **dependência**, essa pode ser de dois tipos:

- **Memória curta:** Quando os valores da série influenciam os valores mais imediatamente seguintes, e seu efeito desaparece rapidamente com o tempo. Podemos citar como exemplo as doenças muito contagiosas ou “explosivas”, como o sarampo, influenza ou covid-19. Para esses agravos, os números de casos em uma determinada semana, por exemplo, dependem fortemente do número de casos nas semanas anteriores, e não tanto do número de casos nos meses ou anos anteriores.
- **Memória longa:** Nesse caso a dependência desaparece lentamente ao longo do tempo, ou seja, os valores do passado influenciam pontos muito adiante no tempo. Esse é o caso para doenças de grande latência, como a hanseníase.

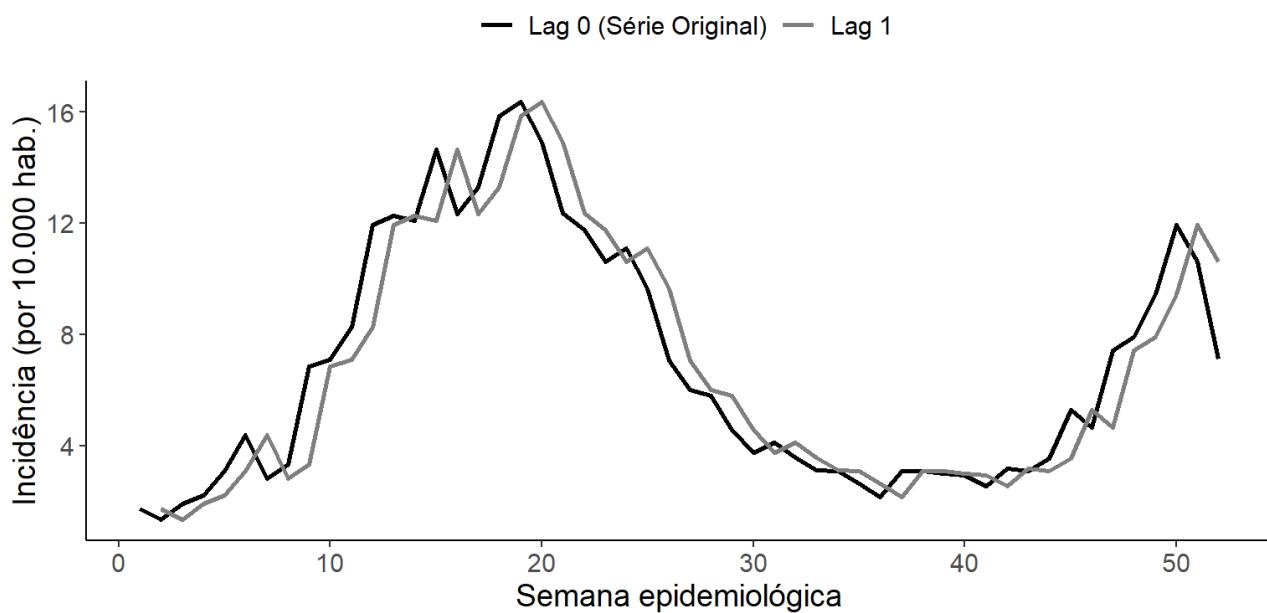
Autocorrelação

Para entender como a dependência serial se comporta em uma série temporal, utilizamos a autocorrelação. A autocorrelação mede o grau de correlação entre os valores de uma série em diferentes momentos ao longo do tempo. Esses momentos são chamados de **lags**, também conhecidos como defasagens ou atrasos.

Por exemplo, um lag de uma semana significa que estamos comparando o número de casos desta semana com o número de casos da semana anterior (um atraso de uma semana). Um lag de duas semanas compara os casos desta semana com os casos de duas semanas atrás, e assim por diante.

Um exemplo prático é analisar a série de incidência de dengue no Município do Rio de Janeiro em 2023 e comparar a incidência de uma semana com a incidência da semana anterior ou de duas semanas anteriores, para verificar se há uma relação entre esses valores em diferentes períodos. Acompanhe a Figura 9.

Figura 9: Incidência de dengue por semana epidemiológica, Rio de Janeiro-RJ, 2023.

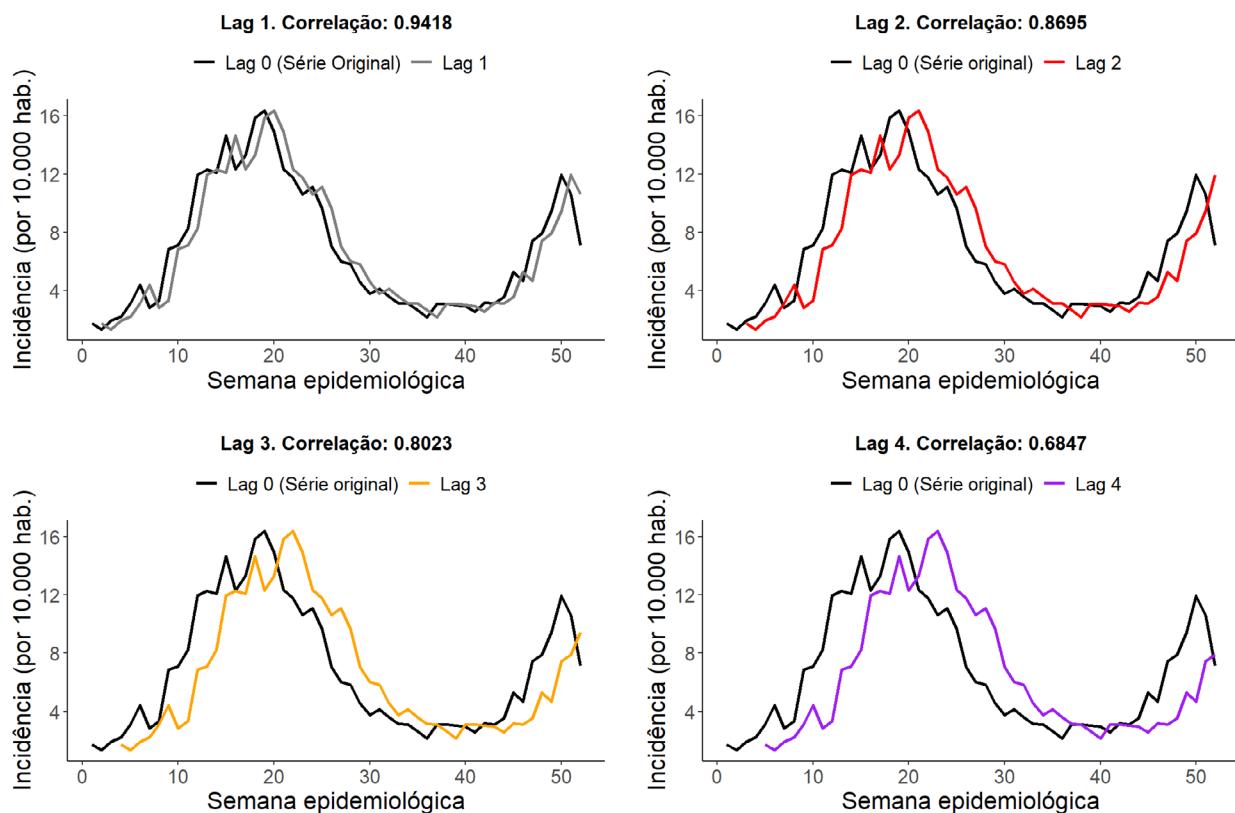


Podemos perceber pela Figura 9 que há um deslocamento da série original representando o lag de uma semana na incidência de dengue no município do Rio de Janeiro em 2023. Para medir o valor da correlação da série original (linha preta) com a série defasada (linha cinza), podemos calcular o ρ . O ρ (rho) representa o coeficiente de autocorrelação em um determinado lag, indicando a força e a direção da relação entre os valores de uma série temporal em momentos diferentes no tempo.

No exemplo anterior, o valor da correlação é $\rho_1 = 0,9418$. Chamamos esse valor de **autocorrelação de primeira ordem**. De forma similar, podemos expandir essa definição para autocorrelação de **h -ésima ordem** (ρ_h), onde calculamos a correlação da série com ela mesma defasada em h lags (Figura 10).



Figura 10: Comparação de séries temporais de incidência de dengue com diferentes lags e coeficientes de autocorrelação.



Função de autocorrelação (ACF)

Na Figura 10 temos que quando $lag = 1$, a correlação entre a série original e sua desfasagem em uma semana possui um alto valor de correlação ($\rho_1 = 0,9418$). Contudo, à medida que o lag aumenta, a correlação começa a enfraquecer. Assim, após um período de 4 semanas, a dependência serial dos casos de dengue é um pouco mais fraca, embora ainda presente ($\rho_4 = 0,6847$).

Dessa forma, temos que ρ_0 é sempre 1, pois a correlação da série com ela mesma, sem defasagem (lag 0), é uma correlação perfeita. E para os demais lags, ρ_h , temos que $-1 \leq \rho_h \leq 1$. Assim, temos que o valor de ρ pode variar entre -1 e 1:

$\rho = 1$ significa que há uma correlação positiva perfeita. Se o valor de uma observação aumenta, a próxima (ou uma observação após um determinado lag) também aumenta na mesma proporção.

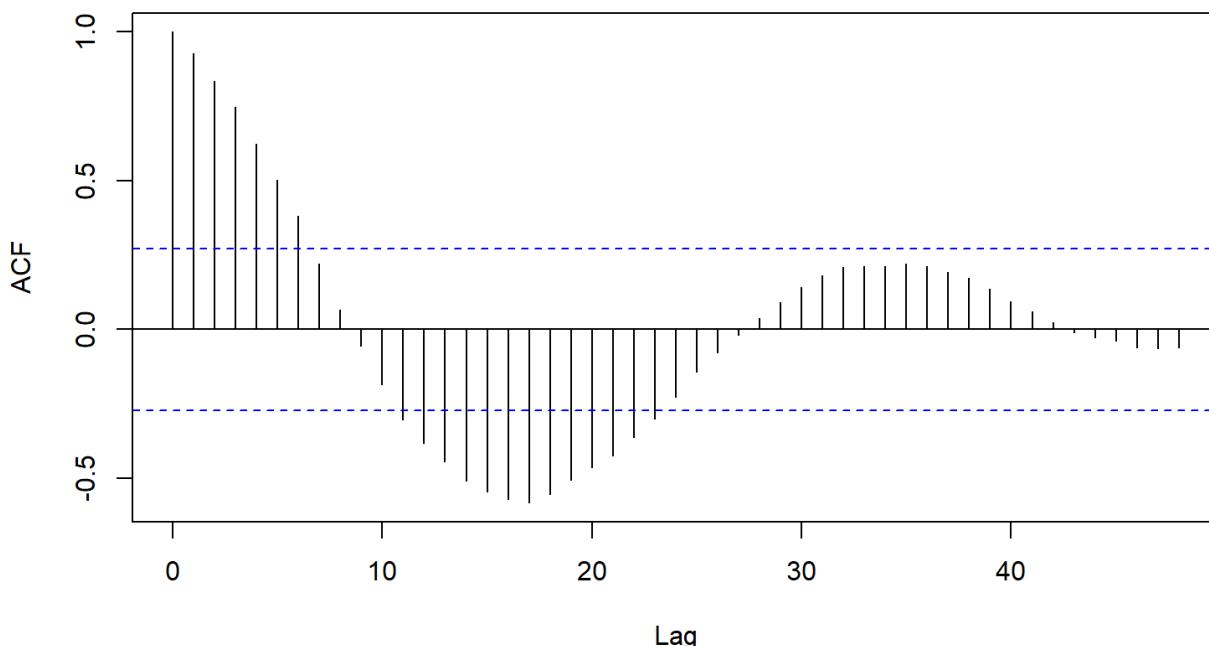
$\rho = 0$ significa que não há correlação entre os valores. Isso indica que os valores em diferentes momentos são independentes.

$\rho = -1$ significa que há uma correlação negativa perfeita. Se o valor de uma observação aumenta, a próxima observação diminui de forma proporcional.

Esse conjunto de correlações, $\rho_h = \rho_0, \rho_1, \rho_2, \dots, \rho_t$, que consiste das correlações da série com ela mesma defasada em 0, 1, 2 lags, é chamado de **função de autocorrelação** da série Z_t .

Uma forma tradicional de observar a função de autocorrelação de uma série é por meio de um gráfico de **correlograma**. Nele podemos visualizar o valor da função de autocorrelação (ACF) da série (no eixo y) em cada lag (no eixo x). Observe na Figura 11.

Figura 11: Função de autocorrelação (ACF) para a série de incidência semanal de dengue no Rio de Janeiro-RJ, 2023.



Na Figura 11 observamos que o valor mais alto de autocorrelação ocorre no lag 0, onde a autocorrelação é 1, representando a correlação perfeita da série com ela mesma (sem defasagem). As barras seguintes mostram os valores de autocorrelação da série em diferentes lags: lag 1, lag 2, e assim sucessivamente, até o lag 48. A partir do lag 1, percebemos que o valor da autocorrelação começa a diminuir progressivamente e, por volta do lag 9, inverte seu sinal, indicando uma correlação negativa.

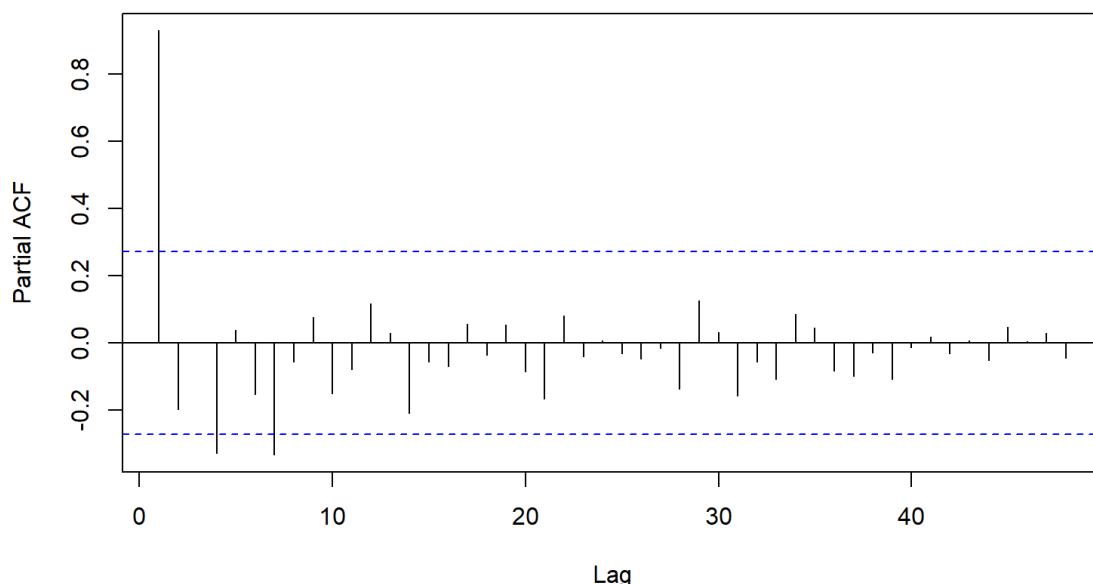
As linhas pontilhadas delimitam uma área de significância estatística. Valores de autocorrelação que ultrapassam essas linhas (para cima ou para baixo) indicam que a correlação para aquele lag é estatisticamente significativa, ou seja, é improvável que tenha ocorrido apenas por acaso. Em outras palavras, quando a autocorrelação ultrapassa essas linhas, isso sugere que há uma relação consistente e relevante entre os valores da série naquele lag específico. Valores dentro das linhas pontilhadas, por outro lado, indicam que a autocorrelação é pequena o suficiente para não ser considerada estatisticamente significativa.

Função de autocorrelação parcial (PACF)

A Função de Autocorrelação Parcial (PACF) também é uma forma de visualizar a autocorrelação da série entre seus lags. A diferença é que, ao comparar a série com ela mesma defasada em h lags, retira-se o efeito dos lags $h - 1$, $h - 2$, e assim por diante. Ou seja, a influência dos lags intermediários é removida, mostrando somente uma correlação “limpa”, apenas do lag de interesse.

Veja na Figura 12 a diferença ao observarmos a PACF para a mesma série de incidência de dengue no município do Rio de Janeiro em 2023.

Figura 12: Função de autocorrelação parcial (PACF) para a série de incidência semanal de dengue no Rio de Janeiro-RJ, 2023.



Para esse gráfico não é exposta a correlação no lag 0, somente a partir do lag 1. A autocorrelação no lag 1 é a mais forte, sugerindo uma alta dependência dos valores da série com os valores no período imediatamente anterior. As correlações já não decaem lentamente como acontecia no gráfico da ACF, justamente porque as influências intermediárias são removidas. Ao invés disso, vemos algumas “batidas” pontuais no sinal da correlação. Além do primeiro lag, vêem-se correlações ligeiramente relevantes no lag 4 e no lag 7, porém no sentido negativo.

Agora que já temos uma visão sobre série temporal e seus pressupostos, vamos entender os elementos que compõem uma série temporal. Vamos lá!

Componentes de uma série temporal

Já vimos diversos exemplos de uso de séries temporais no contexto da vigilância em saúde, seja na vigilância epidemiológica, no monitoramento de doenças, seja na vigilância ambiental, com monitoramento de variáveis climáticas. Independente da área da vigilância, para fazer uma análise eficaz é importante entender que uma série temporal é composta por diferentes componentes que nos ajudam a identificar padrões de comportamento, tendências de aumento ou queda, e variações sazonais.

Esses componentes são fundamentais para prever o comportamento futuro de uma doença, planejar ações de controle e monitorar surtos com mais precisão. A série temporal pode ser dividida em **componentes sistemáticos** e **componentes não sistemáticos**. A seguir, vamos explorar os principais componentes de uma série temporal que são relevantes para a vigilância em saúde.

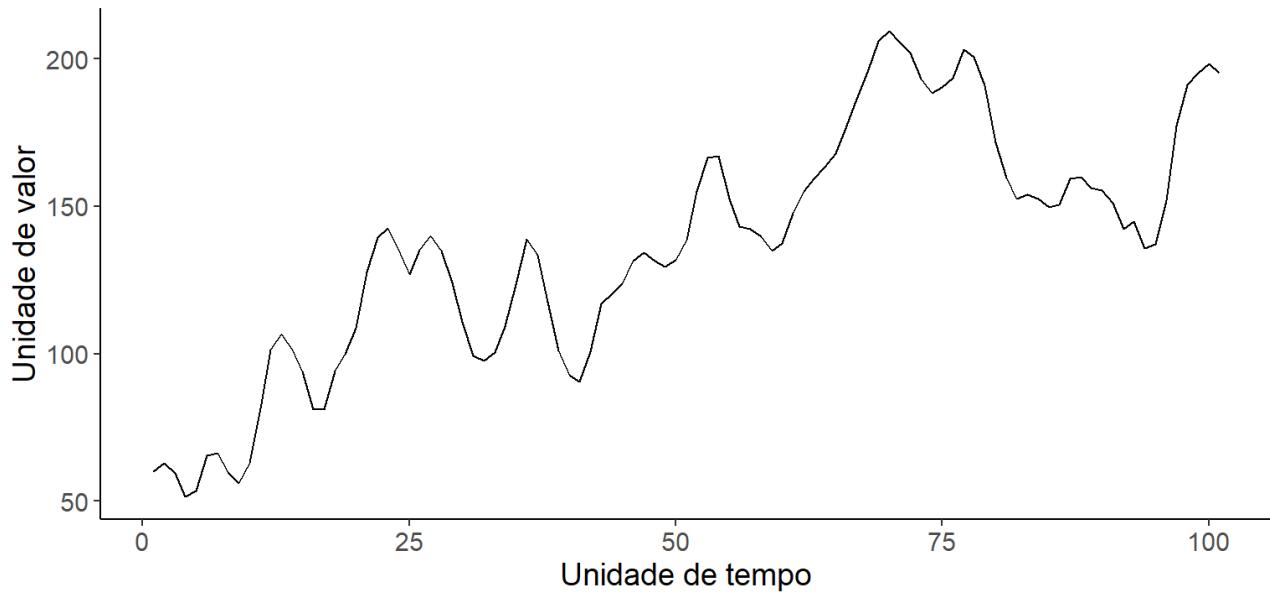
Vamos, primeiro, iniciar com os componentes sistemáticos. Os componentes sistemáticos apontam movimentos regulares, ou seja, que apresentam um determinado padrão. Esses podem ser **tendência, sazonalidade e ciclo**. Vamos entender brevemente seus conceitos e visualizar exemplos de cada um.

Tendência

A tendência é o componente que indica a **direção global** dos dados ao longo do tempo. Seu uso em análises na vigilância pode indicar se o fenômeno que está sendo monitorado (uma doença, por exemplo) está aumentando, diminuindo, ou se mantém estável em períodos mais longos. Dessa forma, saber se os dados estão seguindo uma trajetória crescente, decrescente ou estável pode ser útil para entender se o fenômeno está se agravando, o que pode exigir ações públicas.

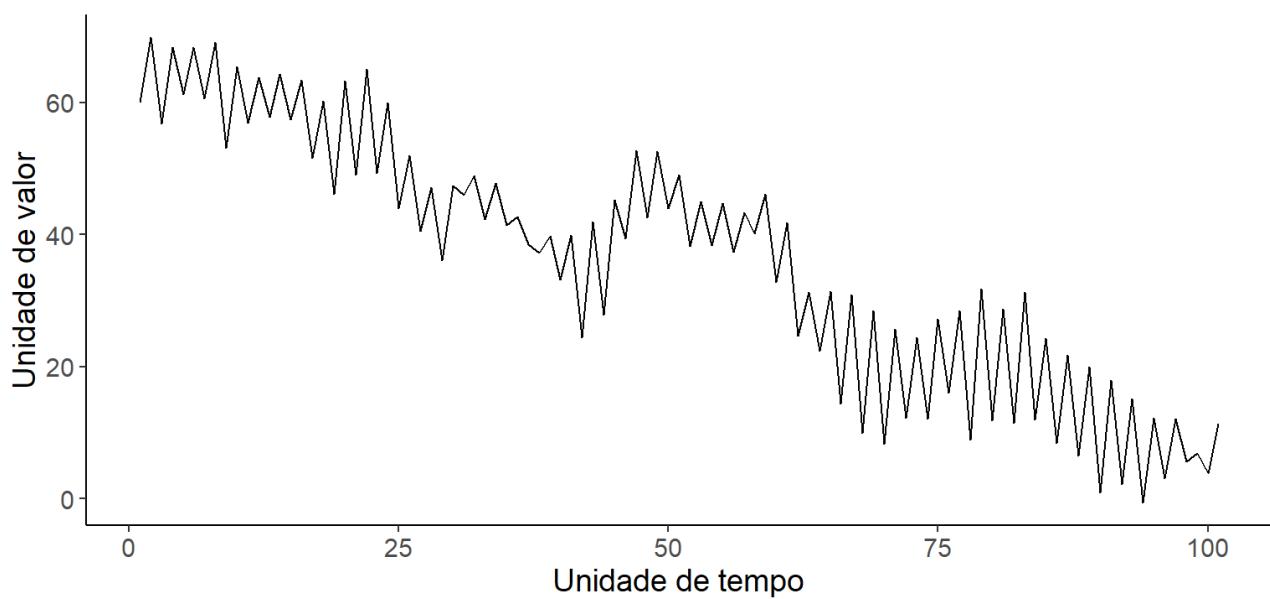
A seguir, veremos três exemplos de séries fictícias apenas para que você possa entender o padrão geral de uma tendência de crescimento, queda e ausência de tendência. Mas não se preocupe pois veremos mais à frente exemplos práticos no R, inclusive como decompor uma série e visualizar seus componentes separadamente. Agora, na Figura 13 temos um exemplo de uma série temporal com uma tendência de crescimento.

Figura 13: Exemplo de série temporal com tendência de crescimento.



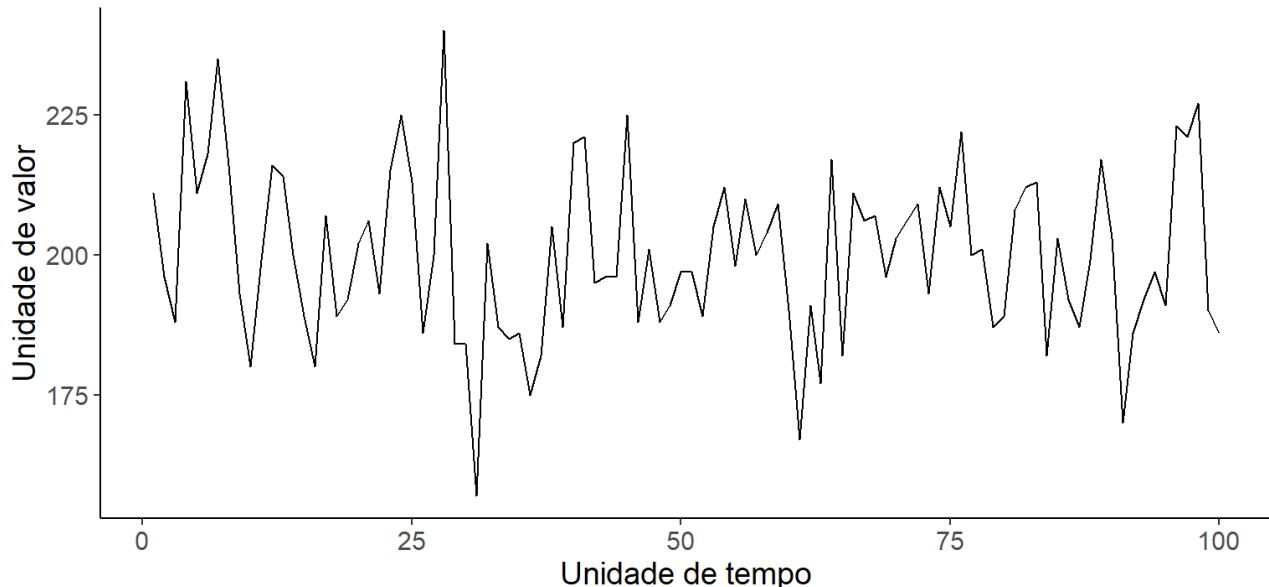
Vamos agora observar outro exemplo, de uma série que também possui tendência, mas cujo padrão é decrescente, com queda ao longo do tempo. Acompanhe na Figura 14:

Figura 14: Exemplo de série temporal com tendência de queda.



Retomando o conceito de séries estacionárias que vimos anteriormente, uma característica importante dessas séries é a ausência de tendência. Em uma série estacionária, a média dos valores permanece estável ao longo do tempo, sem apresentar um movimento claro de aumento ou diminuição a longo prazo. Veja um exemplo na Figura 15.

Figura 15: Exemplo de série temporal estacionária, sem tendência.



Nesse caso a série é, portanto, formada apenas por algum ou alguns dos demais componentes que veremos a seguir.

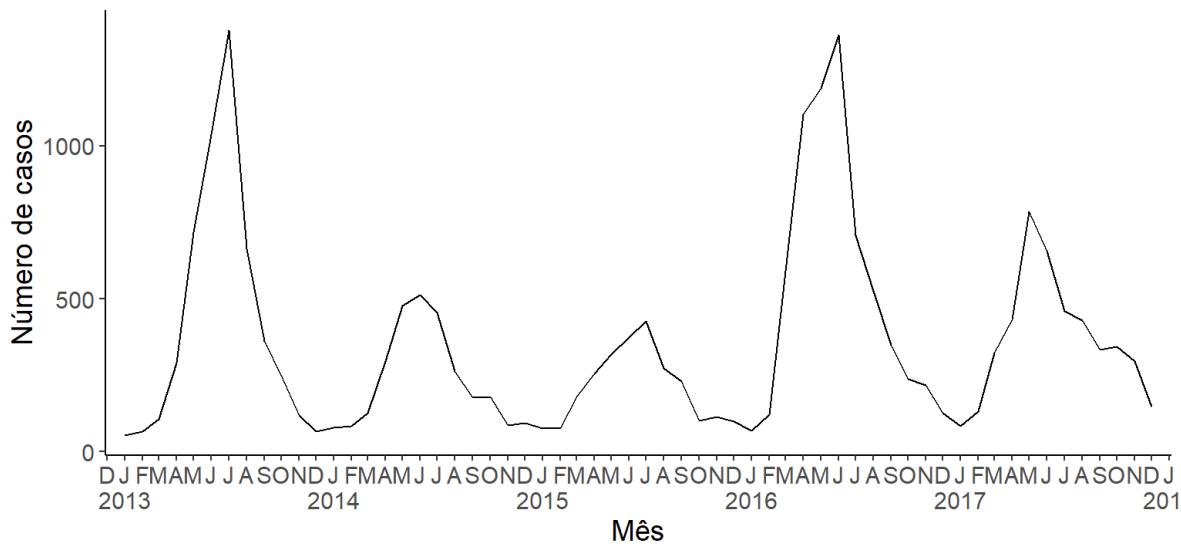
Sazonalidade

Um outro componente de uma série temporal que se refere à repetição de padrões que ocorrem em intervalos regulares e com uma determinada frequência é a **sazonalidade**. Esses padrões podem estar relacionados ao calendário, como estações do ano, meses ou semanas, mas nem sempre têm a ver diretamente com o comportamento de doenças. Por exemplo, a temperatura geralmente apresenta um ciclo diário, com aumento ao longo do dia e queda após o entardecer.

No contexto de séries temporais, a sazonalidade representa esses ciclos regulares ao redor de uma tendência. Esse comportamento pode ou não estar ligado a eventos naturais ou ao calendário. No caso das doenças, como na Síndrome Respiratória Aguda Grave (SRAG), a sazonalidade pode ser observada em padrões repetidos de aumento de casos em meses mais frios, como os meses de inverno, seguidos por uma queda. Esse padrão é o que chamamos de sazonalidade em séries temporais.

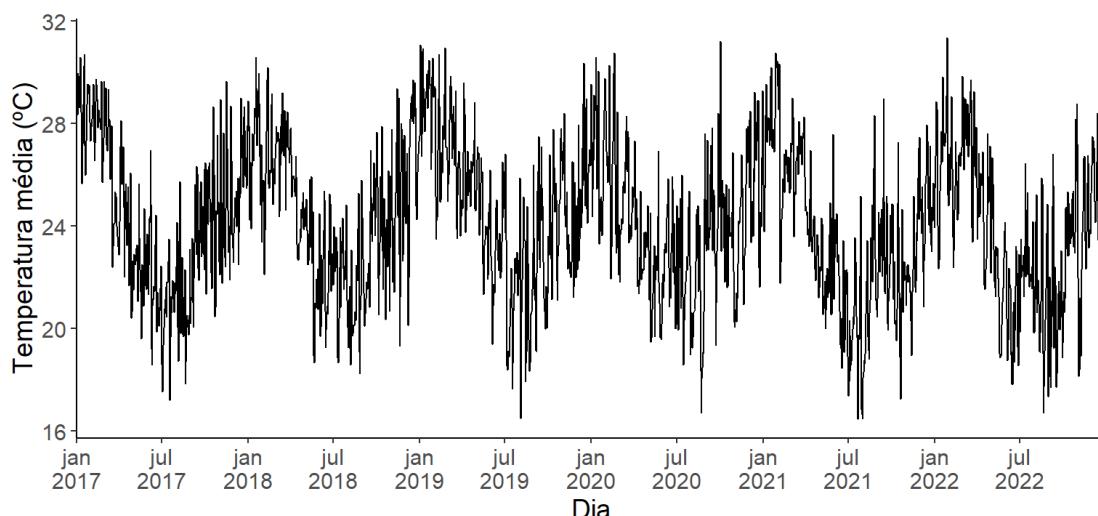
Lembram-se da série que vimos anteriormente do número de casos mensais de SRAG no Paraná, de 2013 a 2017? Retomemos esse exemplo na Figura 16 para observarmos que essa série se comporta de maneira sazonal, com aumento de casos nos meses de inverno e diminuição nos meses subsequentes. Esse padrão se repete ano após ano.

Figura 16: Exemplo de sazonalidade: Casos de Síndrome Respiratória Aguda Grave (SRAG) por mês, Paraná, 2013-2017.



Vejamos agora outro exemplo de sazonalidade. Como citamos antes, a temperatura é uma variável que possui padrões diários e que alternam entre as estações do ano. Veja na Figura 17 a temperatura média diária do Município do Rio de Janeiro entre os anos 2017 e 2022. Perceba que há picos nos meses de janeiro e fevereiro e quedas nos meses de julho e agosto.

Figura 17: Exemplo de sazonalidade: Temperatura média diária, Rio de Janeiro-RJ, 2017-2022.



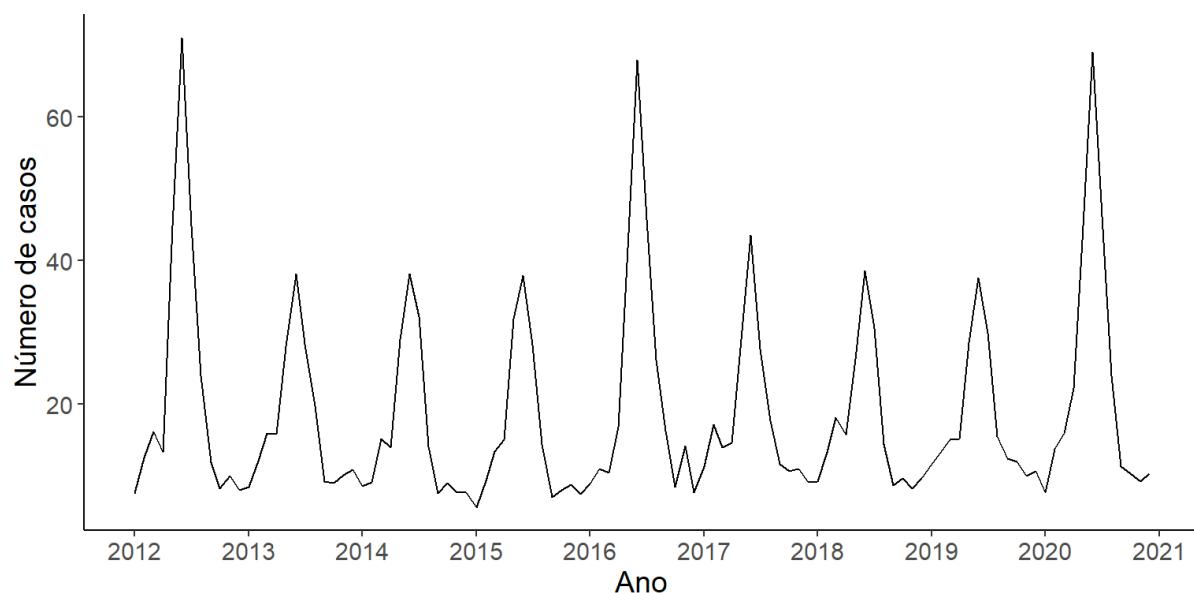
Ciclo

Os ciclos também representam oscilações regulares em uma série temporal, mas se diferenciam da sazonalidade pela frequência e natureza da repetição. Enquanto a sazonalidade está frequentemente ligada a eventos com periodicidade mais previsível, como mudanças sazonais (verão, inverno), os ciclos tendem a ocorrer com menos frequência e não estão necessariamente vinculados ao calendário.

Na vigilância em saúde, os ciclos podem ser mais difíceis de serem observados, mas podem representar padrões mais amplos relacionados a eventos como surtos epidêmicos que acontecem em intervalos irregulares. Por exemplo, imagine que uma epidemia de uma determinada doença ocorra em um município a cada quatro anos (Figura 18). Isso seria um ciclo, com um padrão de aumento de casos a cada quatro anos. Mas também temos a sazonalidade anual da doença, que pode apresentar mais casos no inverno.

Assim, podemos observar dois padrões coexistindo: a sazonalidade anual e um ciclo de longo prazo que se repete, por exemplo, a cada quatro anos, resultando em um aumento significativo de casos nesses anos específicos.

Figura 18: Exemplo de ciclo de um comportamento de uma doença entre 2012 e 2021.



Aleatoriedade ou ruído branco

Agora vamos abordar o componente **não sistemático** de uma série temporal. Enquanto os componentes sistemáticos, como tendência, sazonalidade e ciclo, representam padrões regulares que podem ser descritos e modelados, as séries temporais também contêm um **componente aleatório**, também conhecido como **ruído branco**.

Por exemplo, podemos observar que o número de casos de uma determinada doença tem uma **tendência** de aumento ao longo dos anos, e que há mais casos durante o inverno devido à **sazonalidade**. No entanto, mesmo sabendo disso, nunca podemos prever com exatidão quantos casos irão ocorrer em um mês ou semana específica. Isso ocorre porque existe sempre uma flutuação aleatória — fatores que não conseguimos explicar ou prever completamente. Esse componente aleatório é o ruído branco, uma variabilidade que não segue um padrão claro e que resulta de uma combinação de influências imprevisíveis ou impossíveis de serem mensuradas.

Dessa forma, descreve-se esse componente como aleatório ou ruído branco, ou seja, o que resta da série temporal quando removemos os componentes sistemáticos (a sua tendência e sua sazonalidade). Ele reflete as flutuações aleatórias que não podem ser atribuídas a um padrão conhecido.

Além disso, há séries temporais que podem ser compostas quase exclusivamente por ruído branco, quando não apresentam estruturas claras de tendência, ciclo ou sazonalidade. Essas séries são completamente imprevisíveis e não têm padrões sistemáticos que possamos identificar.

Veja outros exemplos típicos de eventos resultante dos efeitos de múltiplas causas que dificilmente conseguem ser previstos com exatidão:

- Secas;
- Enchentes;
- Terremotos;
- Crise política (guerras);
- Conflitos socioeconômicos.

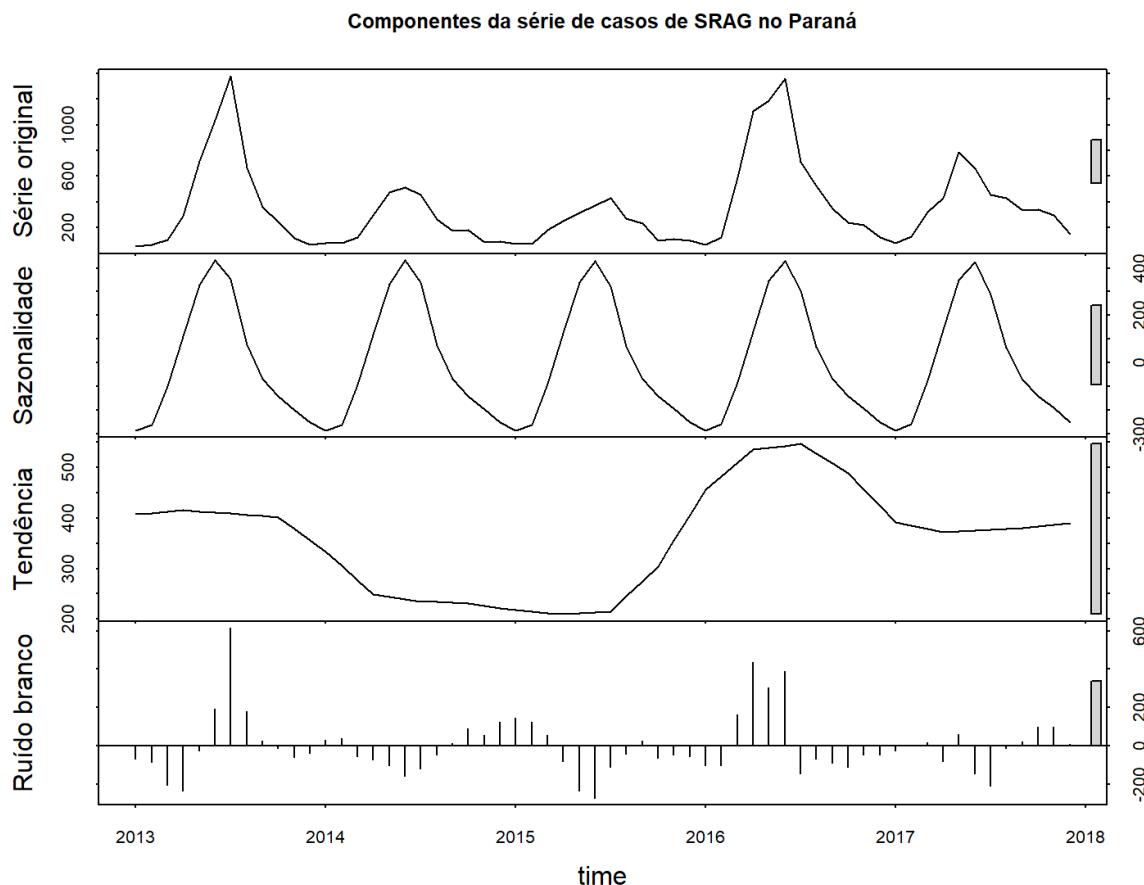
Decomposição de séries temporais

Agora que conhecemos os componentes de uma série temporal, podemos visualizar a série como uma combinação desses elementos. Em outras palavras, uma série temporal pode ser decomposta em diferentes partes: a tendência, a sazonalidade, os ciclos, e as variações aleatórias representadas pelo ruído branco (ou componente aleatório).

Cada série temporal pode ter ou não esses componentes, e em intensidades diferentes. Algumas séries apresentam uma tendência muito evidente, enquanto outras têm uma sazonalidade marcante, como um padrão repetido de aumento e queda em certos períodos do ano. No entanto, muitas séries temporais são compostas, em maior parte, pelo componente aleatório, ou seja, pelo ruído branco, que representa as variações imprevisíveis.

Entender a decomposição da série temporal nos ajuda a entender quais desses componentes estão mais presentes e como eles influenciam o comportamento da série ao longo do tempo. Vamos, novamente, analisar a série de casos mensais de SRAG no Paraná entre 2013 e 2017 (Figura 19). Mas agora veremos os componentes da série também.

Figura 19: Exemplo de decomposição de série temporal: Casos de Síndrome Respiratória Aguda Grave (SRAG) por mês, Paraná, 2013-2017.



A Figura 19 apresenta a série original e sua decomposição nos componentes de sazonalidade, tendência e ruído branco, conforme discutido anteriormente. Essa visualização é fundamental para entender como cada componente contribui para a dinâmica da série temporal.

À direita dos gráficos, vemos uma barra de escala, que serve como indicador da importância relativa de cada componente. Quando a barra de escala de um componente é semelhante à da série original, isso indica que esse componente é muito importante para a explicação da série. Por outro lado, se a barra de escala de um componente for muito maior, significa que os valores daquele componente são relativamente pequenos, ou seja, ele contribui menos para a variação total da série.

Nesse exemplo, podemos observar que tanto a sazonalidade quanto o ruído branco têm contribuições relevantes e similares para explicar a série de SRAG no Paraná. Em contraste, a tendência é um componente mais fraco, o que é visível pela menor influência na série. De fato, ao analisarmos a série original, o que se destaca é a presença clara da sazonalidade, sem um comportamento de tendência acentuado.

Composição de séries temporais: modelo aditivo e modelo multiplicativo

Até agora, vimos que uma série temporal é composta por diferentes componentes, como tendência, sazonalidade, ciclos e ruído branco. Esses componentes formam a base de qualquer série temporal e nos ajudam a entender o comportamento da variável ao longo do tempo. No entanto, a maneira como esses componentes se combinam para formar a série pode variar.

Existem duas formas principais de combinar esses componentes: de forma **aditiva** ou **multiplicativa**. A diferença entre esses dois tipos de modelos está na relação entre os componentes. Em um modelo aditivo, os componentes são somados para formar a série, enquanto em um modelo multiplicativo, os componentes interagem multiplicativamente, ou seja, o impacto de um componente depende diretamente dos outros.

A escolha entre um modelo aditivo ou multiplicativo depende do comportamento da série. A seguir vamos entender em mais detalhes essas duas abordagens.

Modelo Aditivo

No modelo aditivo, os termos são reunidos por meio de uma soma de seus termos. Dada uma série temporal , temos:

$$Z_t = T_t + S_t + \alpha_t$$

Onde:

- T_t é o componente de tendência;
- S_t a sazonalidade e;
- α_t o termo aleatório, ou ruído branco.

Dessa forma, a série é constituída por uma **adição** dos componentes. Geralmente é o caso quando a variação sazonal parece ser constante, não aumentando ou diminuindo sua amplitude ao longo do tempo. Veremos de forma mais clara em um exemplo abaixo (Figura 20).

Figura 20: Exemplo de série temporal aditiva.



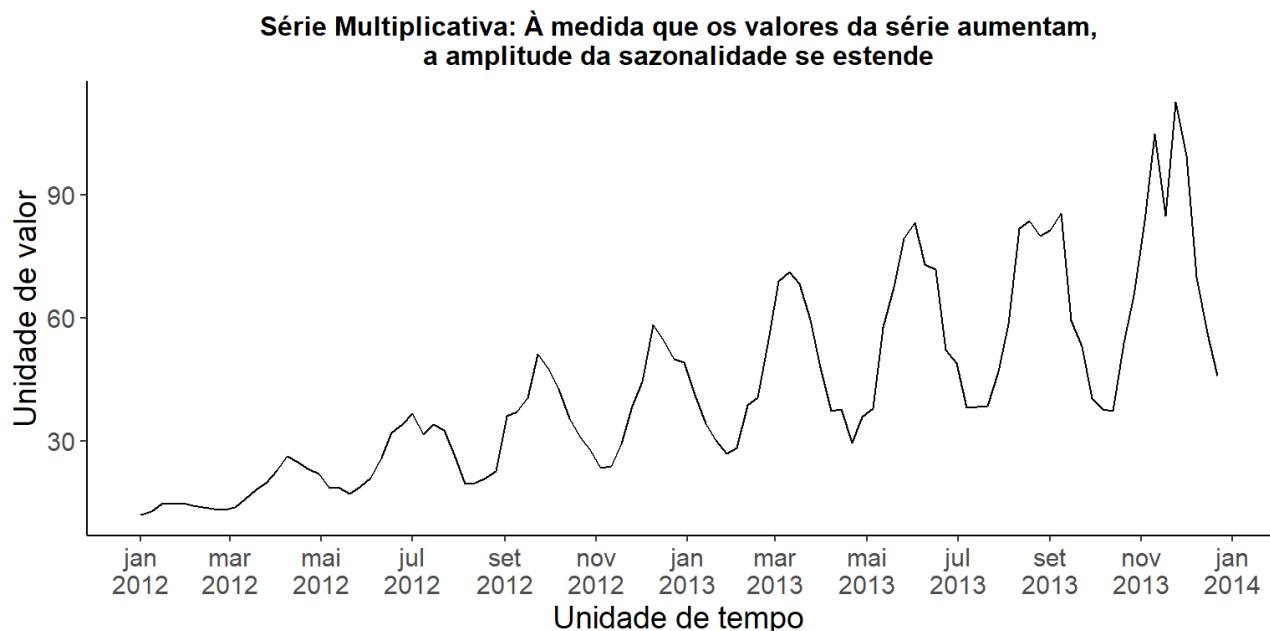
Modelo Multiplicativo

Já no modelo multiplicativo, a série é derivada não de uma adição, mas de uma **multiplicação** de seus termos. Ou seja, para a série Z_t agora temos:

$$Z_t = T_t * S_t * \alpha_t$$

Nessa composição, sugere-se que a sazonalidade da série varia juntamente com a tendência, tendo sua amplitude ampliada ou reduzida ao longo do tempo.

Figura 21: Exemplo de série temporal multiplicativa.



Ufa! Quantos conceitos, não é mesmo? No próximo módulo vamos, finalmente, utilizar o R para manipular alguns dados e praticar a análise de séries temporais. Usaremos dados reais, incluindo nos exemplos a forma de obtê-los.

Módulo 3 - Prática em R

A partir desse momento vamos aprender a realizar análises de séries temporais de interesse para a vigilância utilizando a linguagem R. Caso você ainda não tenha conhecimento intermediário de R, pode fazer nosso curso introdutório. Ele está disponível online e gratuitamente em <<https://sites.google.com/view/cursos-de-vigilancia/curso-inicial>>. Recomendamos que tenha instalada a versão mais recente do R em seu computador.

Seguindo no curso, vamos utilizar como exemplo a série temporal mensal de casos de SRAG no Paraná. Os dados foram obtidos do **INFOGRIPÉ**, e consolidados em um arquivo csv. Podemos ler o arquivo diretamente da web utilizando a função `read.csv` do R:

```
# Carregando o arquivo csv
srag_pr_mes ← read.csv(file = "https://raw.githubusercontent.com/joaohmoraes/
dados-mod-temp/main/csv/srag_PR_mes.csv")
```

```
# Vendo os dados
head(srag_pr_mes)
#>   ano mes   n
#> 1 2013   1  54
#> 2 2013   2  66
#> 3 2013   3 108
#> 4 2013   4 291
#> 5 2013   5 718
#> 6 2013   6 1037
```

- A função `read.csv()` é utilizada para carregar dentro do R um arquivo CSV que, nesse caso, está disponível no repositório github do curso. O arquivo é lido diretamente da internet e é atribuído ao objeto `srag_pr_mes`, que agora armazena os dados do arquivo. O arquivo contém dados sobre Síndrome Respiratória Aguda Grave (SRAG) por mês no Paraná de 2013 a 2017.
- Já a função `head()` exibe as primeiras seis linhas do conjunto de dados armazenado em `srag_pr_mes`. Ele é útil para uma visualização inicial dos dados, permitindo verificar rapidamente a estrutura e as primeiras observações sem exibir o conjunto completo.

Os dados estão num formato tradicional de `data.frame`:

```
# Verificando a classe do objeto
class(srág_pr_mes)
```

```
#> [1] "data.frame"
```

- O comando `class()` retorna o tipo do objeto, ou seja, ele exibe a classe ou as classes do objeto, no caso `srág_pr_mes`.
- `data.frame` é uma estrutura de dados amplamente usada em R que organiza os dados em formato tabular, similar a uma planilha, com linhas e colunas.

No R, o formato mais comum para dados de séries temporais é o formato `time series - ts`. Portanto, vamos converter os dados. Utilizaremos a função `ts()`, que transforma os objetos em série temporal. Para esta função, devemos especificar os seguintes argumentos:

- `data`: Os dados (nesse caso, o número de casos);
- `start`: O início da série;
- `frequency`: Sua frequência (diária = 365, mensal = 12, etc.).

```
# Convertendo os dados para um objeto de classe ts
srág_pr_ts ← ts(
  data = srág_pr_mes$n, # variavel que indica o número de casos
  start = c(2013, 1), # início da série - Janeiro de 2013
  frequency = 12 # frequência da série: mensal
)

class(srág_pr_ts)
```

```
#> [1] "ts"
```

- `ts()`: Esta função transforma um objeto para a classe `ts` (time series, série temporal). Neste caso, o objeto `ts` criado com base nos dados `srag_pr_mes` está sendo armazenado no objeto `srag_pr_ts`.
 - `srag_pr_mes$n`: Esta é a coluna que contém o número de casos de SRAG em cada mês, que estamos usando como os dados principais da série.
 - `start = c(2013, 1)`: Define o ponto inicial da série temporal, neste caso, janeiro de 2013, ou seja, primeiro mês de 2013. O formato `c(ano, mês)` indica o início da série.
 - `frequency = 12`: Define a frequência da série como 12 observações por ano, ou seja, uma série mensal.

Pronto! Agora temos um objeto de série temporal adequado para nossas primeiras análises no R. Antes de começar, vamos ver como é a estrutura do objeto `ts`?

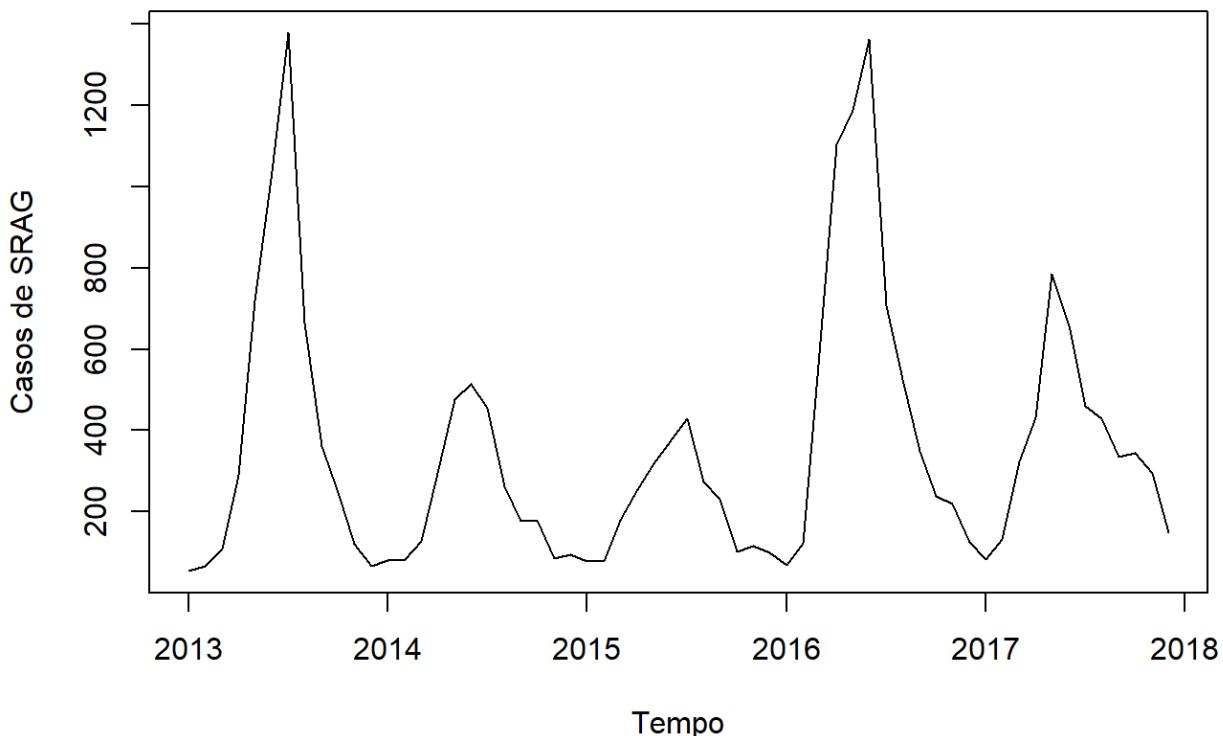
```
# Vendo os dados
srag_pr_ts
```

```
#>      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
#> 2013   54  66 108 291 718 1037 1378 664 362 248 120  67
#> 2014   80  83 128 296 477 514 455 263 178 180  86  94
#> 2015   77  77 180 255 319 373 428 274 231 101 115  99
#> 2016   69 123 590 1104 1189 1362 708 525 350 238 219 126
#> 2017   83 132 324 432 785 657 459 430 334 344 296 148
```

Percebam que o objeto `ts srag_pr_ts` tem os números de casos de SRAG por ano nas linhas e por meses nas colunas.

Agora, vamos fazer um gráfico da série temporal `srag_pr_ts` utilizando a função `plot()`. Este gráfico mostra a evolução do número de casos de SRAG ao longo do tempo, facilitando a identificação de tendências, sazonalidade ou outros padrões nos dados.

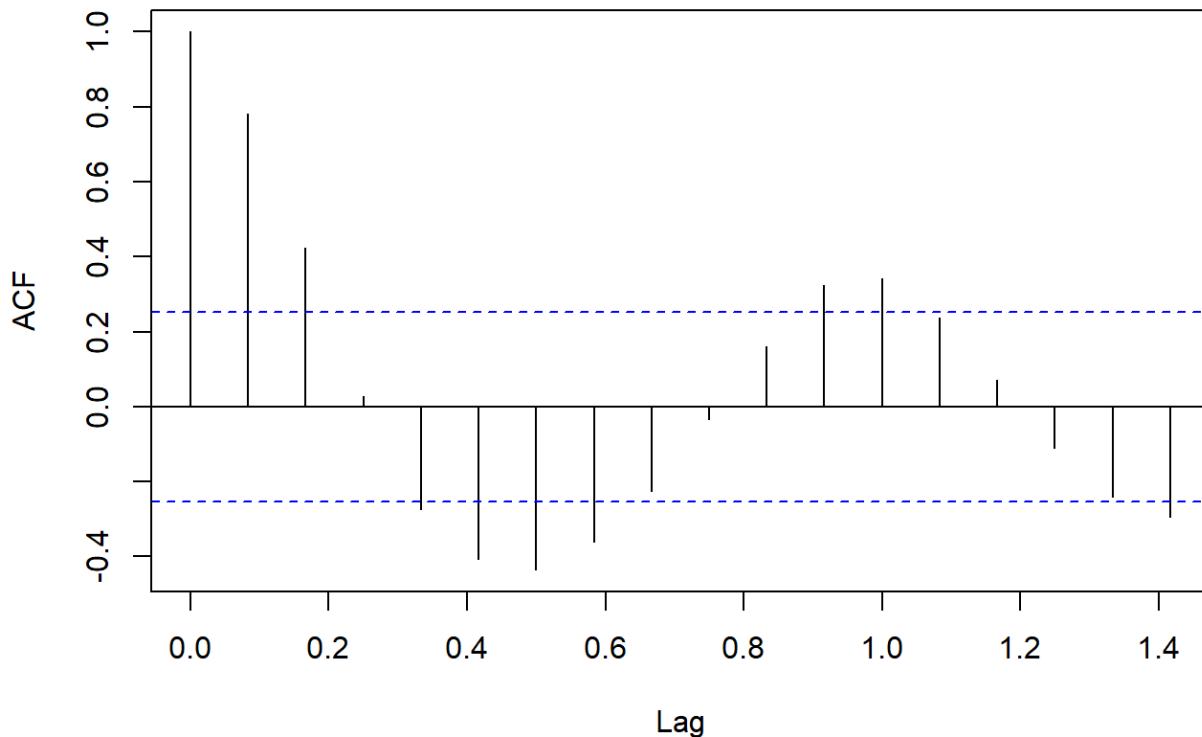
```
# Fazendo o gráfico da série temporal
plot(srag_pr_ts,
      ylab = 'Casos de SRAG', # nomeando o eixo y
      xlab = 'Tempo') # nomeando o eixo x
```



Partindo para as análises, vamos primeiro testar a autocorrelação da série por meio da função de autocorrelação - `acf()`:

```
acf(srag_pr_ts)
```

Series srag_pr_ts



- O comando `acf()` calcula e plota a função de autocorrelação (ACF) para um objeto `ts`, neste caso, a série temporal `srag_pr_ts`. Esse gráfico é utilizado para verificar a correlação entre os valores da série ao longo de diferentes defasagens (lags), indicando o quanto os valores da série em diferentes períodos estão correlacionados com valores anteriores.

Outra abordagem para identificação de autocorrelação é por meio do teste estatístico de Ljung-Box. Nele, tem-se que:

$$H_0: \rho_h = 0$$

$$H_1: \rho_h \neq 0$$

sendo ρ_h a correlação da série com ela mesma (autocorrelação) defasada em h lags. No R, podemos realizar o teste de Ljung-Box com a função `Box.test()`.

```
Box.test(srág_pr_ts, lag = 20, type = "Ljung-Box")
```

```
#>
#> Box-Ljung test
#>
#> data: srág_pr_ts
#> X-squared = 146.35, df = 20, p-value < 2.2e-16
```

O comando `Box.test()` realiza o teste de Ljung-Box para avaliar a independência de uma série temporal armazenada como objeto `ts`, no caso, de `srág_pr_ts`.

- `lag = 20`: especifica que o teste deve incluir até 20 defasagens. Em outras palavras, ele analisa a presença de autocorrelação nas primeiras 20 defasagens da série.
- `type = "Ljung-Box"`: define o tipo do teste como "Ljung-Box", teste comum para verificar autocorrelação em séries temporais.

Vê-se, tanto por meio do gráfico (onde vemos altas correlações nos primeiros lags) quanto por meio do teste (em que rejeitamos a hipótese nula de independência), que há evidências de dependência temporal na série de SRAG mensal no Paraná.

Breve análise descritiva da série

O comando `summary()` fornece medidas resumo sobre a série temporal. Com ele obtemos os valores mínimo, máximo, mediana e intervalos interquartílicos.

```
summary(srág_pr_ts)
```

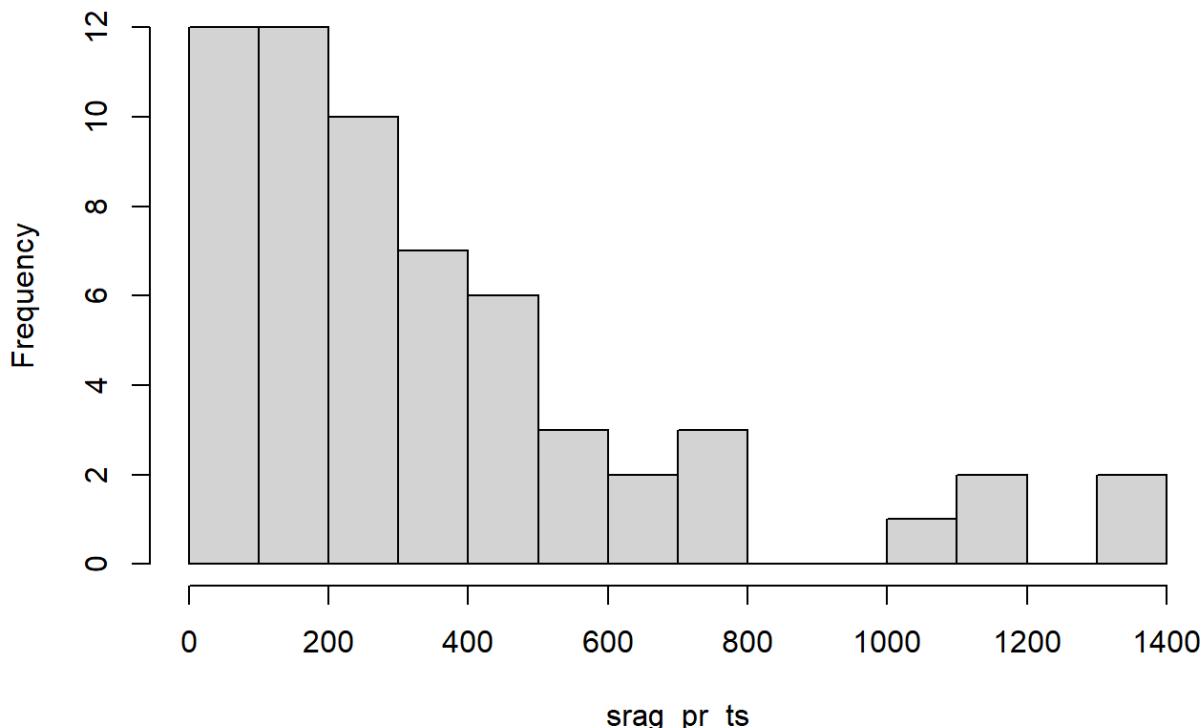
```
#>      Min. 1st Qu. Median      Mean 3rd Qu.      Max.  
#>    54.0   118.8   268.5   358.4   456.0  1378.0
```

O comando `summary()` exibe um resumo estatístico básico, neste caso da série temporal `srág_pr_ts`.

Vemos que a mediana de casos de SRAG por mês é 268.5, variando de 54 até 1378 casos por mês no período. Para entender como o número mensal de casos se distribui podemos fazer um **histograma**:

```
hist(srág_pr_ts, breaks = 10)
```

Histogram of srág_pr_ts



O comando `hist()` gera um histograma, neste caso, da série temporal `srág_pr_ts`.

- `breaks = 10`: Especifica que o histograma terá 10 intervalos. O número de intervalos afeta a granularidade do gráfico, onde um número maior de intervalos fornece mais detalhes, enquanto um número menor de intervalos dá uma visão mais generalizada da distribuição.

Vemos que a maior concentração se dá até 200 casos por mês, mas alguns meses têm mais de 1000 casos.



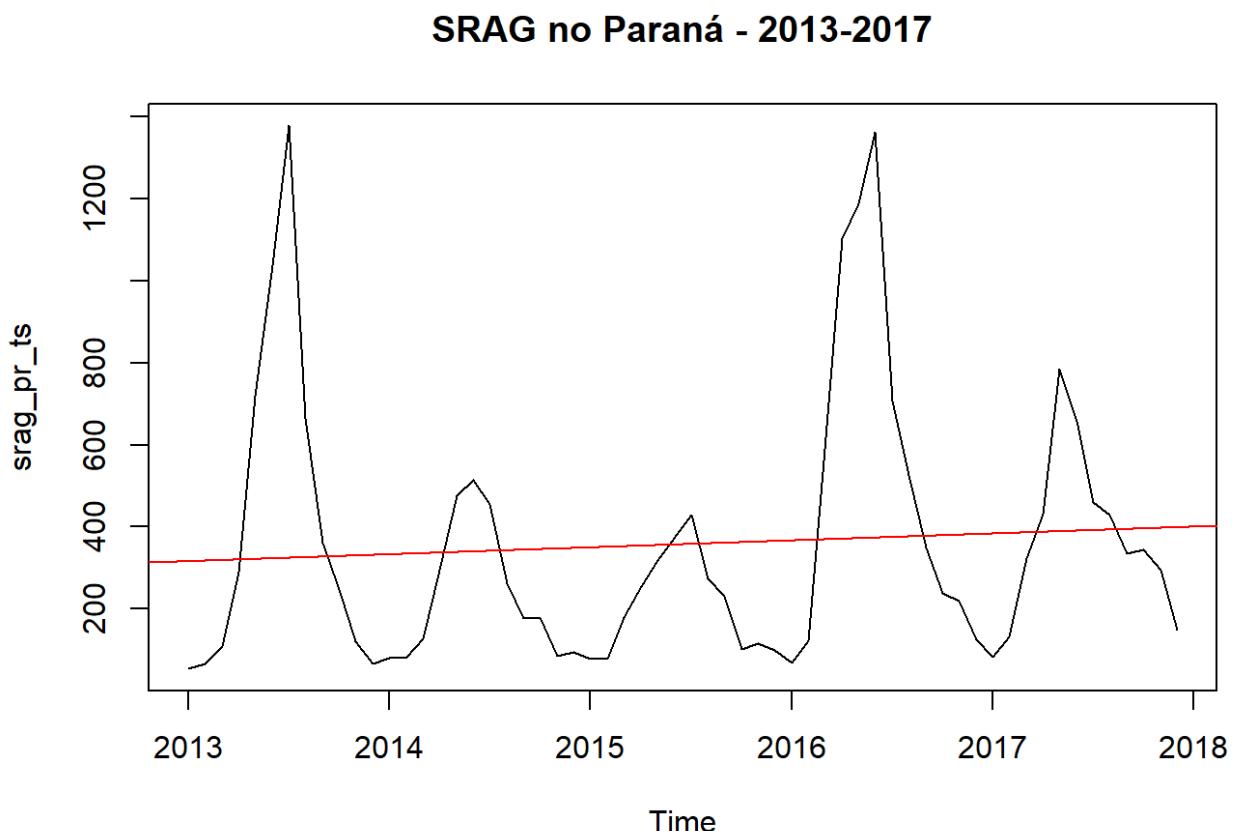
Analizando seus componentes

Vimos que as séries são compostas de tendência, sazonalidade, ciclo e ruído aleatório. Vamos tentar identificar graficamente cada um desses componentes.

Tendência

Um modelo de regressão linear simples pode ser um indicador para uma possível **tendência linear** na série. Vamos verificar se é o caso:

```
plot(srág_pr_ts, main = "SRAG no Paraná - 2013-2017")
abline(reg = lm(formula = srág_pr_ts ~ time(srág_pr_ts)),
      col = "red")
```



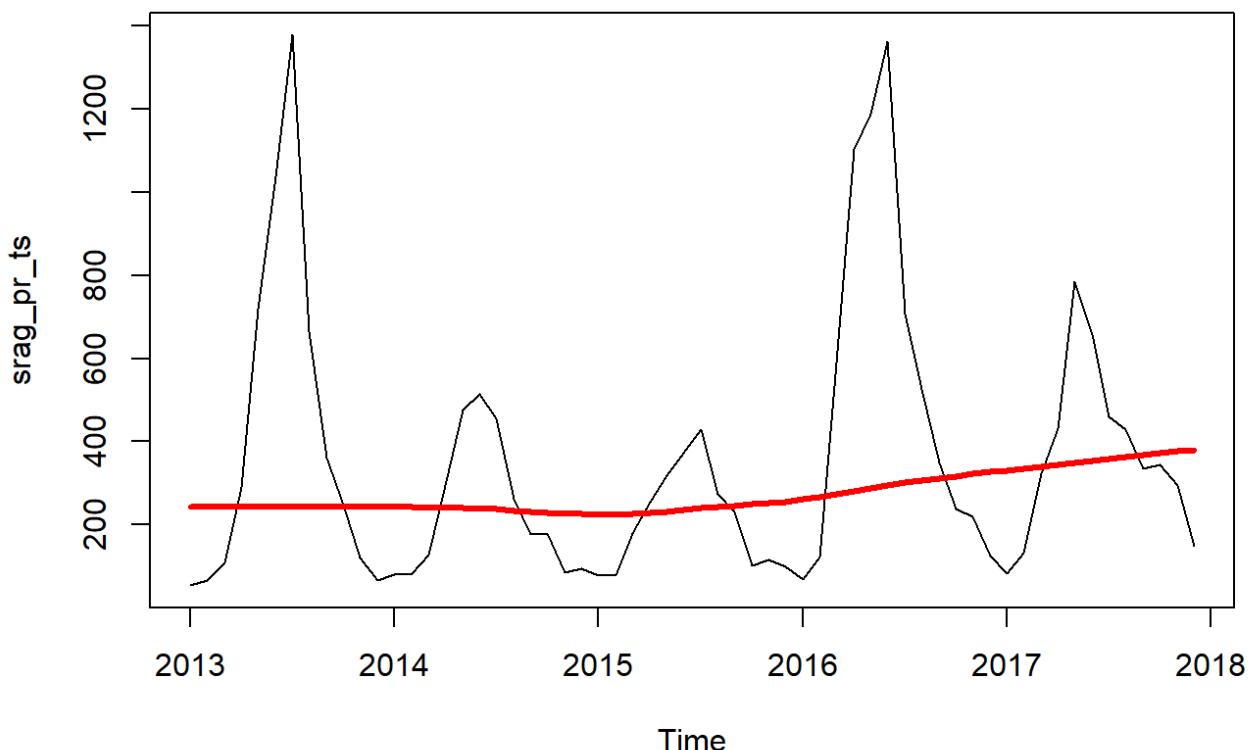
1. O comando `plot()` está sendo utilizado para plotar a série temporal `srag_pr_ts`.
 - O argumento `main` define o título do gráfico como “SRAG no Paraná - 2013-2017”.
2. O comando `abline()` adiciona uma linha ao gráfico criado por `plot()` no comando anterior. Neste caso, `abline()` está adicionando uma linha de regressão linear obtida pela função `lm()`:
 - `lm(formula = srag_pr_ts ~ time(srag_pr_ts))` gera um modelo de regressão linear que usa o tempo `time(srag_pr_ts)` para prever os valores da série `srag_pr_ts`.
 - `col = "red"` define a cor da linha da regressão linear como vermelha, destacando-a no gráfico.

Vê-se que a linha tem uma leve e quase imperceptível inclinação, indicando que a série não possui uma tendência muito forte. Em caso de tendências não lineares ao longo do tempo, a função `lowess()` também pode ser uma boa indicadora. Vamos verificar.

```
# Vamos utilizar um pacote chamado Kendall para calcular a tendência
library(Kendall)

# Visualizando a tendência
plot(srag_pr_ts, main = "SRAG no Paraná - 2013-2017")
lines(lowess(time(srag_pr_ts), srag_pr_ts),
      lwd = 3,
      col = "red")
```

SRAG no Paraná - 2013-2017



O pacote `Kendall` é usado para exibir a tendência na série temporal `srág_pr_ts`. Caso precise, instale-o através do comando `install.packages("Kendall")`.

- `plot(srág_pr_ts, main = "SRAG no Paraná - 2013-2017")`: Plota a série temporal com o título “SRAG no Paraná - 2013-2017”.
- `lines(lowess(time(srág_pr_ts), srág_pr_ts), lwd = 3, col = 2)`: Adiciona uma linha de suavização (`lowess`) sobre os dados para visualizar a tendência, com a linha de espessura 3 (`lwd = 3`) e na cor vermelha (`col = "red"`).

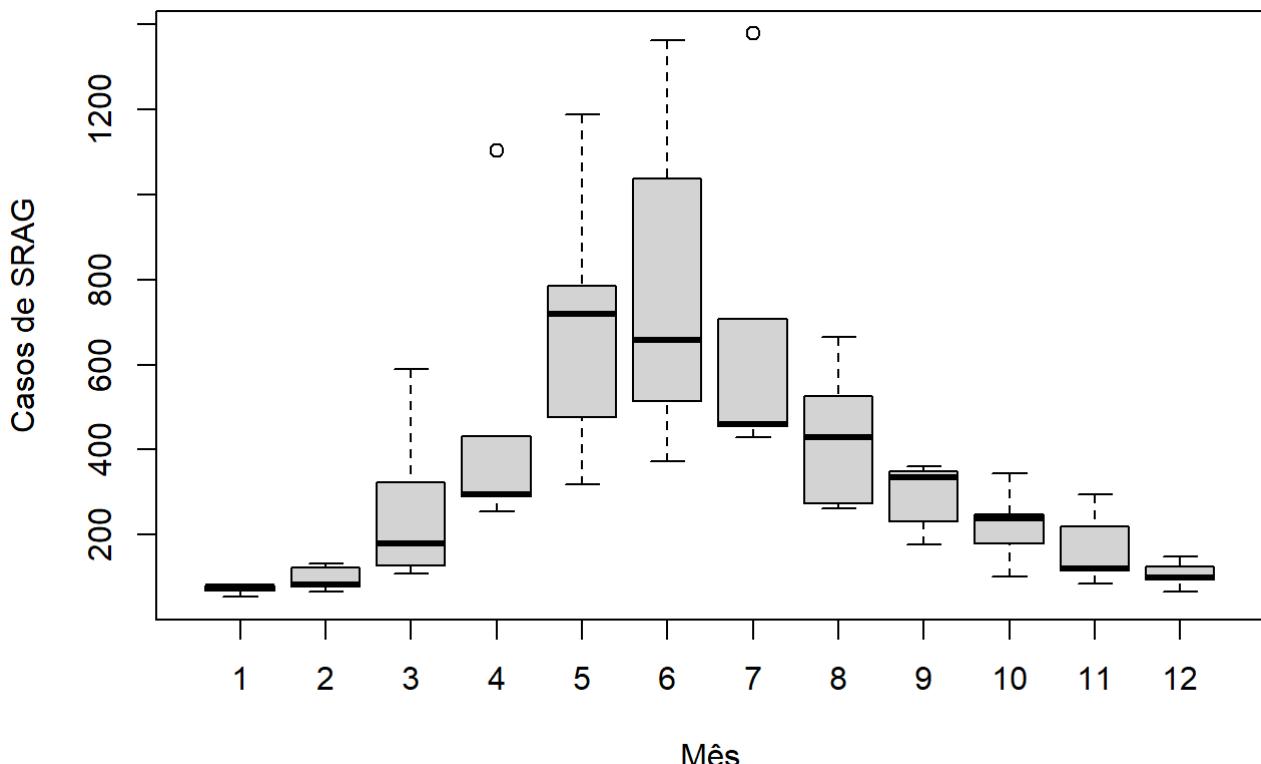
A linha `lowess` facilita a observação da tendência subjacente, ajudando a destacar variações de longo prazo na série temporal.

Vemos que agora a linha de tendência se adapta conforme o tempo, indicando maior inclinação nos anos recentes da série (2016 e 2017). Contudo, ainda vemos uma tendência fraca.

Sazonalidade

Uma maneira simples de verificar a existência de sazonalidade é visualizar como a distribuição de casos se dá em cada mês. Dessa forma, vemos meses em que o número de casos tende a ser maior ou menor. Vamos, então, criar um boxplot para verificar a sazonalidade.

```
# Criando um boxplot para verificar a sazonalidade
boxplot(formula = srag_pr_ts ~ cycle(srag_pr_ts),
        xlab = "Mês",
        ylab = "Casos de SRAG")
```

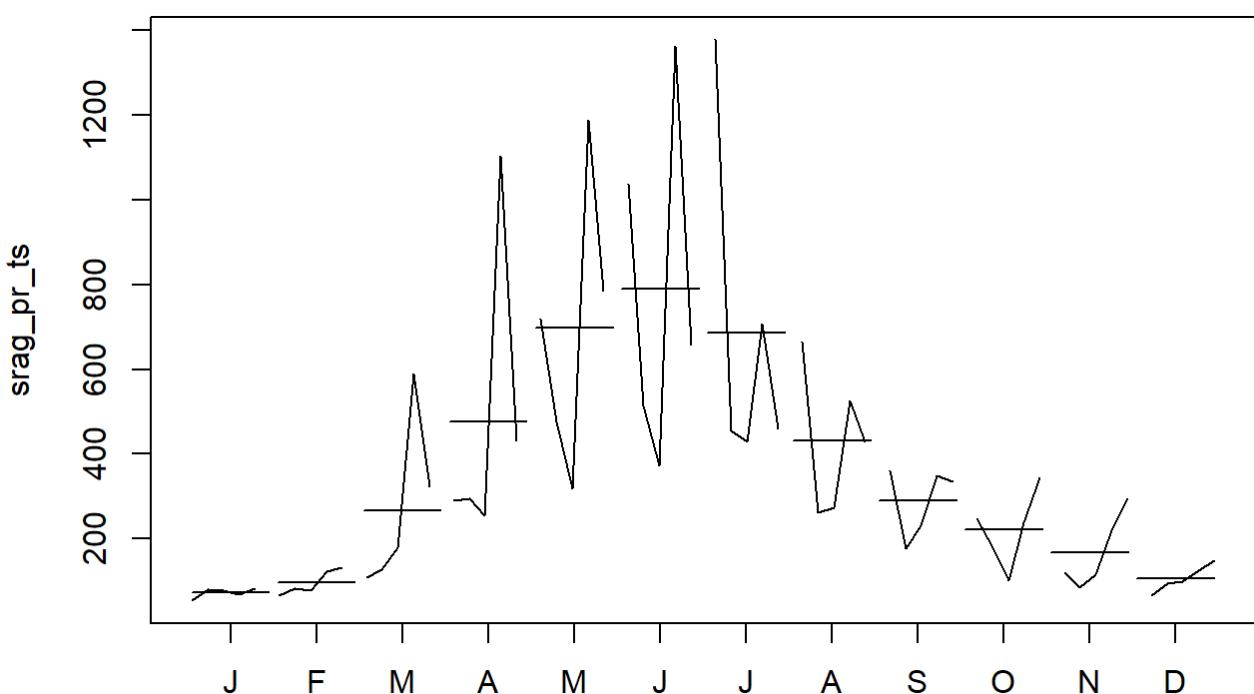


A função `boxplot()` neste caso gera boxplots para verificar a sazonalidade na série temporal `srag_pr_ts`, mostrando a variação dos casos de SRAG ao longo dos meses.

- `srag_pr_ts ~ cycle(srag_pr_ts)`: Agrupa os dados da série temporal `srag_pr_ts` por ciclo mensal. A função `cycle(srag_pr_ts)` extrai o mês de cada observação, possibilitando uma comparação mês a mês.
- `xlab = "Mês" e ylab = "Casos de SRAG"`: Colocar o rótulo nos eixos X e Y como "Mês" e "Casos de SRAG", respectivamente.

Por meio do boxplot obtido, é nítida a maior distribuição de casos no final do outono e inverno, como o período de maior a agosto. Também podemos obter uma visualização com o `monthplot()`:

```
# Utilizando a função monthplot() para visualizar a sazonalidade
monthplot(srag_pr_ts)
```



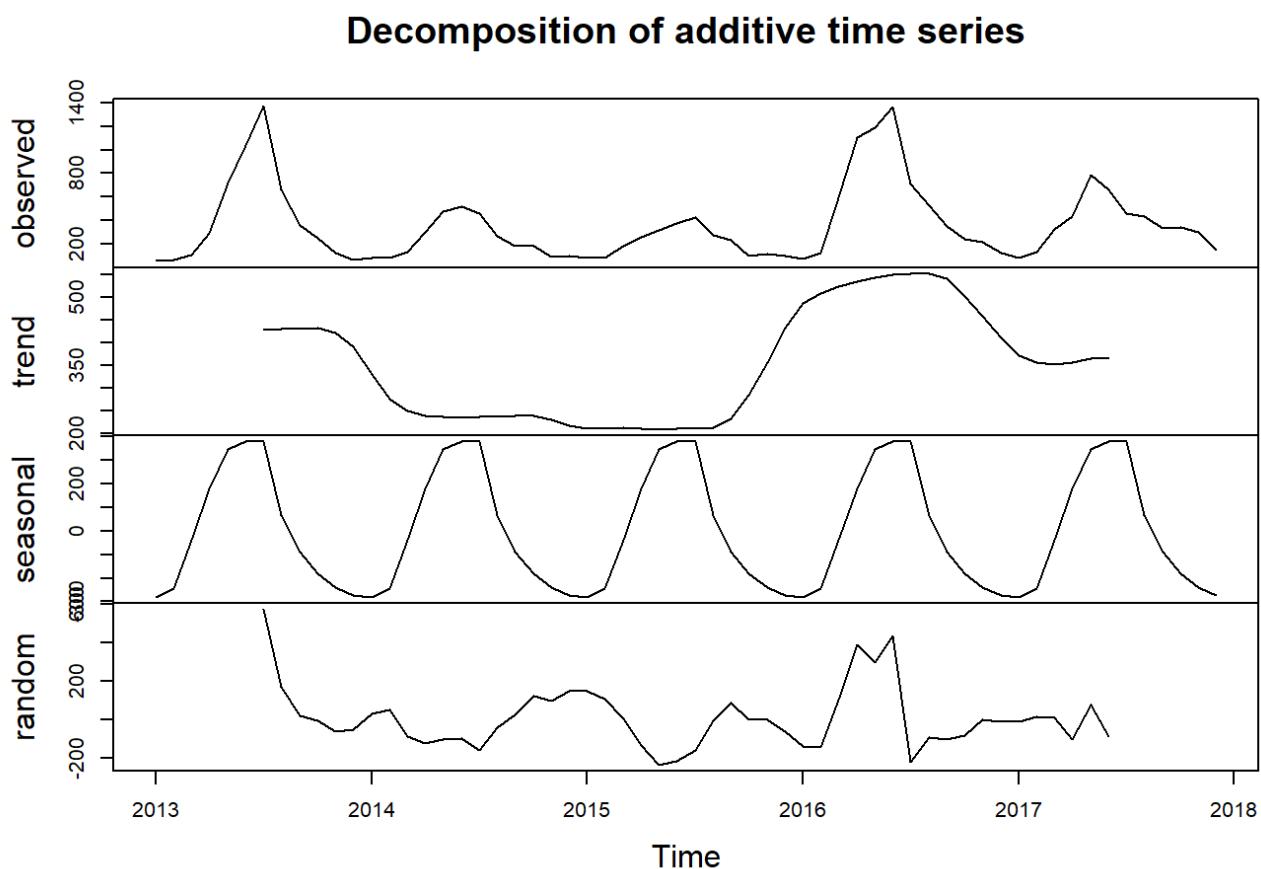
- `monthplot(srág_pr_ts)` gera um gráfico que exibe a média dos valores de `srág_pr_ts` para cada mês, destacando padrões sazonais ao longo dos anos.

Logo, percebemos que a sazonalidade se manifesta mais fortemente na série de SRAG do que a tendência. Podemos verificar isto por meio da função de decomposição da série.

Decomposição da série

Há um método clássico no R para decomposição da série chamado de `decompose()`, que realiza a decomposição da série em seus componentes utilizando **médias móveis**. Veja:

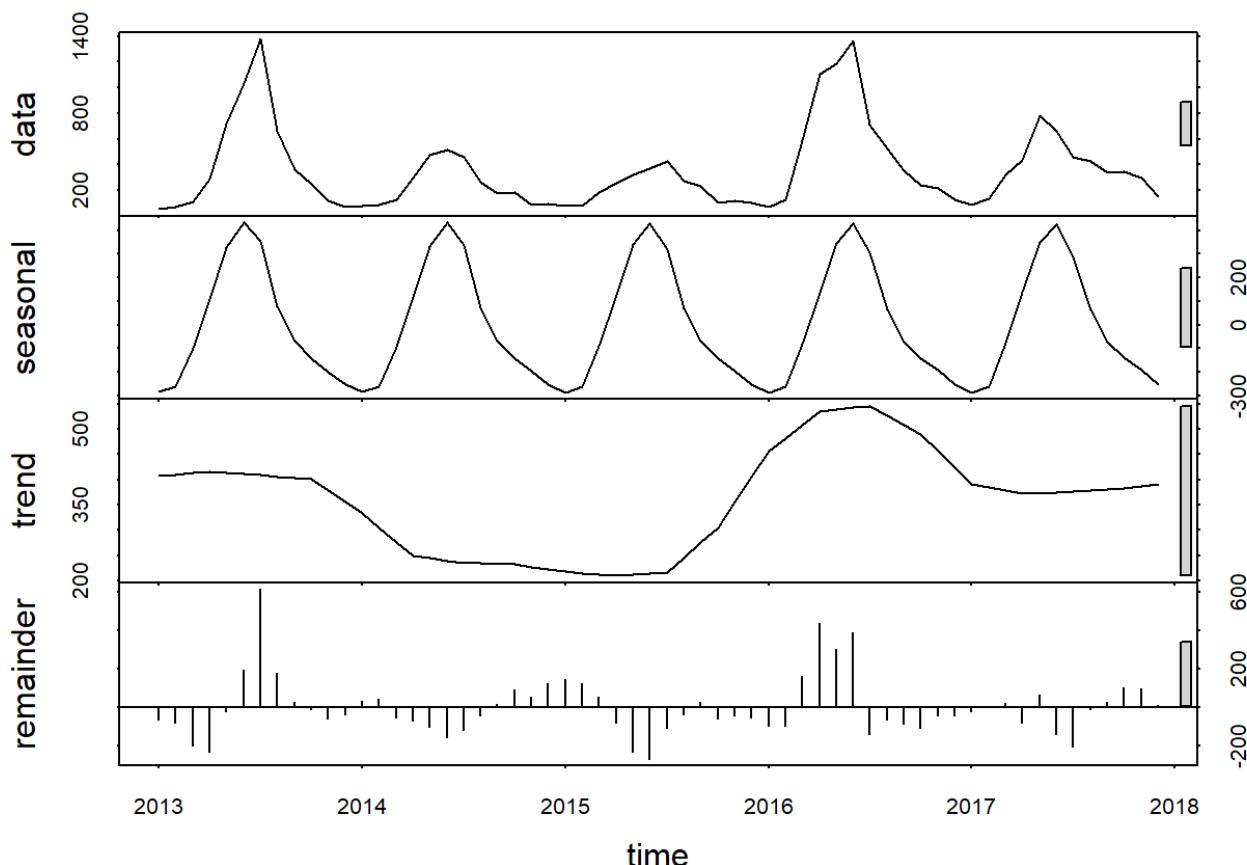
```
# Decomposição da série usando a função decompose()
plot(decompose(srág_pr_ts))
```



- A função `decompose()` decompõe uma série temporal, no caso `srag_pr_ts`, em suas componentes (tendência, sazonalidade e resíduos).
- O comando `plot()` plota cada um desses componentes separadamente para facilitar a análise visual.

Uma das características da aplicação do método de médias móveis é justamente a perda dos valores iniciais e finais da série (veremos mais a frente o motivo). Há outra abordagem de decomposição, chamada de **STL - Seasonal and Trend Decomposition Using Loess**. Como o nome diz, a decomposição é feita por meio do método Loess (que também veremos mais adiante), e realiza uma decomposição com mais robustez, mais sensível a outros tipos de sazonalidade e que lida melhor com *outliers*.

```
# Decomposição da série usando a função stl()
plot(stl(srag_pr_ts, s.window = 12))
```



- A função `stl()` faz uma decomposição STL (Seasonal and Trend decomposition using Loess) de uma série temporal, no caso `srag_pr_ts`, separando suas componentes (tendência, sazonalidade e resíduos). `s.window = 12` define uma janela de suavização sazonal de 12 períodos.
- `plot()`: Plota o resultado da decomposição, exibindo gráficos separados para cada componente.

Como visto anteriormente, vê-se a série original (*data*, no topo), seu componente sazonal (*seasonal*, segundo gráfico), tendência (*trend*, terceiro gráfico) e resíduos (*remainder*, último gráfico). Ao lado direito, a barra de escala indica a importância para composição da série de cada termo (quanto menor, mais relevante). Vê-se que a sazonalidade e o componente aleatório são, nesse caso, os componentes mais presentes na série. Ao olhar o componente aleatório, observa-se os maiores picos em 2013 e 2016, onde o número de casos atingiu patamares maiores do que o comum.

Podemos salvar o objeto decomposto:

```
# Salvando o objeto decomposto e visualizando seus itens
srag_pr_decomp ← stl(srag_pr_ts, s.window = 12)

head(srag_pr_decomp$time.series)
```

```
#>      seasonal    trend remainder
#> [1,] -285.0417 407.0146 -67.97293
#> [2,] -261.9554 409.8916 -81.93621
#> [3,] -100.8078 412.7687 -203.96086
#> [4,]  109.6366 415.6457 -234.28225
#> [5,]  328.4953 413.3284 -23.82369
#> [6,]  433.6265 411.0110 192.36244
```

Esse código utiliza a decomposição STL para separar a série temporal `srag_pr_ts` em suas componentes.

- `srag_pr_decomp ← stl(srag_pr_ts, s.window = 12)`: Aplica a decomposição STL (Seasonal-Trend decomposition using Loess) à série `srag_pr_ts`, com uma janela sazonal com 12 períodos, e armazena no objeto `srag_pr_decomp`.
- `head(srag_pr_decomp$time.series)`: Exibe as primeiras linhas da decomposição, com três componentes: seasonal (sazonalidade), trend (tendência) e remainder (resíduos), para rápida visualização dos dados de cada parte.

Vemos que a sazonalidade é a primeira coluna, tendência a segunda e o componente aleatório a terceira. Vamos extraí-las para objetos próprios:

```
srag_pr_tendencia ← srag_pr_decomp$time.series[,2]
srag_pr_sazonal ← srag_pr_decomp$time.series[,1]
srag_pr_aleatorio ← srag_pr_decomp$time.series[,3]
```

Esses comandos extraem as componentes de tendência, sazonalidade e aleatoriedade de uma decomposição de série temporal armazenada em `srag_pr_decomp`.

- `srag_pr_tendencia ← srag_pr_decomp$time.series[,2]`: Armazena a componente de tendência da decomposição no objeto `srag_pr_tendencia`.
- `srag_pr_sazonal ← srag_pr_decomp$time.series[,1]`: Armazena a componente sazonal (variação periódica) no objeto `srag_pr_sazonal`.
- `srag_pr_aleatorio ← srag_pr_decomp$time.series[,3]`: Armazena a componente aleatória (ruído) no objeto `srag_pr_aleatorio`.

Essas variáveis permitem analisar cada componente separadamente para entender como a tendência, sazonalidade e ruído contribuem para o comportamento da série.

```
par(mfrow = c(3, 1))
plot(srag_pr_tendencia, main = "Componente de Tendência")
plot(srag_pr_sazonal, main = "Componente Sazonal")
plot(srag_pr_aleatorio, main = "Componente Aleatório")
```

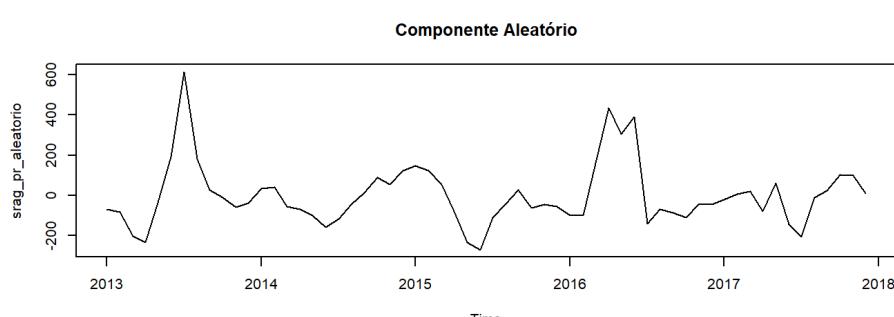
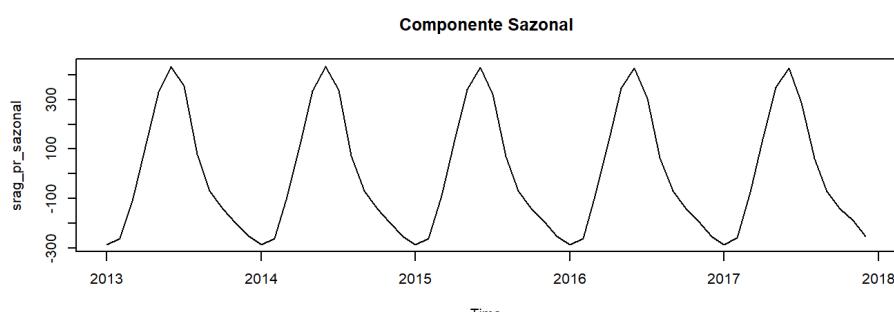
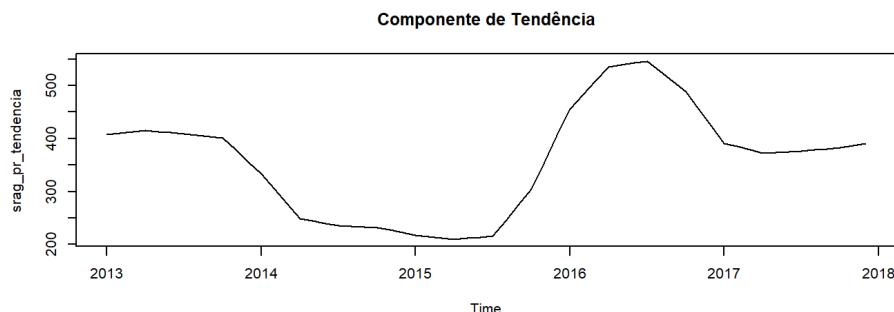


Esse comando cria uma visualização conjunta dos três componentes de uma decomposição da série temporal, cada um em seu próprio gráfico:

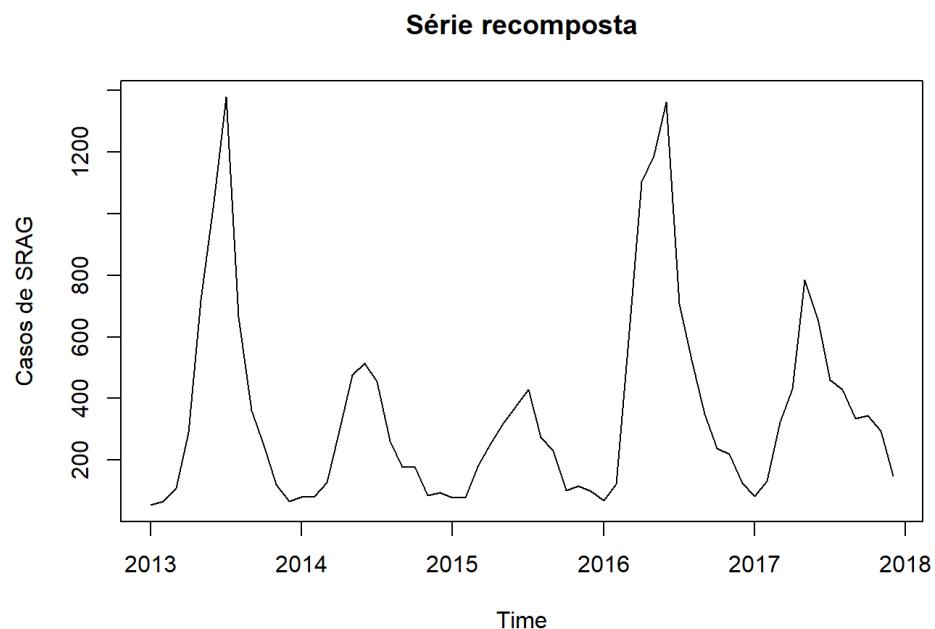
- `par(mfrow = c(3, 1))`: Divide a área de plotagem em 3 linhas e 1 coluna, exibindo três gráficos verticalmente.
- `plot(srág_pr_tendencia, main = "Componente de Tendência")`: Plota o componente de tendência da série.
- `plot(srág_pr_sazonal, main = "Componente Sazonal")`: Plota o componente sazonal.
- `plot(srág_pr_aleatorio, main = "Componente Aleatório")`: Plota o componente aleatório (residual).

Esse layout facilita a análise visual de cada componente individual da série temporal.

E como a série é formada pela soma dos componentes, podemos obter a série original adicionando-os:



```
plot(
  srag_pr_tendencia + srag_pr_sazonal + srag_pr_aleatorio,
  main = "Série recomposta",
  ylab = "Casos de SRAG"
)
```



Esse comando cria um gráfico da série recomposta somando os objetos com as componentes: `srag_pr_tendencia`, `srag_pr_sazonal` e `srag_pr_aleatorio` de `srag_pr_ts`.

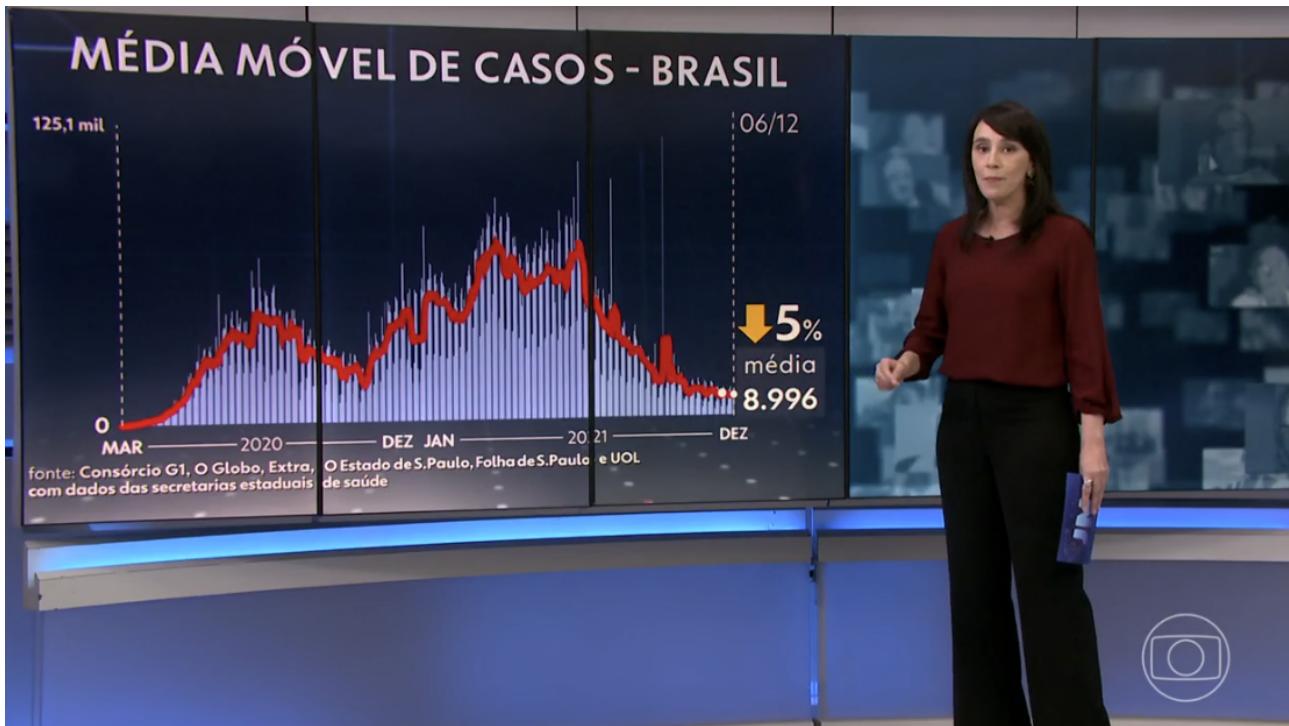
- `srag_pr_tendencia + srag_pr_sazonal + srag_pr_aleatorio`: Soma as componentes de tendência, sazonalidade e aleatoriedade da série para reconstruir a série original.
- `main = "Série recomposta"`: Define o título do gráfico como "Série recomposta".
- `ylab = "Casos de SRAG"`: Define o rótulo do eixo Y como "Casos de SRAG".

Chegamos ao final da nossa primeira prática com R. Agora, vamos seguir com o curso, dedicando nosso tempo a entender as técnicas de transformações e suavizações aplicadas às séries temporais, especialmente quando os dados estão muito "ruidosos". Preparados? Vamos lá!

Módulo 4 - Transformações e suavizações

Ao analisar séries temporais, nem sempre os padrões que queremos investigar ficam claros à primeira vista. Isso ocorre, muitas vezes, porque a série contém ruído – flutuações aleatórias que podem dificultar a identificação de tendências ou padrões sazonais. Um exemplo comum é quando os dados diários possuem muita variação, tornando difícil distinguir uma verdadeira mudança de uma simples variação aleatória.

Um exemplo frequente disso foi durante a pandemia de covid-19, quando a imprensa e as agências de saúde utilizaram a técnica da média móvel para suavizar os valores diários de casos confirmados e mortes. Essa suavização ajudou a acompanhar a evolução da pandemia de forma mais clara, destacando a tendência geral dos casos e permitindo uma visualização mais estável dos dados.



Média móvel de casos da covid-19 durante a pandemia. Fonte: <https://g1.globo.com/jornal-nacional/noticia/2021/12/06/media-diaria-de-mortes-por-covid-no-brasil-fica-abixo-de-200-pelo-terceiro-dia-seguido.ghtml>

No gráfico acima, os dados diários (em cinza) mostram muitas flutuações, o que dificulta perceber imediatamente uma tendência clara. Quando olhamos apenas para um dia específico, como um valor baixo, não podemos saber se isso indica realmente uma queda nos casos ou apenas uma flutuação aleatória. Ao utilizar a **média móvel** (linha vermelha), conseguimos visualizar essas variações e obter uma visão mais clara da tendência geral, facilitando a interpretação dos dados.

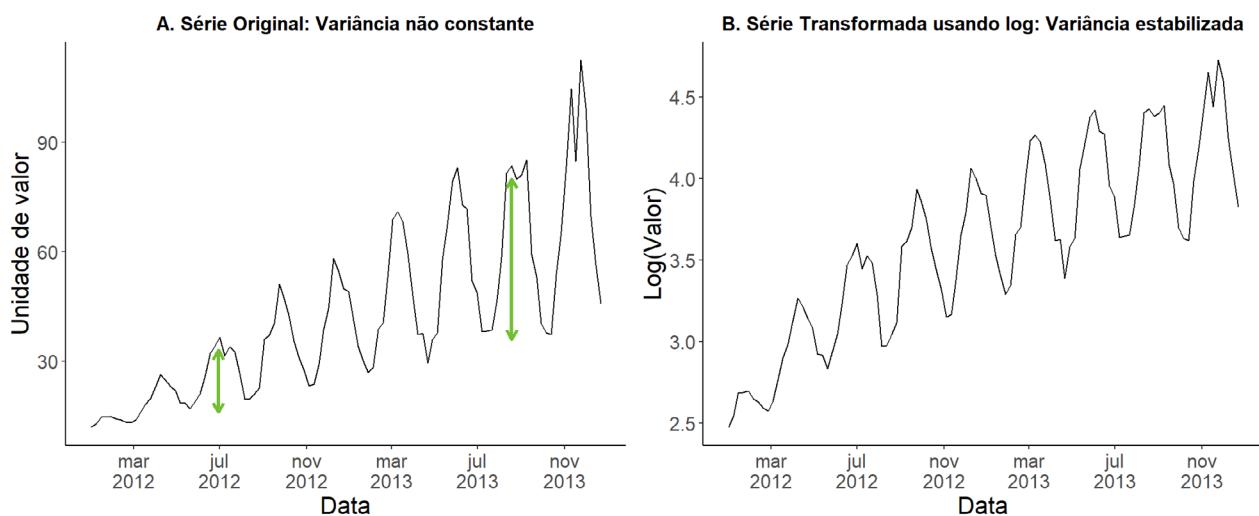
Nas próximas seções, vamos explorar técnicas de transformações e suavizações que podem ser aplicadas para lidar com ruídos e variações indesejadas, permitindo uma análise mais precisa e confiável das séries temporais.

Funções

A aplicação de certas funções, como a função **logarítmica** e a transformação de **Box-Cox** são úteis para **estabilização da variância** dos dados. Por exemplo, se os casos de uma doença crescem exponencialmente, a transformação logarítmica pode “linearizar” esse crescimento, facilitando a análise. Poderemos usar isso nas técnicas de modelagem de séries temporais, mas veremos só mais à frente. Não se preocupe.

Na Figura 22A vemos uma série cuja variância não é constante, ou seja, a amplitude dos valores ao redor da tendência (média) varia ao longo do tempo. Já na Figura 22B vemos a série após a aplicação da transformação logarítmica.

Figura 22: Exemplo de transformações de séries temporais.





Observação: Por padrão, quando utilizamos a função `log()` no software R e, por consequência, aplicamos a transformação logarítmica, estamos nos referindo ao log na **base natural (e)**. Em outras notações pode ser conhecido como “log natural”, ou \ln .

Box-Cox

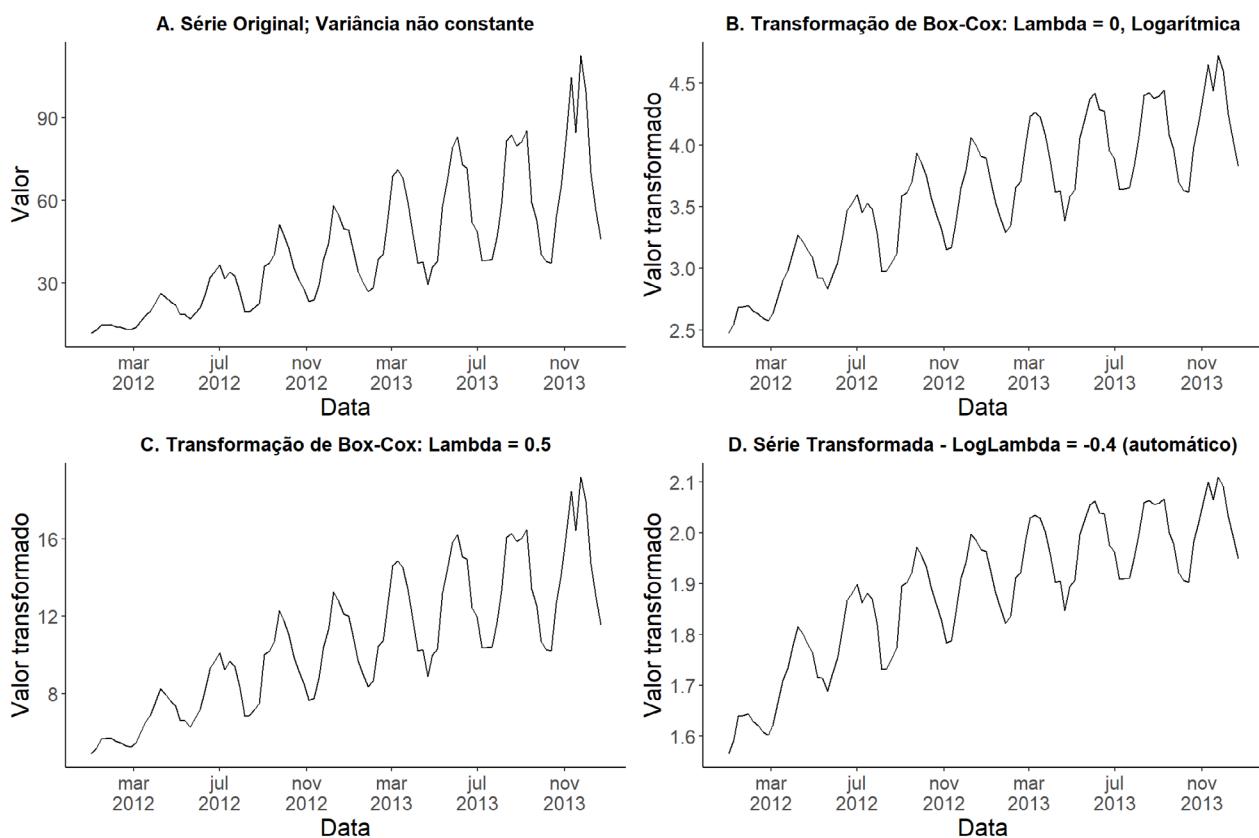
Box e Cox (1964) propuseram uma transformação mais versátil, chamada transformação de Box-Cox. Essa depende de um parâmetro lambda (λ), que define como será a transformação. Caso λ seja zero, a transformação logarítmica que já conhecemos é aplicada. Caso contrário, aplica-se a transformação exponencial da série à potência λ , ou seja:

$$Z_t(\lambda) = \begin{cases} Z_t^\lambda, & \text{se } \lambda \neq 0, \\ \ln(Z_t), & \text{se } \lambda = 0. \end{cases}$$

A transformação de Box-Cox também é útil para estabilização da variância e pode funcionar melhor em mais situações, além de aproximar a distribuição dos dados à distribuição normal para melhorar o ajuste de modelos de séries temporais, como o modelo ARIMA.

Podemos testar, portanto, vários valores de λ . Acompanhe na Figura 23 um exemplo de uso de vários λ diferentes:

Figura 23: Exemplo de uso de λ para transformar séries temporais.



É possível trabalhar com um valor de λ escolhido automaticamente, que nesse caso foi o de $\lambda = -0.4$. O método automático foi proposto por Guerrero (1993) e busca a escolha de um valor de λ que minimiza o coeficiente de variação para a série.



Diferenciação

Uma vez que verificamos que a série é não estacionária (ou seja, possui tendência), podemos trabalhar para remover essa tendência da visualização. Novamente, essa operação pode ser necessária para aplicação de modelos que pressupõem estacionariedade da série, como veremos mais a frente. Uma das formas de atingir esse objetivo é por meio da **diferenciação**.

O processo de diferenciação de primeira ordem é simples e se dá, para cada valor da série, pela subtração do valor anterior. Ou seja, em uma série Z_t , o valor em cada momento da série se dá por:

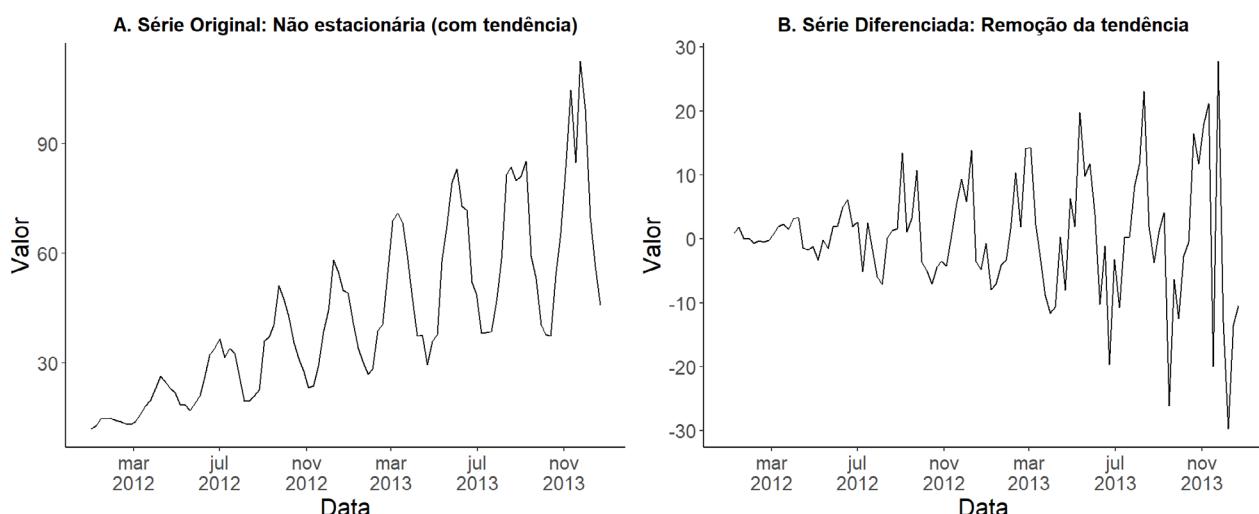
$$Z'_t = Z_t - Z_{t-1}$$

Onde:

- Z'_t representa os valores da série já diferenciada.

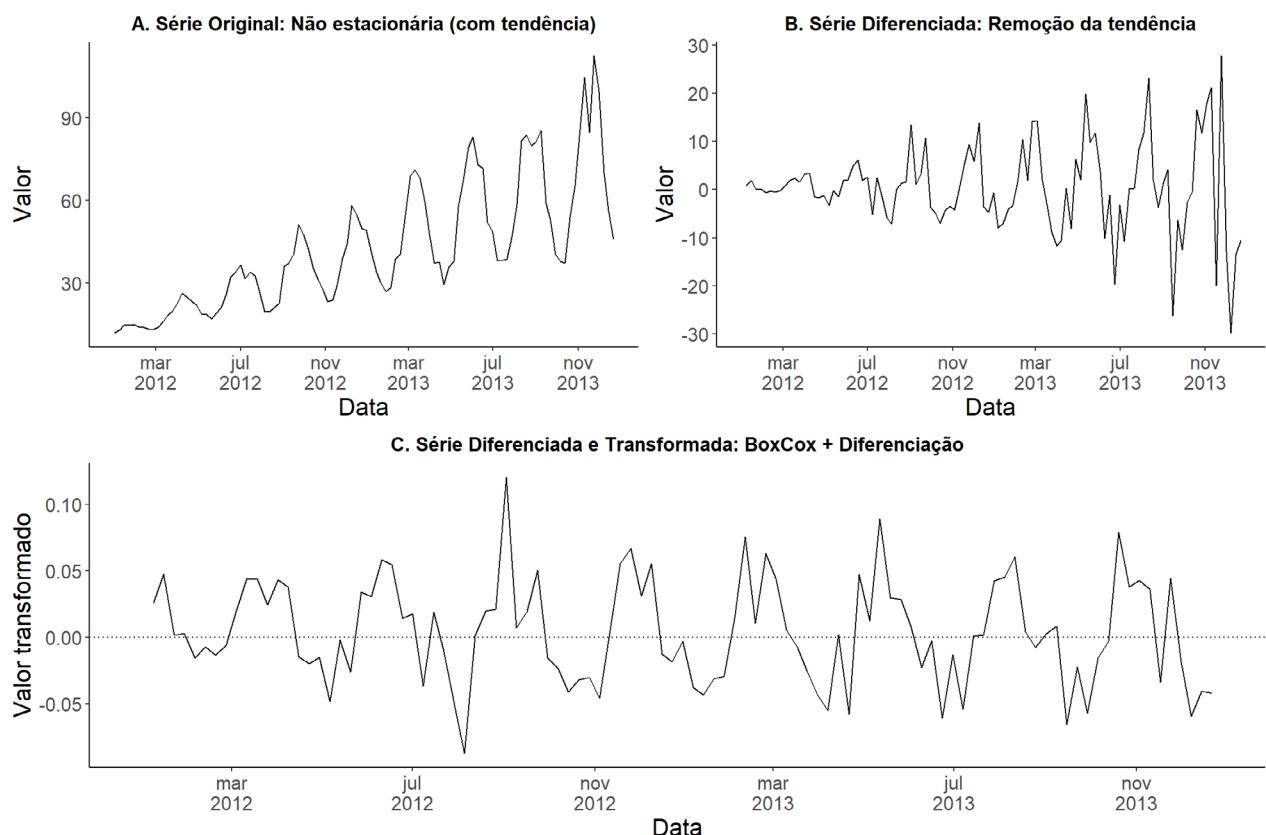
Vamos aplicar a diferenciação sobre a mesma série usada anteriormente.

Figura 24: Exemplo de uso de diferenciação em séries temporais.



Vemos que agora a tendência foi removida, e temos uma série centrada no valor zero. Podemos combinar as duas transformações vistas para obter uma série estacionária e com variância estabilizada:

**Figura 25: Exemplo de série temporal após
remoção da tendência e estabilização da variância.**

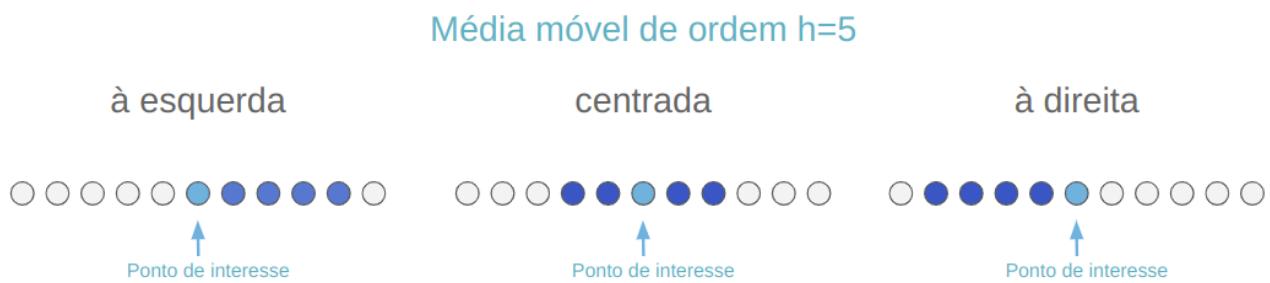




Médias móveis

Assim como vimos no exemplo da covid-19, a média móvel é uma técnica para suavizar a série temporal, destacando sua tendência e tirando a influência do ruído aleatório. A técnica, como o nome diz, consiste em uma média aritmética das observações. Neste caso, são várias médias: para cada ponto na série, calcula-se a média dele mesmo com as h observações mais próximas. Esse valor de h define a ordem da média móvel. Por exemplo, uma média móvel de ordem 3 se refere a média do próprio ponto e seus dois pontos próximos, podendo ser à esquerda, à direita ou em ambas direções.

Figura 26: Média móvel de ordem $h=5$, alinhada à esquerda, à direita e centrada.

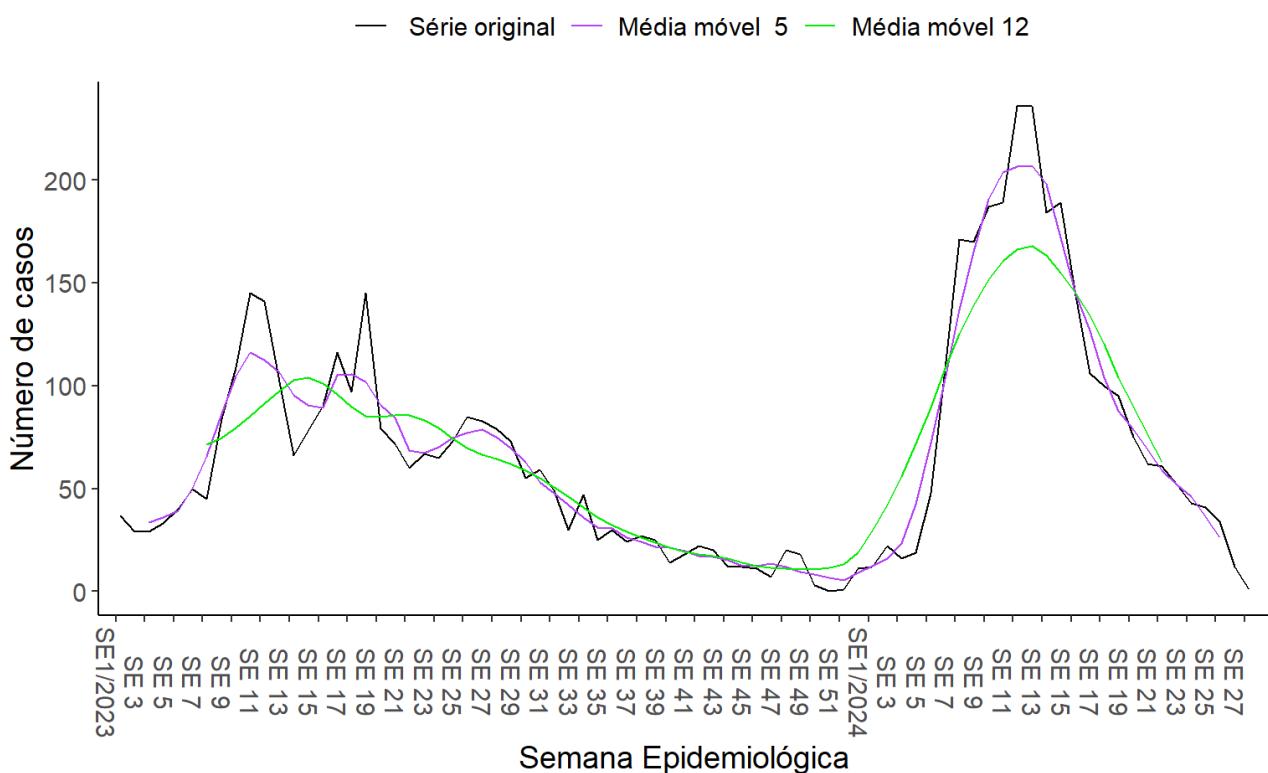


Dessa forma, temos que:

$$M_t = \frac{Z_t + Z_{t-1} + \dots + Z_{t-(h-1)}}{h}$$

Para representar uma média móvel à direita (onde olhamos para os $h - 1$ valores passados, como era o exemplo da covid-19), vamos aplicar uma média móvel centrada com $h - 5$ e $h = 12$ para a série semanal de dengue no Maranhão nos anos de 2023 e 2024 (Figura 27).

Figura 27: Casos prováveis de dengue, São Luís-MA, 2023 e 2024.



Ao analisarmos a Figura 27 podemos perceber que, à medida que aumentamos o valor de h , a série temporal se torna mais suavizada. No entanto, também observamos que as séries suavizadas acabam perdendo parte de sua extensão. Mas por que isso acontece?

A razão para essa perda está na forma como a média móvel é calculada. Para cada ponto da série, a média é calculada com base em h valores próximos. Por exemplo, se $h = 5$, podemos usar os 2 valores anteriores e os 2 valores seguintes, ou algum outro arranjo próximo, dependendo da posição. Porém, em alguns pontos da série, especialmente no início e no final, não existem vizinhos suficientes para calcular a média. Isso resulta em uma perda de valores nas extremidades da série.

Veja, por exemplo, o que acontece ao calcular a média móvel de ordem 5:

Figura 28: Série original (primeiros valores).

Time Series:																
Start = c(2023, 1)																
End = c(2023, 48)																
Frequency = 52																
[1]	1	2	2	3	8	7	5	14	12	17	16	4	2	5	1	5
[17]	7	5	10	18	29	41	39	73	53	40	81	112	138	178	215	268
[33]	315	274	215	197	158	105	104	85	63	36	13	32	23	15	12	14

Média centrada, k=5: olhamos para os dois valores à direita e os dois valores à esquerda, para realizar a média

Figura 29: Série suavizada com média móvel de ordem 5 (primeiros valores).

Time Series:											
Start = c(2023, 3)											
End = c(2023, 45)											
Frequency = 52											
[1]	NA	NA	3.2	4.4	5.0	7.4	9.2	11.0	12.8	12.6	10.2
[12]	8.8	5.6	3.4	4.0	4.6	5.6	9.0	13.8	20.6	27.4	40.0
[23]	47.0	49.2	57.2	71.8	84.8	109.8	144.8	182.2	222.8	250.0	257.4
[34]	253.8	231.8	189.8	155.8	129.8	103.0	78.6	60.2	45.8	33.4	

Série agora inicia na SE 3/2023

Não possui dois valores à esquerda devido ao cálculo da média

Nesse caso, podemos ver que os primeiros valores da série original não podem ser suavizados, pois não há vizinhos suficientes para calcular a média. Isso também ocorrerá com os últimos valores da série.



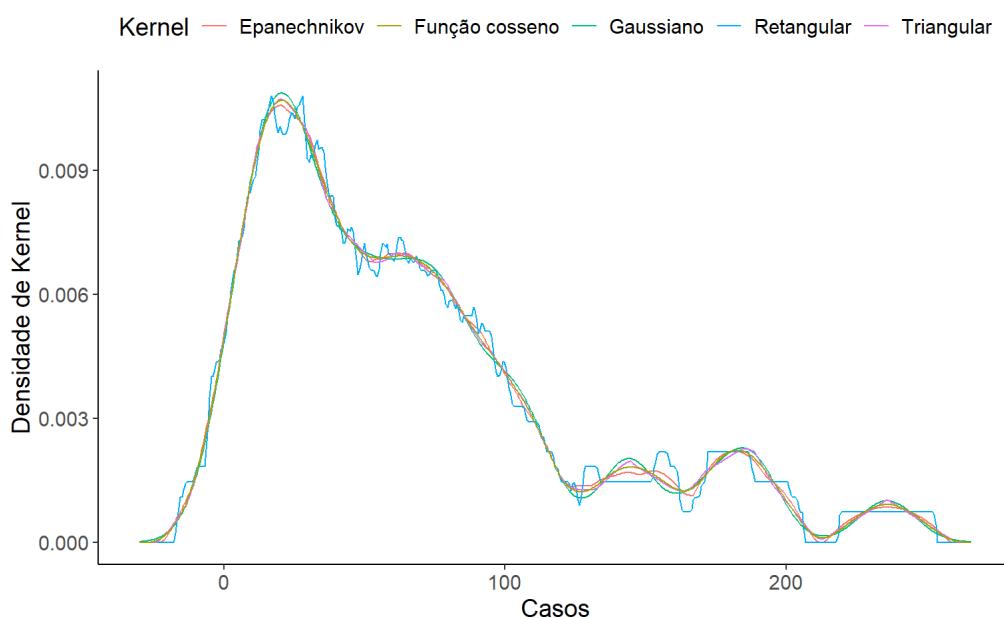
Kernel

Outra possível técnica de suavização de uma série temporal é a suavização por Kernel. Essa técnica está inclusa em métodos mais avançados que ajustam curvas suaves aos dados, levando em conta as vizinhanças locais. Essas técnicas são mais flexíveis e são particularmente úteis quando a série tem padrões complexos.

Um Kernel é uma função ponderada que é aplicada sobre cada ponto de uma série temporal e seus vizinhos. Essa função atribui **mais peso ao ponto de interesse e menores pesos aos pontos mais distantes**, suavizando a série. O formato da função de suavização (Kernel) e o tamanho da janela utilizada, chamado de **largura de banda**, determinam o tipo e a intensidade da suavização. Diferentes tipos de Kernel podem ser usados, como o **Gaussiano, Retangular, Triangular, Cosseno, ou Epanechnikov**, cada um proporcionando uma forma diferente de suavização.

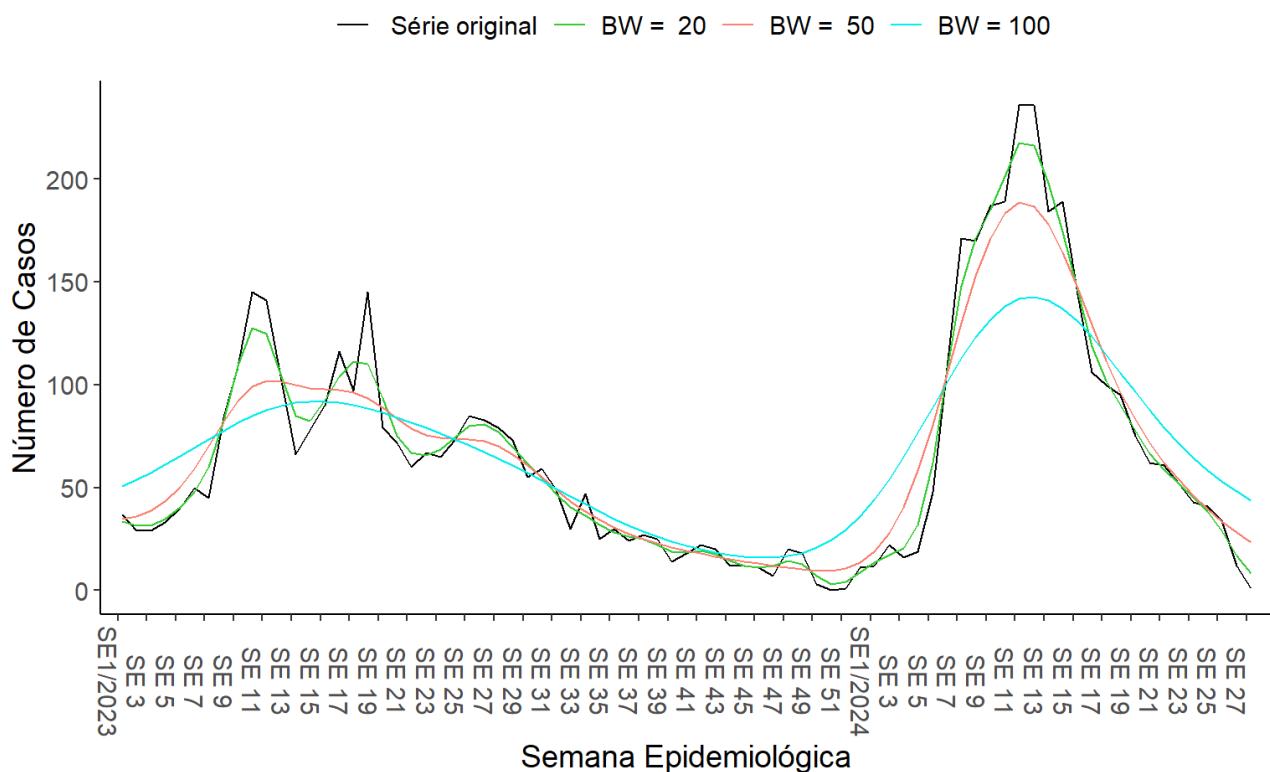
A escolha da função Kernel e da largura de banda tem impacto direto na forma como a suavização ocorre. Repare na Figura 30 como o comportamento de suavização pode variar conforme a função escolhida:

Figura 30: Casos prováveis de dengue, São Luís-MA, 2023 e 2024.



O parâmetro de largura de banda (*Bandwidth - BW*) é o que define a quantidade de suavização aplicada ao kernel. Vamos visualizar, na Figura 31, a aplicação de variações desse parâmetro à série de casos prováveis de dengue no Maranhão:

Figura 31: Variações da largura de banda: Casos prováveis de dengue, São Luís-MA, 2023 e 2024.



Percebe-se que uma largura de banda maior implica numa maior suavização. A curva verde, com $BW = 20$, se aproxima das variações de médio termo da série, enquanto as suavizações de $BW = 50$ ou $BW = 100$ reproduzem uma tendência de mais longo termo.

Loess

Loess (Locally Estimated Scatterplot Smoothing) e **Lowess** (Locally Weighted Scatterplot Smoothing) são métodos de suavização semelhantes ao Kernel, mas, em vez de usar médias ponderadas, esses métodos fazem **regressões locais** para estimar os valores suavizados.

No caso do **Loess**, para cada ponto da série, um polinômio é ajustado a um subconjunto de dados utilizando o método dos mínimos quadrados, resultando em uma curva suave. Já no **Lowess**, uma reta é ajustada, mas utilizando mínimos quadrados ponderados, dando mais peso aos pontos mais próximos e menos peso aos mais distantes.

O subconjunto de dados usado em cada estimativa é chamado de **janela** ou **largura de banda** (também conhecido como **parâmetro de suavização**). Esse parâmetro controla a flexibilidade da função de regressão. Se a janela for muito ampla (muitos dados considerados se igualando ao total de pontos, por exemplo), o alisamento será maior; se a janela for menor, a suavização será mais limitada. Quanto maior a janela, maior o alisamento da série.

A largura da janela é expressa como uma fração dos dados, variando de 0 a 1. Por exemplo, uma janela de 0,1 indica que 10% dos dados no eixo horizontal estão sendo usados para estimar o valor suavizado em cada ponto.

Outro aspecto importante é o grau do polinômio utilizado nas regressões locais:

- Polinômio de grau 0: é equivalente a uma média móvel simples.
- Polinômio de grau 1: resulta em uma regressão linear local.
- Polinômio de grau 2: resulta em uma regressão quadrática local.

Se a janela incluir 100% dos pontos, o resultado será equivalente a calcular a média simples de toda a série.

Vantagens:

- **Simples e eficaz:** Loess e Lowess são ótimos para análise exploratória, fornecendo uma curva suave e flexível.
- **Menos sensível às bordas:** Essas técnicas lidam bem com pontos próximos às extremidades da série, onde pode ser mais impreciso.

Desvantagens: - **Sensibilidade a valores extremos:** O método pode ser influenciado por outliers, o que pode distorcer a suavização.

Splines

Os **Splines** são um conjunto de funções polinomiais que têm diversos usos na análise de dados. No nosso caso, os splines são utilizados como uma técnica de suavização, permitindo que diferentes segmentos da série temporal sejam ajustados de forma suave e contínua. Eles são particularmente úteis porque permitem unir diferentes pontos, chamados de nós, de forma suave, mantendo propriedades matemáticas ideais.

A mais usada para suavização é a *spline cúbica natural*. Trata-se de uma forma de regressão penalizada, em que os nós estão posicionados em diferentes valores de x_i . A suavização entre esses nós ocorre por meio de funções polinomiais cúbicas, que garantem uma transição suave entre os segmentos da série.

A escolha do parâmetro de suavização (ou seja, quanto “suave” a curva deve ser) pode ser feita de maneira visual, observando o gráfico, ou de forma mais formal, utilizando métodos que minimizam o erro quadrático médio ou através de técnicas de validação cruzada.

Na regressão penalizada, o objetivo é encontrar a função $\hat{f}(x)$ que minimize o seguinte:

$$\sum [y_i - f(x_i)]^2 + \tau \int [f''(x)]^2 dx$$

Sendo τ o parâmetro de alisamento:

- Se $\tau = 0 \Rightarrow \hat{f}(x)$ é interpolação pontual (conecta os pontos);
- Se $\tau = 0 \Rightarrow \hat{f}(x)$ é interpolação linear simples;
- Se $\tau \neq 0 \Rightarrow \hat{f}(x)$ será tal que $f''(x)$ é muito grande será tal que seja zero em todos os pontos, ou seja, mínimos quadrados.



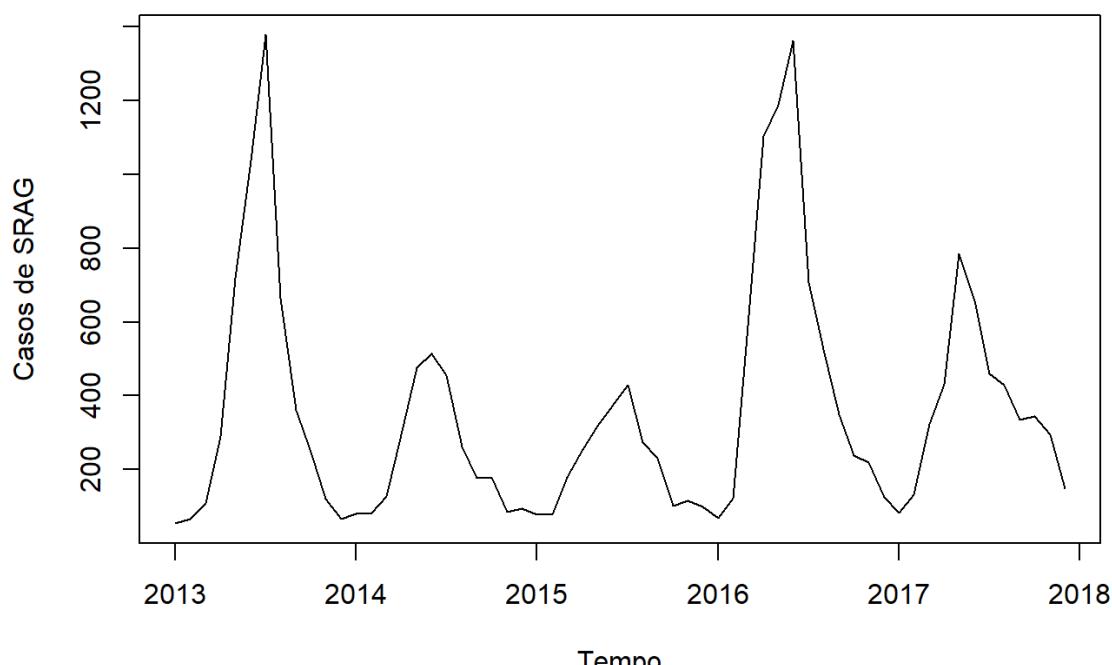
Splines vs Loess/Lowess

- No *spline* se minimiza uma função-objetivo explícita, é mais elegante matematicamente que *loess/lowess*.
- O ajuste dos dois é muito semelhante para o mesmo número de graus de liberdade.
- Pode-se ajustar essas funções para diversos preditores.
- Existe uma forma mais complexa de spline mais complexo para splines (thin plate splines) que pode ser usada em modelos GAM.

Prática em R: Transformações e suavizações

Vamos continuar com nossa série de SRAG no Paraná:

```
# Visualizando a série temporal
plot(srág_pr_ts, ylab = "Casos de SRAG", xlab = "Tempo")
```



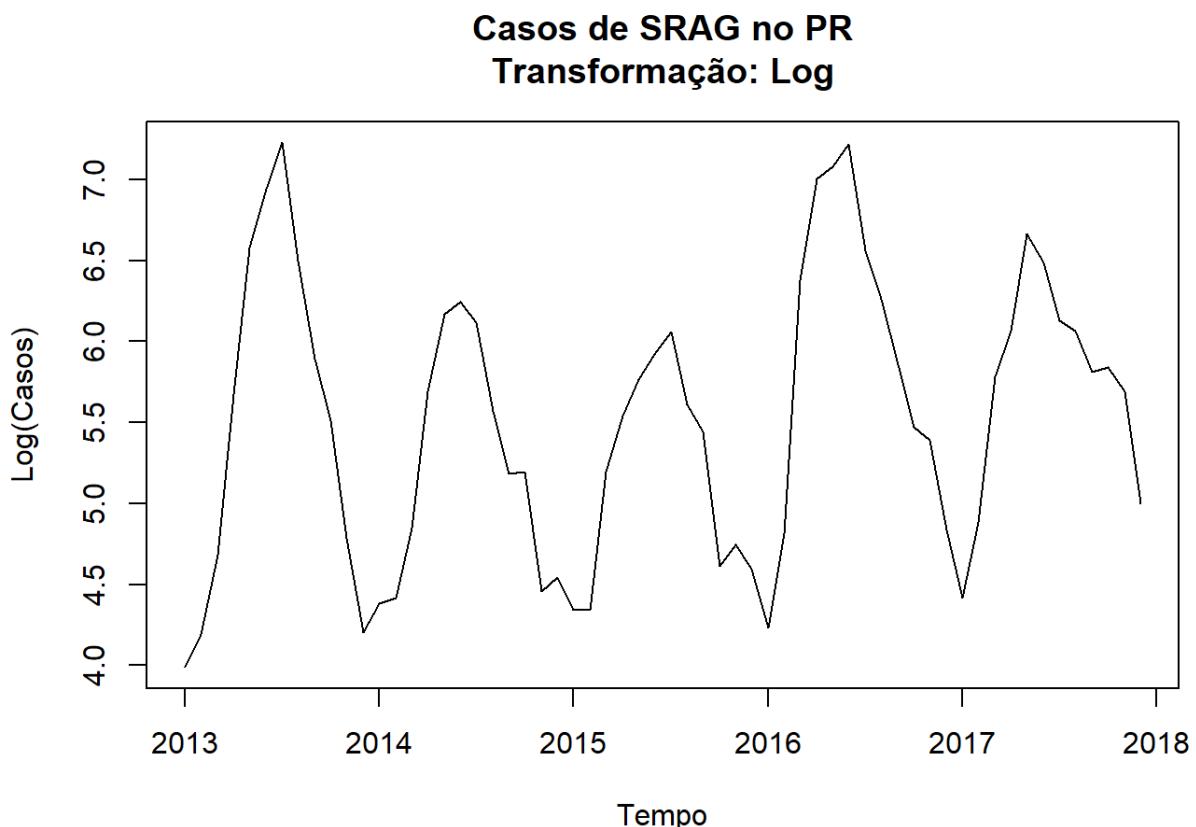


O comando `plot(srág_pr_ts, ylab = "Casos de SRAG")` gera um gráfico de linha da série temporal `srág_pr_ts`, com o eixo Y rotulado como “Casos de SRAG”.

Vamos aplicar as transformações vistas à série: transformação logarítmica, Box-Cox e diferenciação.

Para a transformação logarítmica podemos usar a própria função `log()` do R:

```
srág_pr_log<- log(srág_pr_ts)
plot(srág_pr_log, ylab = "Log(Casos)", xlab = "Tempo", main = "Casos de SRAG no PR\\nTransformação: Log")
```



Este código aplica uma transformação logarítmica aos dados da série temporal `srag_pr_ts` e exibe o gráfico da série transformada.

- `srag_pr_log ← log(srag_pr_ts)`: Cria uma nova série, `srag_pr_log`, contendo os logaritmos dos valores de `srag_pr_ts`, suavizando variações extremas e tornando os dados mais adequados para análises de tendência e sazonalidade.
- `plot(srag_pr_log, ylab = "Log(Casos)", main = "Casos de SRAG no PR\\nTransformação: Log")`: Plota a série transformada com o rótulo do eixo Y como "Log(Casos)" e o título "Casos de SRAG no PR Transformação: Log". O "n" é utilizado para quebrar a linha do título.

Vemos como a diferença entre os picos mais altos e dos mais baixos já não é mais tão grande, ou seja, há uma redução na amplitude de variação nos dados. Agora, vamos verificar como a série fica após a transformação de Box-Cox, e para isso usaremos o pacote `forecast`.



*** ATENÇÃO:** De agora em diante, usaremos de forma intensiva o pacote `forecast`, que contém um conjunto de funções e utilidades na tarefa de análise de séries temporais. Caso ele não esteja instalado em seu ambiente R, basta executar o código abaixo. Acompanhe:

```
install.packages("forecast")
```

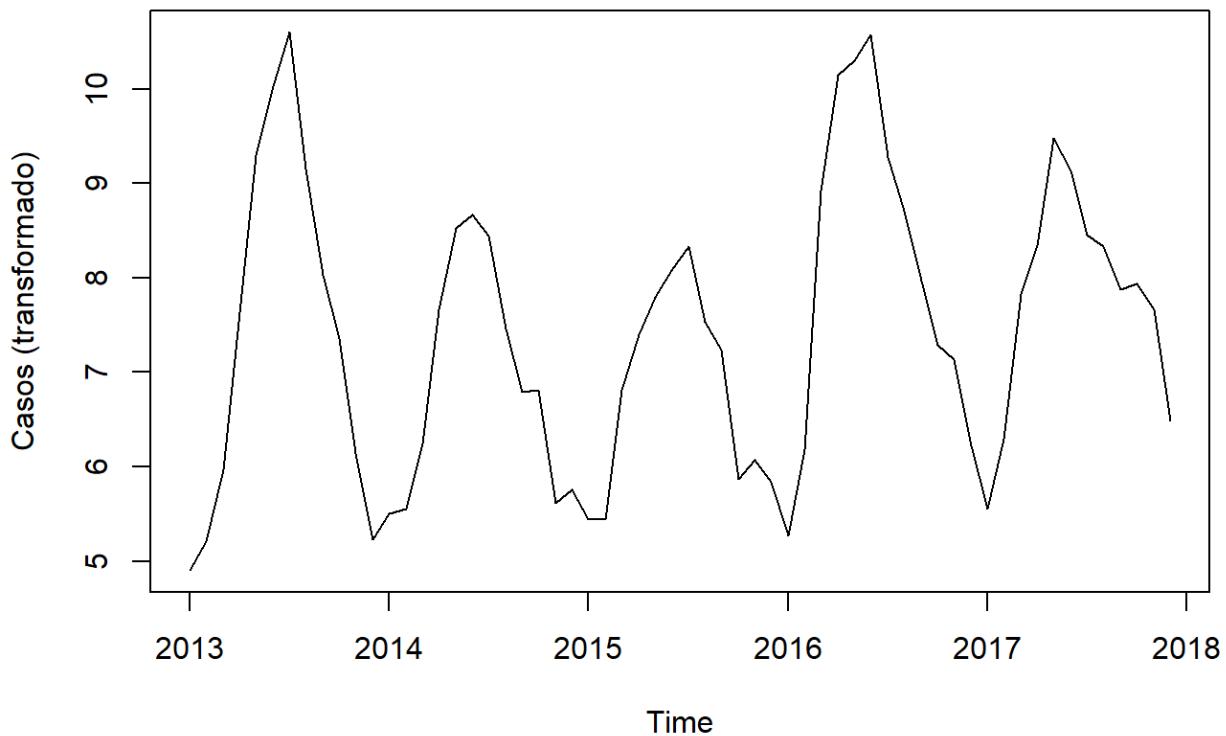
```
# Carregando o pacote
library(forecast)

# Aplicando a transformação de Box-Cox
srag_pr_bc1 ← BoxCox(srag_pr_ts, lambda = 0.1)

# Visualizando a série transformada
plot(srag_pr_bc1,
      ylab = "Casos (transformado)",
      main = "Casos de SRAG no PR\\nTransformação: BoxCox, Lambda = 0.1")
```



Casos de SRAG no PR Transformação: BoxCox, lambda = 0.1



Este código aplica uma transformação de Box-Cox na série temporal `srag_pr_ts` para estabilizar a variância e facilitar a modelagem.

- `library(forecast)`: Carrega o pacote forecast, que contém a função BoxCox.
- `BoxCox(srag_pr_ts, lambda = 0.1)`: Aplica a transformação de Box-Cox com parâmetro `lambda = 0.1` na série `srag_pr_ts`, criando uma versão transformada, `srag_pr_bc1`.
- `plot()`: Exibe a série transformada com rótulo “Casos (transformado)” e título indicando a transformação.



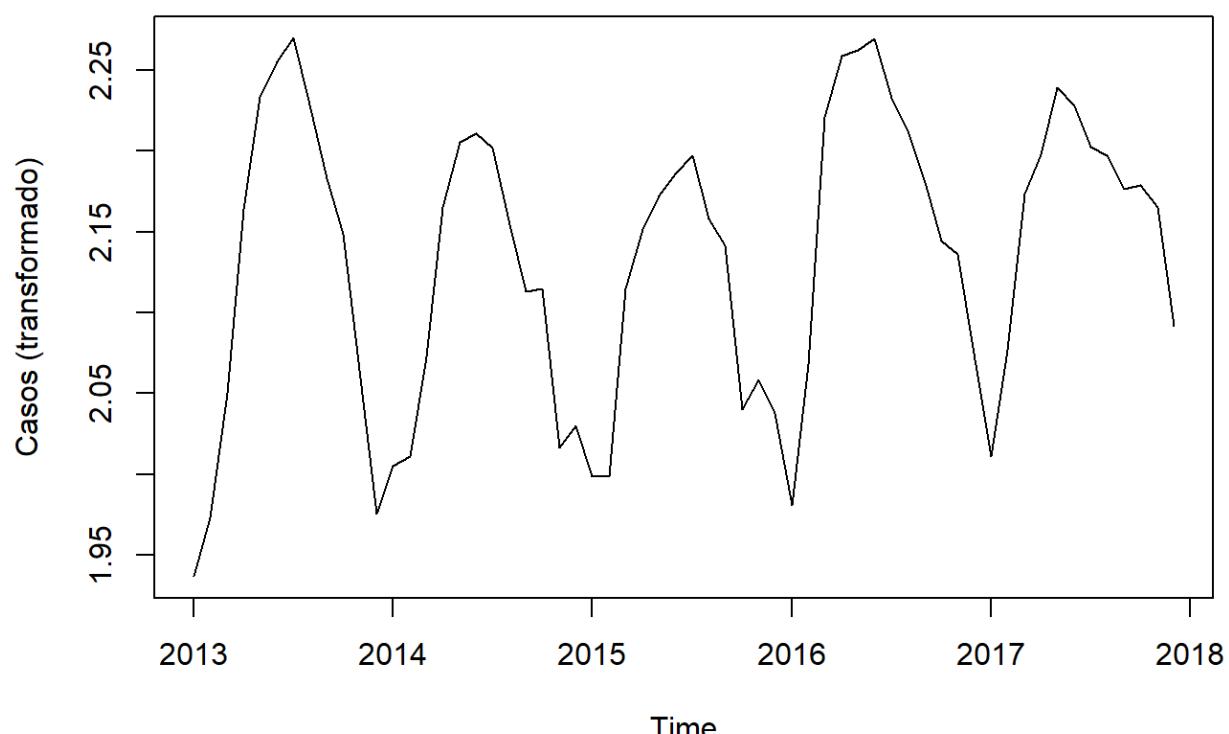
Conforme vimos anteriormente, a transformação de Box-Cox requer o parâmetro `lambda`, que determina a forma da transformação. No exemplo acima usamos `lambda = 0.1`. Podemos testar diferentes valores do parâmetro e comparar os resultados. Podemos testar também, conforme visto anteriormente, o lambda automático, através do comando `BoxCox.lambda()`:

```
# Aplicando a transformação de Box-Cox
lambda_auto <- BoxCox.lambda(srág_pr_ts)

srág_pr_bc2 <- BoxCox(srág_pr_ts, lambda = lambda_auto)

# Visualizando a série transformada
plot(srág_pr_bc2,
      ylab = "Casos (transformado)",
      main = paste0(
        "Casos de SRAG no PR\nTransformação: BoxCox, lambda = ",
        round(lambda_auto, 2)
      )
)
```

**Casos de SRAG no PR
Transformação: BoxCox, lambda = -0.42**



Esse código aplica a transformação de Box-Cox na série temporal `srag_pr_ts` para estabilizar a variância:

- `lambda_auto ← BoxCox.Lambda(srag_pr_ts)`: Calcula o parâmetro `lambda` ideal para a transformação de Box-Cox, armazenando-o em `lambda_auto`.
- `srag_pr_bc2 ← BoxCox(srag_pr_ts, lambda = lambda_auto)`: Aplica a transformação de Box-Cox usando o `lambda` calculado para criar `srag_pr_bc2`, a série transformada.
- `plot()`: Plota a série transformada com um título que exibe o valor de `lambda` arredondado.

Vemos que a diferença de amplitude entre as curvas diminui, aproximando-as. Como visto anteriormente, tais técnicas de transformação podem ser úteis nas etapas de modelagem da série temporal.

Podemos aplicar a última transformação vista: a diferenciação. Vimos que a diferenciação é útil para remoção da tendência da série, contudo já percebemos durante a última prática que a série não possui tendência relevante. Assim, esperamos uma série parecida ao diferenciá-la. Vamos testar por meio do comando `diff()`:

```
par(mfrow = c(2,1))

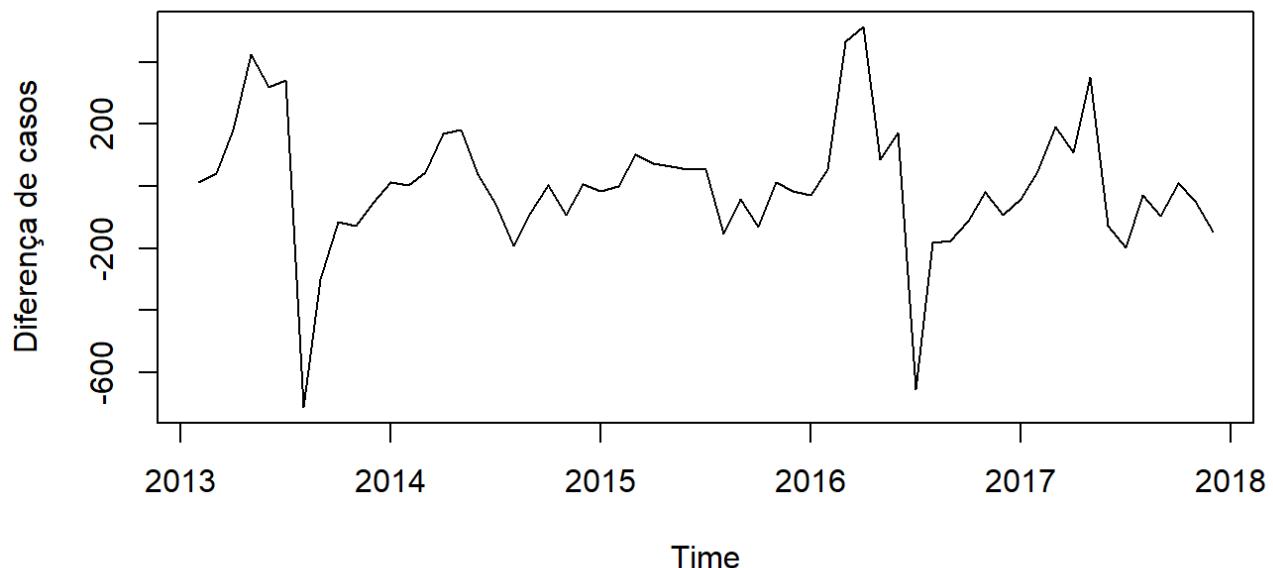
# Diferenciação da série original
srag_pr_diff ← diff(srag_pr_ts)

plot(
  srag_pr_diff,
  ylab = "Diferença de casos",
  main = "a. Casos de SRAG no PR\nTransformação: Diferenciação"
)

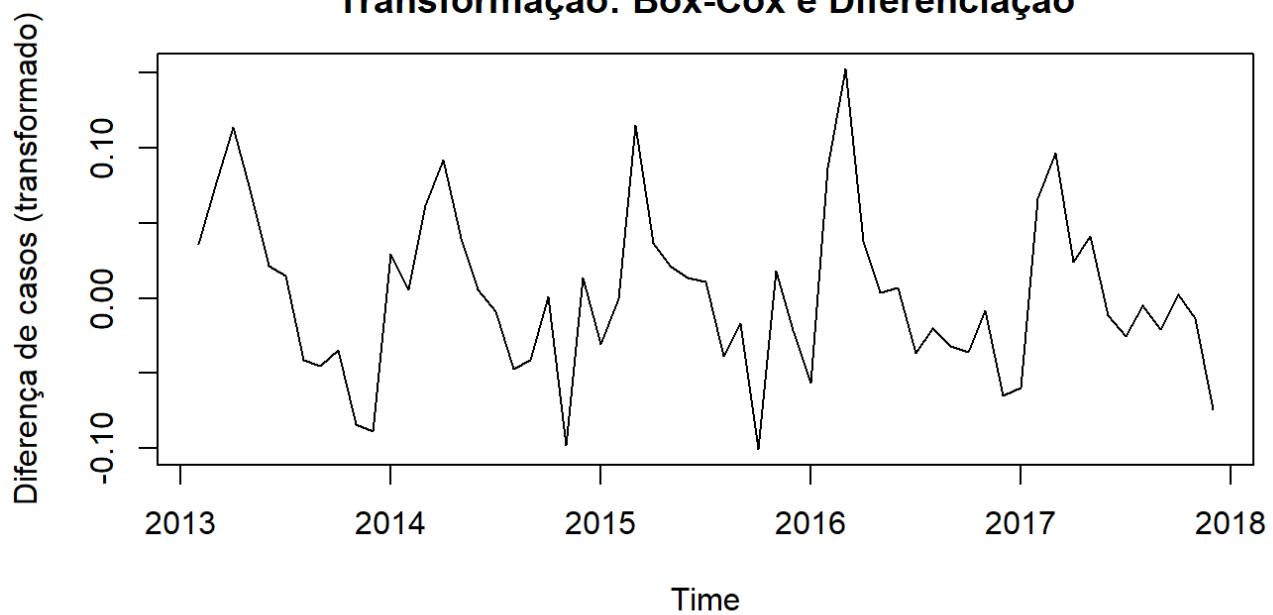
# Diferenciação da série já transformada
srag_pr_bc_diff ← diff(srag_pr_bc2)

plot(
  srag_pr_bc_diff,
  ylab = "Diferença de casos (transformado)",
  main = "b. Casos de SRAG no PR\nTransformação: Box-Cox e Diferenciação"
)
```

**a. Casos de SRAG no PR
Transformação: Diferenciação**



**b. Casos de SRAG no PR
Transformação: Box-Cox e Diferenciação**



Esse código cria um painel com dois gráficos para comparar transformações de diferenciação na série temporal de casos de SRAG:

- `par(mfrow = c(2,1))`: Define a disposição dos gráficos em duas linhas e uma coluna, permitindo a visualização dos dois gráficos um sobre o outro.
- `srag_pr_diff ← diff(srag_pr_ts)`: Calcula a primeira diferenciação da série `srag_pr_ts`, capturando mudanças entre períodos consecutivos.
- `plot()`: Plota a série diferenciada, mostrando as diferenças nos casos de SRAG e destacando variações temporais.
- `srag_pr_bc_diff ← diff(srag_pr_bc2)`: Calcula a primeira diferenciação da série transformada por Box-Cox `srag_pr_bc2`, uma técnica que ajusta a distribuição antes da diferenciação.
- `plot()`: Plota a série diferenciada transformada, mostrando as diferenças após ambas as transformações.

Vemos que a diferenciação traz a média dos casos para zero e de uma forma constante, sem qualquer tendência (a). Contudo, somente esta transformação mantém a diferença na amplitude entre as curvas. Buscando estabilizar isso, transformamos também a série transformada por meio do método Box-Cox (b).



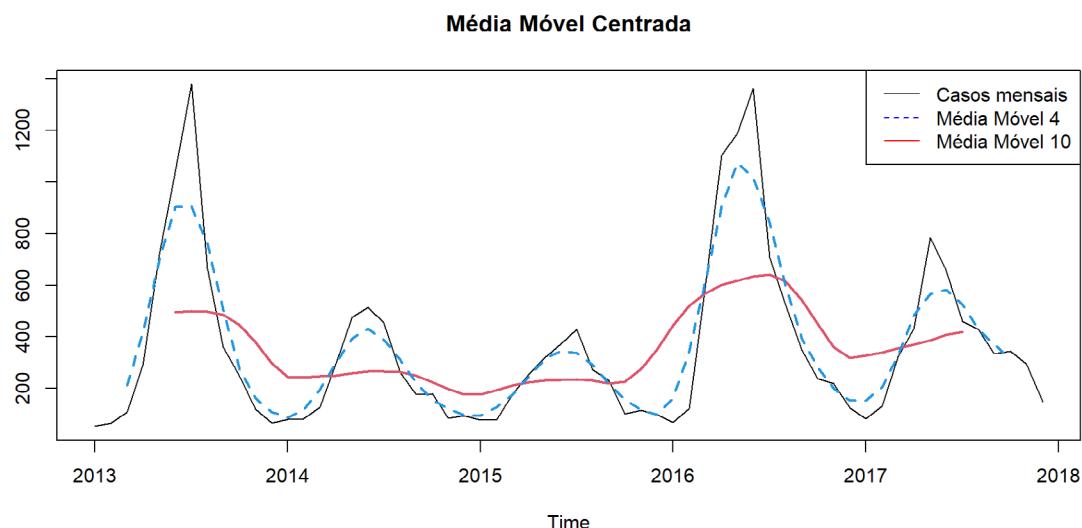
Suavizações

Podemos comparar também, diferentes estratégias vistas para obter uma suavização da série.

Médias móveis

Podemos obter a série suavizada por médias móveis por meio da função `ma()`, também do pacote `forecast`.

```
plot(  
  srag_pr_ts,  
  ylab = "",  
  main = "Média Móvel Centrada"  
)  
lines(ma(srag_pr_ts, order = 10), col = 2, lty = 1, lwd = 2)  
lines(ma(srag_pr_ts, order = 4), col = 4, lty = 2, lwd = 2)  
legend("topright",  
      legend = c("Casos mensais", "Média Móvel 4", "Média Móvel 10"),  
      col = c("black", "blue", "red"),  
      lty = c(1, 2, 1),  
      cex = 1)
```



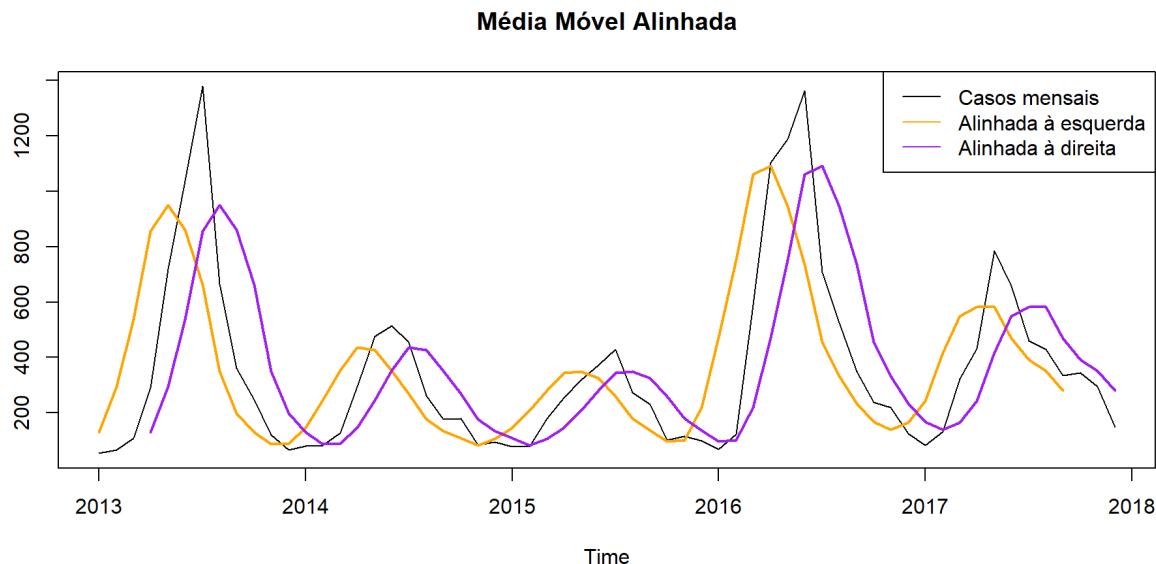
Esse código gera um gráfico da série temporal `srag_pr_ts` com médias móveis centradas para suavizar e destacar tendências de curto e longo prazo.

- `plot(srag_pr_ts, ylab = "", main = "Média Móvel Centrada")`: Plota a série temporal `srag_pr_ts` com o título “Média Móvel Centrada”.
- `lines(ma(srag_pr_ts, order = 10), col = 2, lty = 1, lwd = 2)`: Adiciona uma linha de média móvel de ordem 10 (longa) em vermelho (`col = 2`), linha contínua (`lty = 1`) e espessura 2 (`lwd = 2`).
- `lines(ma(srag_pr_ts, order = 4), col = 4, lty = 2, lwd = 2)`: Adiciona uma linha de média móvel de ordem 4 (curta) em azul (`col = 4`), linha tracejada (`lty = 2`) e espessura 2.
- `Legend()`: Adiciona uma legenda no canto superior direito que identifica a série original, a média móvel de ordem 4 e a de ordem 10.

Vemos que a média móvel de ordem $h = 4$ captura bem as variações importantes e suaviza a série de forma satisfatória, enquanto a média de ordem $h = 10$ realiza uma suavização mais intensa, de forma a perder algumas flutuações relevantes na série. Note que o período que perdemos ao aumentar o valor de h é maior, de forma que a série em vermelho se inicia em meados de 2013 e vai até meados de 2017.

Também vimos que podemos alinhar as médias móveis à esquerda ou à direita, conforme o que for mais adequado. No R, podemos trocar esse alinhamento com a função `rollmean()`, do pacote `zoo`. Ela também é uma função para médias móveis, porém com mais parâmetros:

```
library(zoo)
plot(srag_pr_ts, ylab = "", main = "Média Móvel Alinhada")
lines(rollmean(srag_pr_ts, k = 4, align = 'left'), col = "orange", lwd = 2)
lines(rollmean(srag_pr_ts, k = 4, align = 'right'), col = "purple", lwd = 2)
Legend("topright", legend = c("Casos mensais", "Alinhada à esquerda", "Alinhada à
direita"),
      col = c("black", "orange", "purple"), lty = c(1, 1, 1), cex = 1)
```



Esse código utiliza o pacote zoo para plotar a série temporal `srag_pr_ts` com médias móveis alinhadas à esquerda e à direita, destacando variações suavizadas na série.

- `plot(srag_pr_ts, ylab = "", main = "Média Móvel Alinhada")`: Plota a série temporal com o título “Média Móvel Alinhada”.
- `Lines(rollmean(srag_pr_ts, k = 4, align = 'left'), col = "orange", lwd = 2)`: Adiciona uma linha de média móvel de 4 períodos alinhada à esquerda, em laranja e espessura de 2.
- `Lines(rollmean(srag_pr_ts, k = 4, align = 'right'), col = "purple", lwd = 2)`: Adiciona uma linha de média móvel de 4 períodos alinhada à direita, em roxo e espessura de 2.
- `Legend()`: Adiciona uma legenda no canto superior direito para identificar as linhas.

Kernel

Vimos que outra possibilidade para obter uma suavização da série é a utilização de uma função de kernel. Há inúmeras possibilidades de realizar essa operação em R, sendo a mais simples por meio da utilização da função `ksmooth()`. Nela, passamos os valores de `x` e `y` da relação que queremos suavizar. Como estamos tratando de uma série temporal, nosso valor de `x` representa o momento da série temporal (podemos obtê-lo com a função `time()`), e `y` é o próprio valor da série. Dessa forma, podemos implementar uma suavização por kernel, variando os valores da largura de banda, que determina a quantidade de suavização:

```

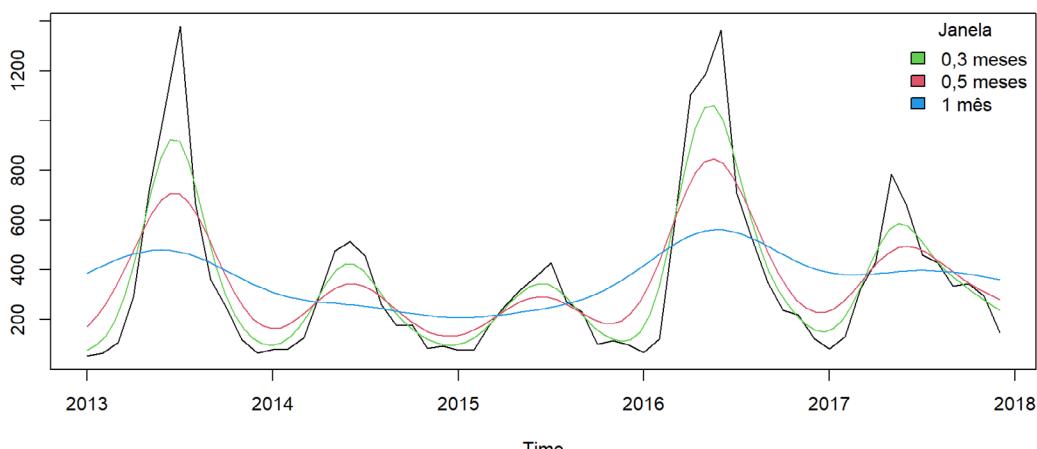
kernel1 <- ksmooth(time(srág_pr_ts), srág_pr_ts, kernel = "normal", bandwidth = 0.3)
kernel2 <- ksmooth(time(srág_pr_ts), srág_pr_ts, kernel = "normal", bandwidth = 0.5)
kernel3 <- ksmooth(time(srág_pr_ts), srág_pr_ts, kernel = "normal", bandwidth = 1)

plot(
  x = srág_pr_ts,
  yLab="",
  main="Suavização por Kernel"
)
lines(kernel1$x, kernel1$y, col = 3)
lines(kernel2$x, kernel2$y, col = 2)
lines(kernel3$x, kernel3$y, col = 4)

legend("topright",c("0,3 meses","0,5 meses","1 mês"),fill=c(3,2,4), title="Janela", bty="n")

```

Suavização por Kernel



Esse código aplica a suavização por kernel à série temporal `srag_pr_ts` com diferentes larguras de banda para analisar tendências:

- `ksmooth()`: Suaviza a série temporal usando o kernel “normal” com três larguras de banda distintas (0.3, 0.5 e 1). Larguras de banda menores captam mais detalhes, enquanto maiores produzem uma suavização mais geral.
- `plot()`: Plota a série original.
- `lines()`: Adiciona as linhas suavizadas de `kernel1`, `kernel2` e `kernel3` com cores diferentes.
- `legend()`: Adiciona uma legenda no canto superior direito, indicando as diferentes larguras de banda e suas respectivas cores.

Vemos que, conforme aumentamos a largura de banda, a suavização aplicada é maior. O kernel com largura de 1 mês, por exemplo, captura apenas as variações mais longas da série.

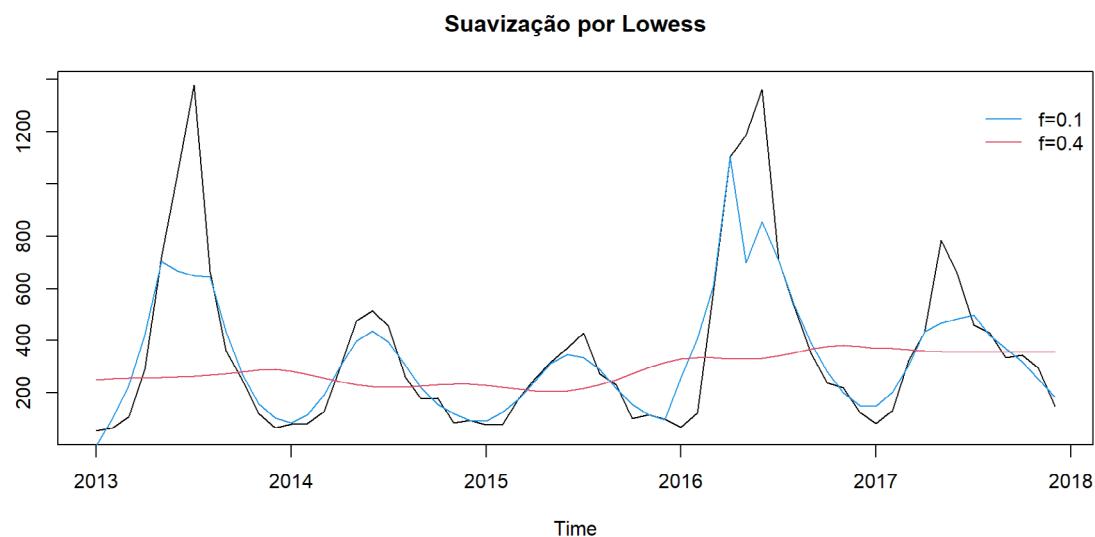
Lowess

A suavização por meio do método lowess se dá de forma parecida em R, com a função `lowess()`. Neste caso, apenas temos que escrever nossos parâmetros na forma de uma fórmula, ou seja: `y ~ x`. Veja abaixo:



```
lowess1 <- lowess(srags_pr_ts ~ time(srags_pr_ts), f = 0.1)
lowess2 <- lowess(srags_pr_ts ~ time(srags_pr_ts), f = 0.4)

plot(srags_pr_ts, ylab = "", main = "Suavização por Lowess")
lines(lowess1$x, lowess1$y, col = 4)
lines(lowess2$x, lowess2$y, col = 2)
legend(
  "topright",
  c("f=0.1", "f=0.4"),
  col = c(4, 2),
  title = "",
  bty = "n",
  lty = c(1, 1)
)
```



Esse código aplica a suavização Lowess na série temporal `srag_pr_ts` usando duas intensidades de suavização diferentes ($f = 0.1$ e $f = 0.4$), e plota os resultados para comparação.

- `lowess1` e `lowess2`: Guardam as suavizações de `srag_pr_ts` com os parâmetros $f = 0.1$ (mais sensível às variações) e $f = 0.4$ (mais suave), onde f controla a fração dos dados usada em cada ponto de suavização.
- `plot()`: Plota a série temporal original com o título “Suavização por Lowess”.
- `lines(lowess1$x, lowess1$y, col = 4)` e `lines(lowess2$x, lowess2$y, col = 2)`: Adicionam as linhas de suavização para `lowess1` (em azul) e `lowess2` (em vermelho).
- `Legend()`: Coloca uma legenda no canto superior direito (`topright`), indicando os valores de f usados para cada linha.

De forma similar ao aumentarmos o parâmetro f , que determina a janela de suavização, temos uma suavização mais expressiva. Ao utilizarmos um valor menor, como $f = 0.1$, vemos que alguns pontos ainda mantêm uma variação brusca.

Splines

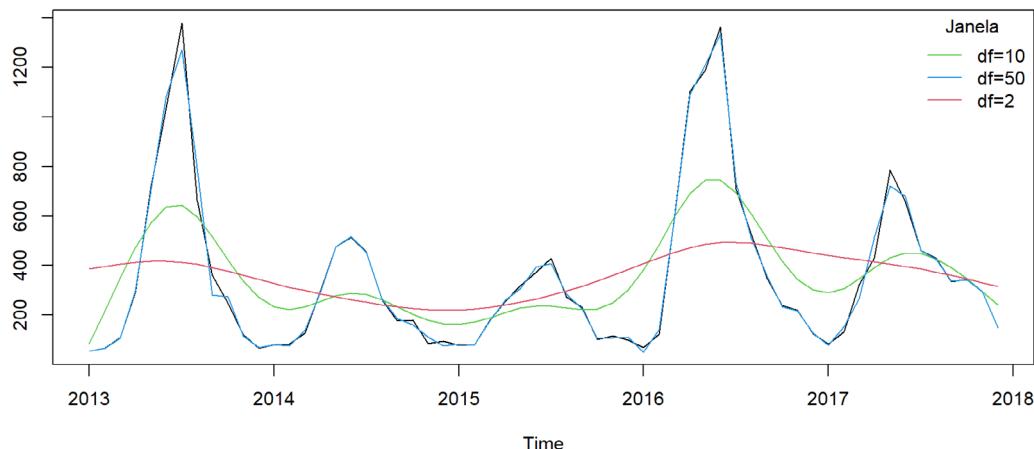
A função utilizada para ajustar splines é a `smooth.spline`. Tentemos comparar alguns cenários com diferentes graus de liberdade (df):



```
spline1 <- smooth.spline(time(srág_pr_ts), srág_pr_ts, df = 10)
spline2 <- smooth.spline(time(srág_pr_ts), srág_pr_ts, df = 50)
spline3 <- smooth.spline(time(srág_pr_ts), srág_pr_ts, df = 5)

plot(srág_pr_ts, ylab = "", main = "Suavização por Spline")
lines(spline1$x, spline1$y, col = 3)
lines(spline2$x, spline2$y, col = 4)
lines(spline3$x, spline3$y, col = 2)
legend(
  "topright",
  c("df=10", "df=50", "df=2"),
  col = c(3, 4, 2),
  lty = c(1, 1, 1),
  title = "Janela",
  bty = "n"
)
```

Suavização por Spline



Esse código aplica suavizações por splines com diferentes graus de liberdade (df) na série temporal `srág_pr_ts`, e plota as linhas suavizadas no gráfico para comparação.

- `spline1, spline2, spline3`: Criam três suavizações por splines usando `smooth.spline`, com graus de liberdade de 10, 50 e 5, respectivamente, ajustando diferentes níveis de suavidade.
- `plot(srág_pr_ts, ylab = "", main = "Suavização por Spline")`: Plota a série temporal original sem rótulo no eixo Y e com o título "Suavização por Spline".
- `lines()`: Adiciona ao gráfico as três curvas suavizadas (verde, azul e vermelha) usando os resultados de `spline1, spline2 e spline3`.
- `legend()`: Adiciona uma legenda no canto superior direito, especificando os diferentes graus de liberdade (10, 50 e 5) e suas cores.

Vemos que as splines produzem curvas suaves e contínuas, que se ajustam aos padrões dos dados. Diferentemente das outras abordagens, vemos que ao aumentar os graus de liberdade (df), temos curvas menos suaves e mais ajustadas aos dados, enquanto poucos graus mostram uma tendência mais geral dos dados.

Pronto! Agora você já sabe tudo sobre os componentes de uma série temporal, como transformá-la para reduzir sua variação ou remover sua tendência, e estratégias de suavização para obter padrões mais claros. No próximo módulo, vamos aprender a modelar as séries temporais para ajudar a descrevê-la, testar associações e realizar previsões.

Módulo 5 - Introdução à modelagem de séries temporais

Até agora, aprendemos a **explorar e descrever séries temporais**, identificando se uma série possui **tendência, sazonalidade** ou se é **estacionária**. Também vimos como **transformações** podem ser úteis para preparar os dados antes da modelagem. Mas como podemos ir além da descrição e **utilizar esses dados para prever cenários futuros e apoiar a tomada de decisão na vigilância em saúde?**

Neste módulo, vamos explorar um dos modelos mais amplamente utilizados para **análise e previsão de séries temporais: o ARIMA**. Esse modelo é essencial na **vigilância epidemiológica**, pois permite não apenas identificar padrões nos dados temporais, mas também projetar tendências futuras, ajudando a prever o comportamento de doenças e orientar ações de saúde pública. Por exemplo, ele pode ser usado para antecipar aumentos nos casos de uma doença com base em dados históricos, auxiliando no planejamento de recursos e respostas mais eficazes.

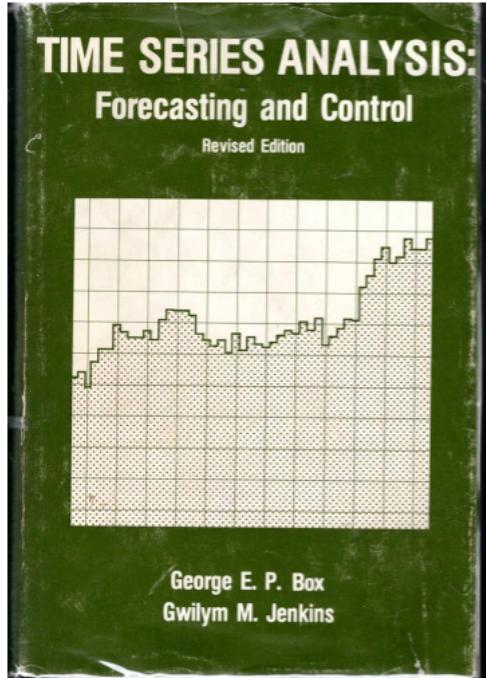
Além do ARIMA, veremos uma introdução aos **Modelos Aditivos Generalizados (GAM)**, uma abordagem mais flexível que permite capturar relações não lineares nos dados. Essa técnica é útil para análises mais detalhadas, especialmente quando os padrões observados não seguem um comportamento estritamente linear.

Ao final deste módulo, você compreenderá como esses modelos podem ser aplicados na prática para **monitoramento de doenças, previsão de surtos e planejamento de intervenções em saúde pública**. Vamos lá?

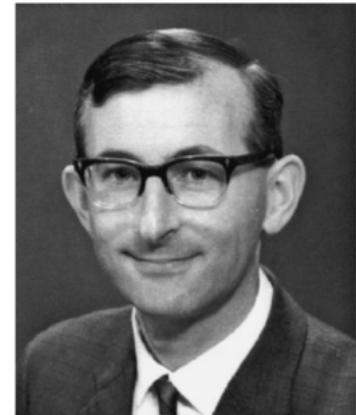
Modelo Box & Jenkins - o modelo ARIMA

Os modelos ARIMA foram introduzidos e popularizados por George Box e Gwilym Jenkins, em 1970, e são descritos no livro *"Time Series Analysis: Forecasting and Control"*, dos autores (Figura 32).

Figura 32: Livro *Time Series Analysis: Forecasting and Control* (1970) e os autores George Box e Gwilym Jenkins.



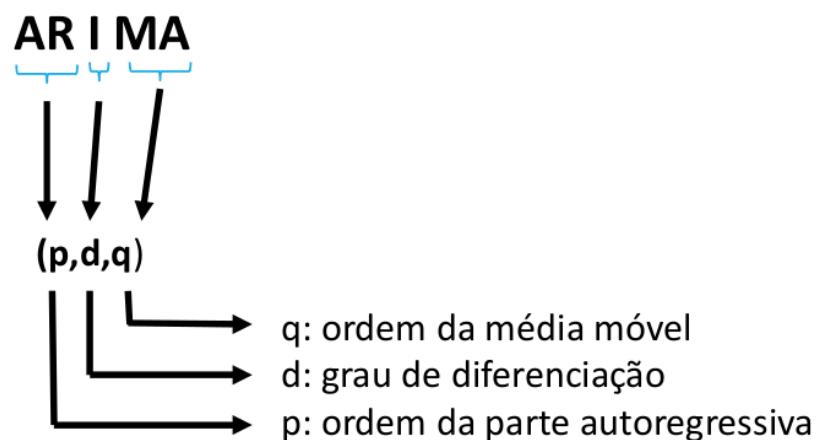
George E. P. Box



Gwilym M. Jenkins

A sigla **ARIMA** resume a estrutura básica do modelo: é composta por termos autorregressivos (**AR**), por termos de diferenciação (ou Integração, **I**) e de médias móveis (**MA**, do inglês *moving averages*). Logo, um modelo ARIMA pode sempre ser descrito pela forma $ARIMA(p, d, q)$, de forma que p , d e q representam a **ordem** de cada um desses termos (Figura 33).

Figura 33: Termos de um modelo ARIMA.



A seguir vamos nos debruçar sobre o que cada um desses termos representam.

Termo AR (autorregressivo)

O termo **AR (autorregressivo)** se refere à dependência que a série temporal possui de seus próprios **valores passados**. Ou seja, por meio dele incluímos a relação do valor da série em um momento t com seu valor no momento $t - 1$, ou no $t - 2$, e assim em diante. Como um exemplo prático, podemos dizer que o número de casos de covid-19 em um município em uma determinada semana **depende** do número de casos da semana anterior e do número de casos de duas semanas atrás.

O termo p se refere à **ordem** do termo autorregressivo, de forma que um modelo $AR(1)$ se refere à dependência da série com seu valor imediatamente anterior e um $AR(2)$ se refere à dependência com os últimos dois valores.

Matematicamente, expressamos uma série com termo autorregressivo como:

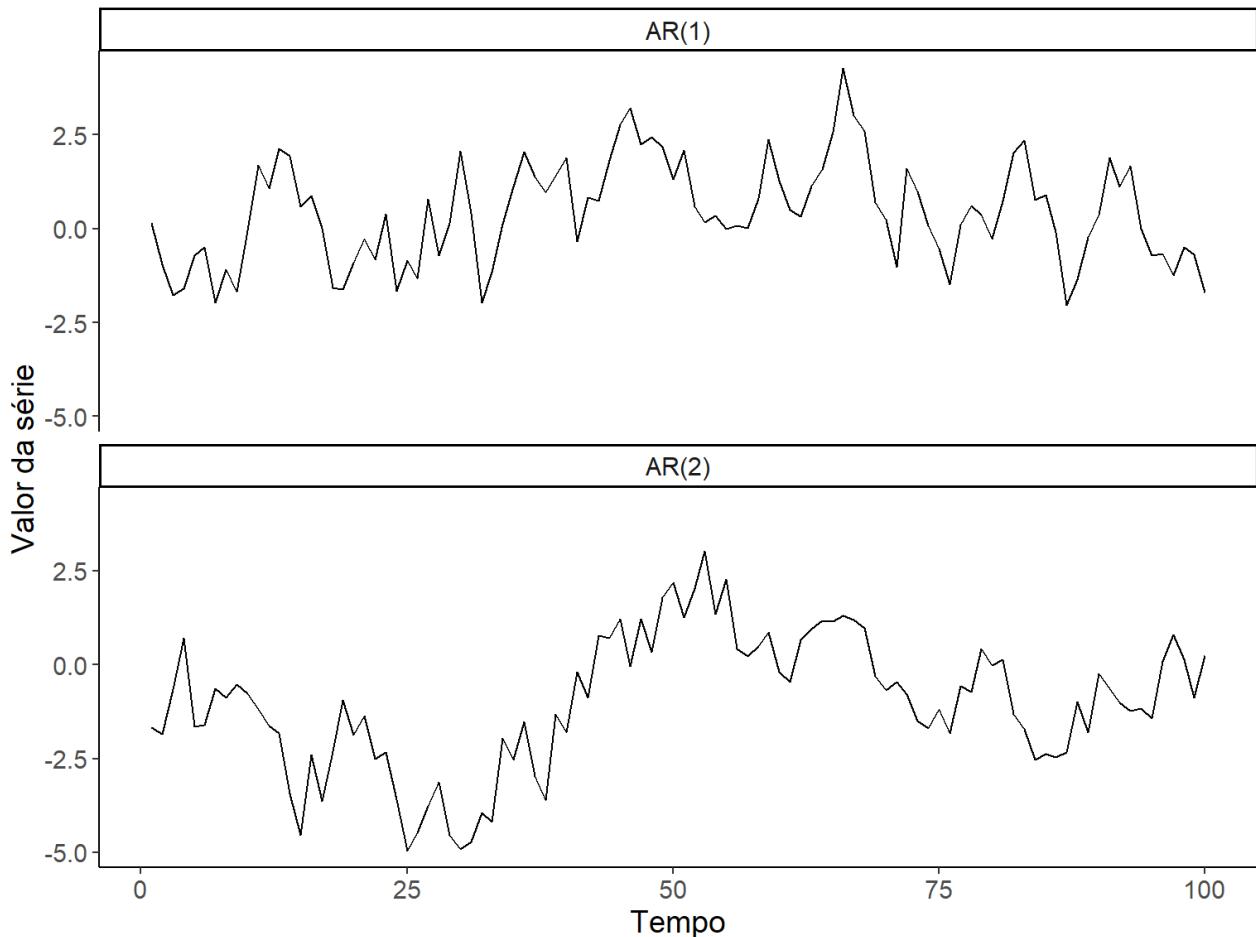
$$Z_t = c + \phi_1 Z_{t-1} + \phi_2 Z_{t-2} + \dots + \phi_p Z_{t-p} + a_t$$

Na qual:

- c é uma constante, representando a média do processo temporal;
- Os parâmetros ϕ são os coeficientes que indicam o **peso** da relação com cada valor no passado;
- Z_{t-1}, \dots, Z_{t-p} representam os valores da série nos momentos passados;
- a_t é o erro aleatório.

Agora vamos conferir na Figura 34 exemplos de duas séries com estruturas autorregressivas, uma $AR(1)$ e outra $AR(2)$:

Figura 34: Exemplos de séries AR(p).



Apenas visualmente não conseguimos identificar se uma série é $AR(1)$ ou $AR(2)$. Porém, no geral, espera-se que uma série $AR(1)$ seja mais suave, onde valores altos tendem a ser seguidos por valores altos. Em uma série $AR(1)$, espera-se mais oscilações devido ao impacto da dependência dos valores há dois tempos anteriores.



Não precisaremos identificar visualmente, para cada série temporal, se ela se trata de um $AR(1)$, $AR(2)$ ou outro $AR(p)$ qualquer. Atualmente, existem **técnicas de identificação automática** dessas estruturas do modelo que facilitam o processo de modelagem e sobre as quais veremos mais adiante.

Termo MA (médias móveis)

O termo de **médias móveis (MA)** no modelo ARIMA se refere à dependência de um determinado valor da série com os erros aleatórios **passados**. Ou seja, ao invés de tratar da relação com os valores passados como faz o termo **AR**, o termo **MA** se refere à relação com os erros (diferença entre os valores ajustados pelo modelo e os valores observados) no passado.



Apesar do nome, o termo de médias móveis (MA) **NÃO deve ser confundido** com a estratégia de suavização por médias móveis vista nos módulos anteriores! Aqui trata-se de outro contexto: é uma relação entre o erro da série em um determinado instante com os erros nos momentos passados.

Os modelos **MA** também possuem um parâmetro que define sua ordem: o parâmetro p , que define até que ponto da série essa dependência existe. Assim, matematicamente o definimos como:

$$Z_t = c + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \dots - \theta_p a_{t-q}$$

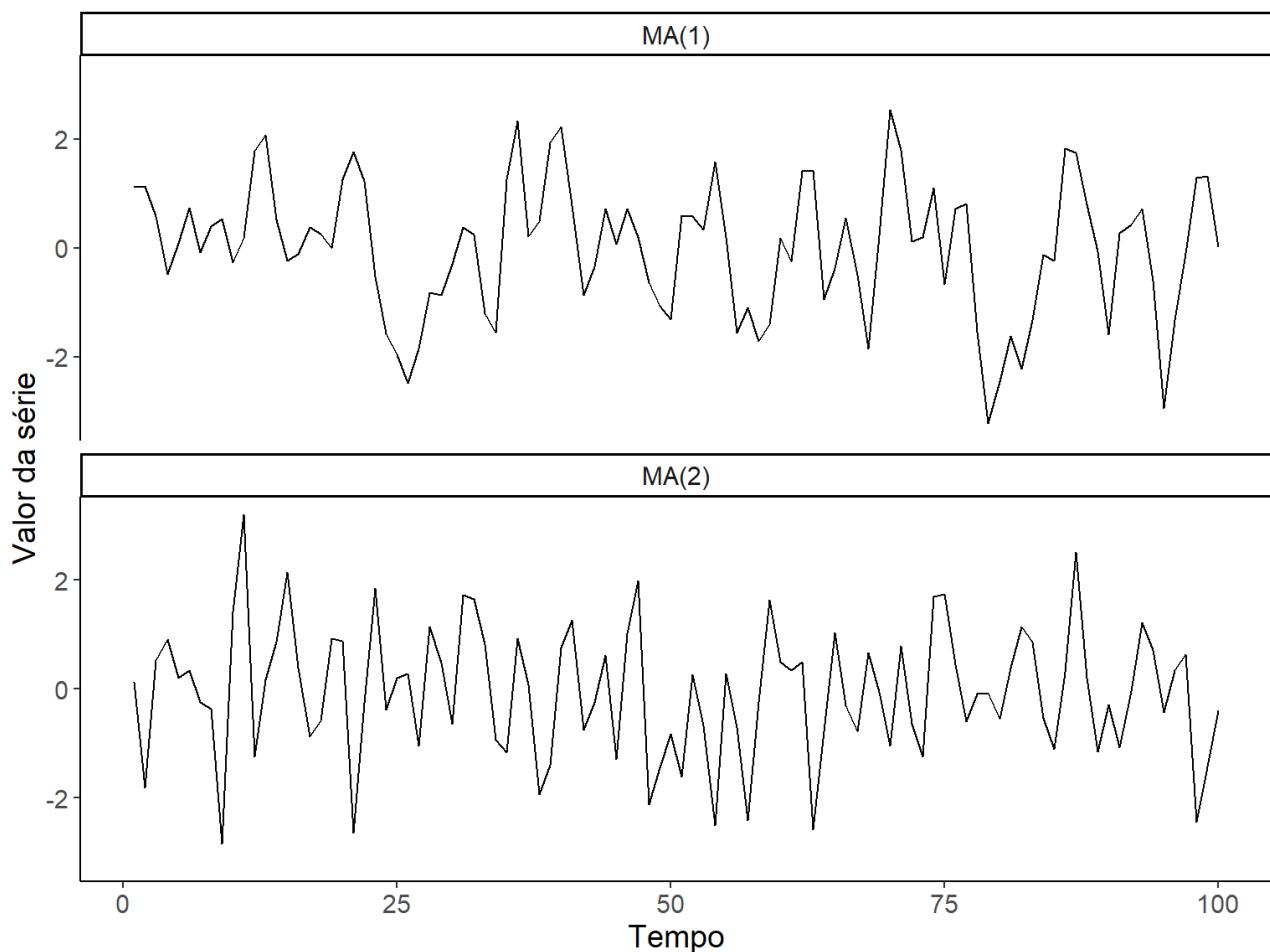
Agora, além do erro aleatório do momento t (a_t), introduzimos a dependência dos erros aleatórios anteriores: a_{t-1}, \dots, a_{t-q} . Os parâmetros θ representam o peso de cada erro passado, de forma similar aos ϕ nos modelos AR. No entanto, os termos são definidos por convenção com o sinal negativo (-). Vamos ver um exemplo prático paraclarear.

Exemplo prático

Imagine que temos uma série temporal do número de atendimentos em uma determinada unidade de saúde. Em um cenário de surto de diarreia na comunidade, por exemplo, pode ser que haja uma diferença entre o valor esperado de atendimentos e o valor observado, exatamente devido a esse aumento repentino e específico. Essa diferença seria capturada pelo erro da série: a diferença entre os valores ajustados e observados. Como no momento seguinte a esse aumento repentino a comunidade provavelmente ainda está sob os efeitos do mesmo surto, podemos esperar que haja também um aumento além do esperado de atendimentos no dia seguinte: ou seja, essa dependência que o valor atual da série possui em relação aos **erros** dos dias anteriores permanece e pode ser capturada através de um termo $MA(p)$.

A Figura 35 apresenta duas séries temporais exemplificando séries $MA(1)$ e $MA(2)$.

Figura 35: Exemplos de séries $MA(p)$.



Novamente, vemos que não é simples identificar a ordem da média móvel das séries de forma visual (e nem deve ser nosso objetivo!), mas podemos dizer que, em termos gerais, esperamos oscilações mais complexas em um modelo $MA(2)$ devido à dependência dos dois erros anteriores. Os exemplos são apenas ilustrativos e **não** é nossa tarefa identificar visualmente qual estrutura (**AR**, **MA**, ou as duas) está presente na série e qual a ordem dos termos p e q .

Termo I (diferenciação)

Agora que aprendemos sobre os termos autorregressivos e médias móveis, já temos material para modelar séries **estacionárias** em função de termos $AR(p)$ e termos $MA(q)$. Mas... somente séries estacionárias? Recapitulando dos módulos anteriores:



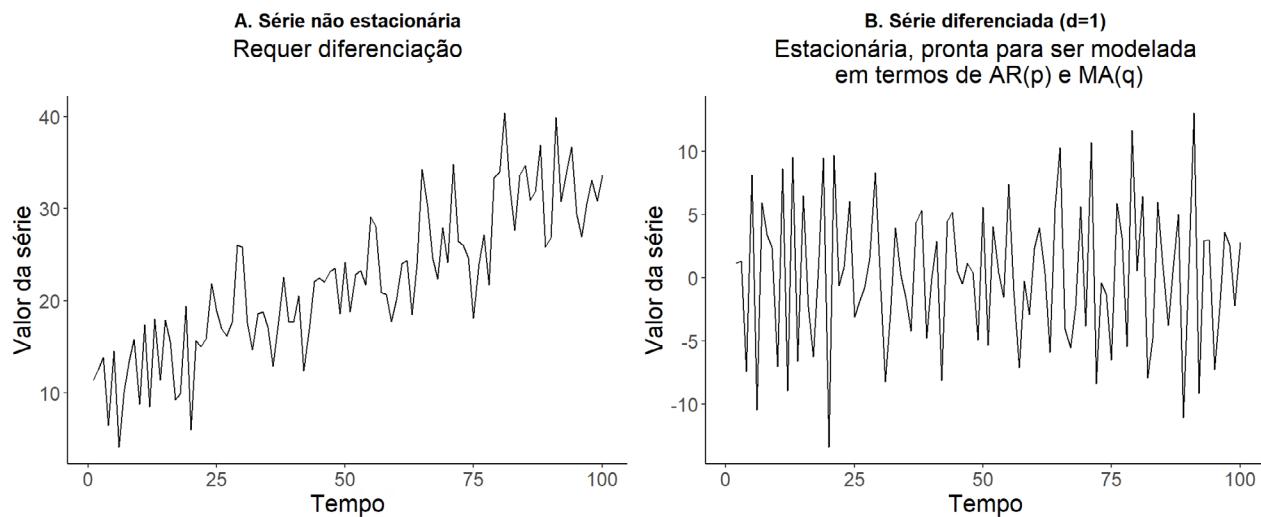
Uma série temporal é considerada **estacionária** quando seus valores flutuam aleatoriamente ao redor de uma média constante, com uma variância também constante. Ou seja, séries que possuem **tendência** já são consideradas **não estacionárias**, pois sua média varia ao longo do tempo.

Assim, séries não estacionárias não podem ser modeladas apenas com os termos $AR(p)$ e $MA(q)$, dado o requisito de estacionariedade. É aí que entra o terceiro componente do modelo ARIMA: a **diferenciação (I)**.

O termo $I(d)$ do modelo é responsável por aplicar, uma ou mais vezes, o procedimento de **diferenciação** sobre a série, com o intuito de torná-la estacionária caso não seja. Como vimos no primeiro módulo, a operação de diferenciação nada mais é que o cálculo das diferenças de uma observação da série para outra. A Figura 36 mostra uma série não estacionária antes e depois da diferenciação.



Figura 36: Diferenciação (B) de uma série não estacionária (A)



No termo de diferenciação $I(d)$, indica o grau de diferenciação que é o número de vezes em que a operação de diferenciação será realizada - geralmente uma ou duas vezes são suficientes. A diferenciação é essencial para adaptar as séries reais e torná-las aptas à modelagem.

Ajustando nosso primeiro modelo

Agora sim já sabemos tudo que é preciso para ajustar nosso primeiro modelo ARI-MA(p,d,q).

Para isso, vamos ler os dados de novos casos de tuberculose por mês no Rio de Janeiro-RJ, mostrados na Figura 37:

```
Library(tidyverse)
tb_rio <- read.csv2("https://raw.githubusercontent.com/joaohmoraes/dados-mod-temp/
refs/heads/main/csv/tb_rio_2007_2023.csv")
head(tb_rio) # primeiras observacoes
```

```
#>   ano_diagnostico mes_diagnostico   n
#> 1           2007       Jan  636
#> 2           2007       Fev  487
#> 3           2007       Mar  665
#> 4           2007       Abr  591
#> 5           2007       Mai  593
#> 6           2007       Jun  543
```

```
tail(tb_rio) # últimas observacoes
```

```
#>   ano_diagnostico mes_diagnostico   n
#> 199        2023       Jul  851
#> 200        2023       Ago  933
#> 201        2023       Set  791
#> 202        2023       Out  833
#> 203        2023       Nov  818
#> 204        2023       Dez  794
```

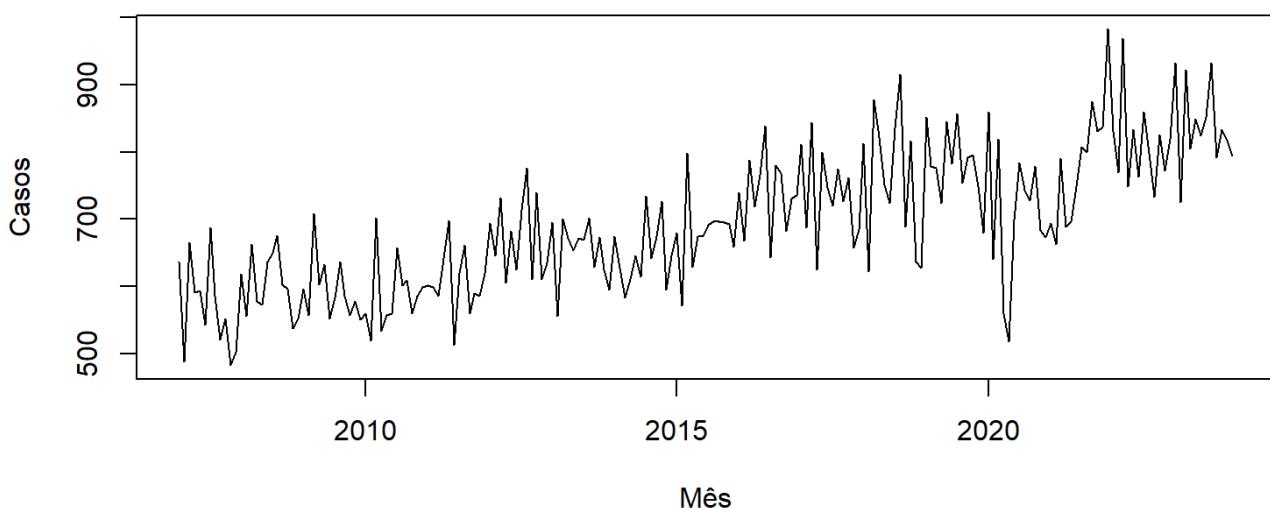
Vamos transformar o objeto em uma série temporal por meio da função `ts()`.

```
tb_rio_ts ← ts(tb_rio$n, # informação de casos: coluna "n"
                start = c(2007, 1), # início da série: Jan de 2007
                end = c(2023,12), # fim da série: Dez de 2023
                frequency = 12) # frequência da série: mensal
```

```
plot(tb_rio_ts,
      main = "Casos de TB por mês de diagnóstico\nRio de Janeiro-RJ, 2007-2023",
      ylab="Casos",
      xlab="Mês")
```

Figura 37: Série de casos de tuberculose, por mês de diagnóstico, no Rio de Janeiro-RJ. 2007-2023

**Casos de TB por mês de diagnóstico
Rio de Janeiro-RJ, 2007-2023**



Neste momento, fique à vontade para utilizar as técnicas vistas nos módulos anteriores para explorar e decompor a série, verificando a existência de componentes como tendência e sazonalidade. Esse processo ajudará a compreender os termos do modelo; se houver tendência, por exemplo, esperamos que o modelo ARIMA utilizado possua um grau de diferenciação $d > 0$.

Vamos ajustar nosso primeiro modelo ARIMA! Para isso, vamos utilizar a função `arima()`, nativa do R. Há funções de outros pacotes que também possibilitam o ajuste de modelos ARIMA no R, mas trataremos disso mais adiante.

Por meio do parâmetro `order`, especificamos os valores p (ordem da parte autorregressiva), d (grau de diferenciação) e q (ordem da média móvel) do modelo. Vamos ajustar um $ARIMA(1, 1, 0)$, ou seja, com um termo autorregressivo de ordem um, um grau de diferenciação e nenhum termo de média móvel:

```
arima_tb1 ← arima(
  tb_rio_ts,
  order = c( # Aqui no parâmetro order definimos a ordem dos termos
    1, # p = 1
    1, # d = 1
    0 # q = 0
  )
)
```

- A função `arima()` implementa um modelo de séries temporais ARIMA (Autoregressive Integrated Moving Average) para o objeto `tb_rio_ts`. O modelo é configurado com os seguintes parâmetros:

- `p = 1`: indica que o modelo inclui 1 termo autorregressivo.
- `d = 1`: indica que a série é diferenciada uma vez para torná-la estacionária.
- `q = 0`: indica que o modelo não inclui termos de médias móveis.

- O objeto `arima_tb1` armazena o modelo ajustado.

Pronto! Temos nosso primeiro modelo ARIMA. Vamos chamar o resumo do modelo:

```
arima_tb1 # visualizando um resumo do modelo
```

```
#>
#> Call:
#> arima(x = tb_rio_ts, order = c(1, 1, 0))
#>
#> Coefficients:
#>         ar1
#>     -0.6249
#> s.e.   0.0549
#>
#> sigma^2 estimated as 5658:  log likelihood = -1165.34,  aic = 2334.67
```

No resumo, vemos o coeficiente estimado (ϕ_1) para o termo autorregressivo em `Coefficients: ar1` e seu erro padrão (`s.e.`). Outro valor importante é o valor de `aic`, que expressa a qualidade de ajuste do modelo. Trataremos dele mais a frente, mas, por enquanto, vamos frisar a ideia de quanto menor o valor de `aic`, melhor o ajuste do modelo.

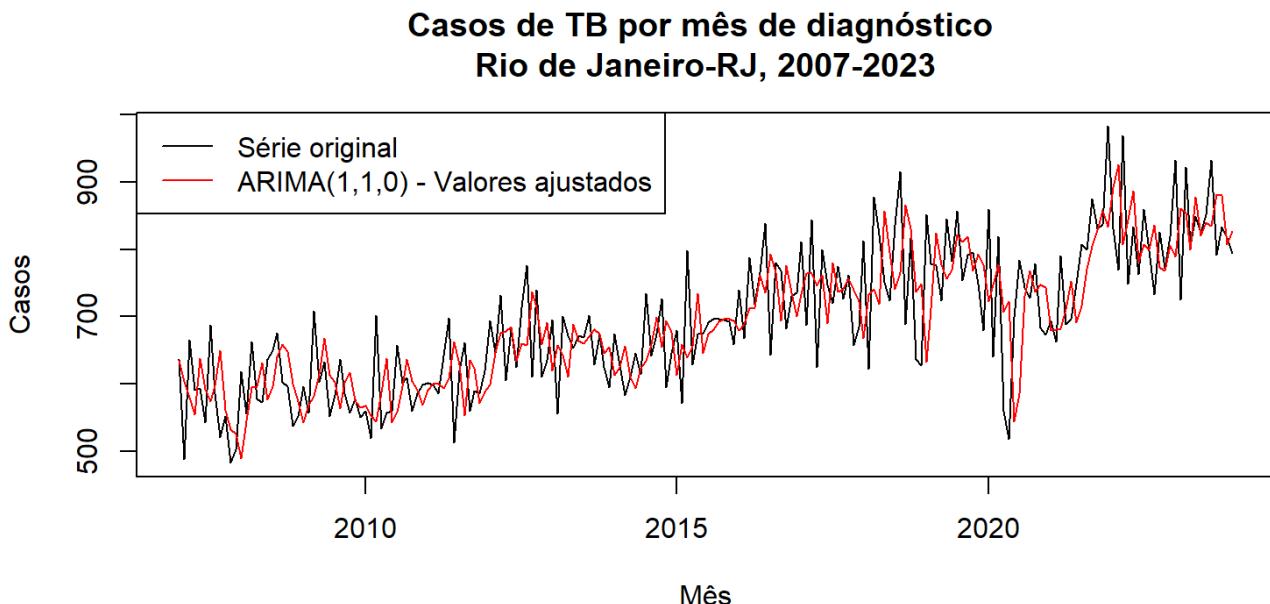
Agora vamos ver os valores ajustados pelo modelo comparado com a série original (Figura 38).

```
ajustados_tb1 ← tb_rio_ts - residuals(arima_tb1)

plot(tb_rio_ts,
      main = "Casos de TB por mês de diagnóstico \nRio de Janeiro-RJ, 2007-2023",
      ylab="Casos",
      xlab="Mês")
lines(ajustados_tb1, col = "red")
legend("topleft",
       legend = c("Série original", "ARIMA(1,1,0) - Valores ajustados"),
       col = c("black", "red"),
       lty = c(1, 1),
       cex = 1)
```



Figura 38: Série de TB e valores ajustados pelo primeiro modelo ARIMA



As linhas de comando realizam as seguintes ações:

- `ajustados_tb1 ← tb_rio_ts - residuals(arima_tb1)`: Calcula os valores ajustados do modelo ARIMA subtraindo os resíduos do modelo (`residuals(arima_tb1)`) da série original `tb_rio_ts`. O resultado é armazenado no objeto `ajustados_tb1`.
- `plot(tb_rio_ts, ...)`: Plota a série temporal original `tb_rio_ts`, configurando o título e os rótulos dos eixos (`ylab` e `xlab`).
- `lines(ajustados_tb1, col = "red")`: Adiciona uma linha ao gráfico representando os valores ajustados pelo modelo ARIMA, em vermelho.
- `legend(...)`: Insere uma legenda no canto superior esquerdo ("topleft") do gráfico. A legenda descreve as linhas do gráfico: a série original é exibida em preto, e os valores ajustados pelo modelo ARIMA (1,1,0) são exibidos em vermelho.

Resumindo, este script compara visualmente os dados observados (série original) com os valores ajustados pelo modelo ARIMA.

Porém, será que um modelo com um termo de médias móveis (MA) não seria mais adequado? Vamos experimentar, ajustando um *ARIMA(1, 1, 1)*:

```
arima_tb2 ← arima(
  tb_rio_ts,
  order = c( # Aqui no parâmetro order definimos a ordem dos termos
    1, # p = 1
    1, # d = 1
    1 # q = 1
  )
)
arima_tb2
```

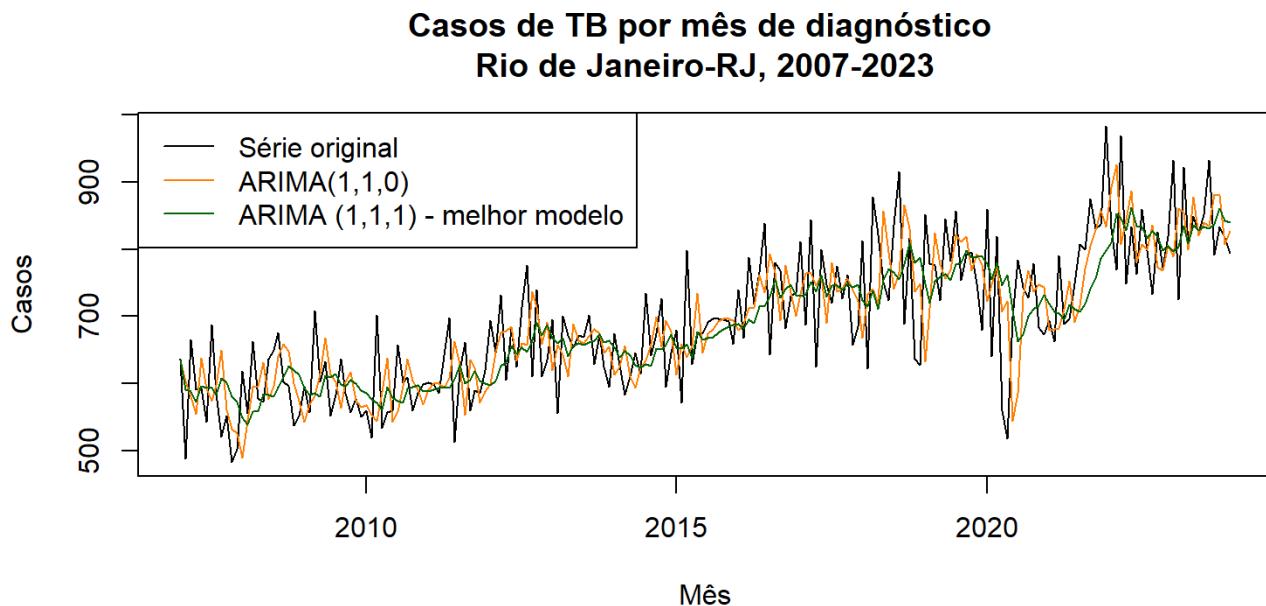
```
#>
#> Call:
#> arima(x = tb_rio_ts, order = c(1, 1, 1))
#>
#> Coefficients:
#>       ar1     ma1
#>     -0.2453  -0.7444
#> s.e.   0.0884   0.0718
#>
#> sigma^2 estimated as 4529:  log likelihood = -1143.09,  aic = 2292.18
```

- A função `arima()` ajusta um modelo ARIMA à série temporal `tb_rio_ts`, com os parâmetros `order = c(1, 1, 1)`. Isso configura o modelo para incluir:
 - `p = 1`: um termo autorregressivo,
 - `d = 1`: uma diferenciação para tornar a série estacionária,
 - `q = 1`: um termo de média móvel.
- O modelo ajustado é atribuído ao objeto `arima_tb2`.
- A chamada ao objeto `arima_tb2` exibe um resumo do modelo ajustado, mostrando os parâmetros estimados, erros padrão e medidas de ajuste.

Ao chamar o resumo para o modelo $ARIMA(1, 1, 1)$, vemos que o valor de `aic` exibido é **menor** que o visto para o primeiro modelo $ARIMA(1, 1, 0)$, indicando que o segundo modelo ajustado, incluindo um termo **MA**, é **melhor**.

Vamos visualizar os valores ajustados pelos dois modelos ($ARIMA(1, 1, 0)$ e $ARIMA(1, 1, 1)$) comparado com a série original na Figura 39.

Figura 39: Série de TB e valores ajustados pelos modelos ARIMA.



Ótimo! Mas será que esse é o melhor modelo dentre todas as opções? E quanto aos modelos com graus maiores de **AR**, como $AR(2)$ em diante? Ou ainda, modelos sem nenhum termo autoregressivo, considerando apenas médias móveis? E os diferentes valores para o termo de médias móveis (q) que ainda não testamos? São muitas possibilidades!

A escolha do modelo é um passo essencial para garantir a qualidade da análise de séries temporais. Um modelo mal ajustado pode levar a previsões distorcidas, influenciando decisões questionáveis e que foram baseadas nesses resultados. Ajustar diferentes modelos ARIMA e comparar seus desempenhos nos permite identificar aquele que melhor se adapta ao nosso objetivo, minimizando erros e proporcionando maior confiança nas interpretações.

Historicamente, o **processo de modelagem** com modelos ARIMA envolvia uma seleção manual dos parâmetros, com base na análise dos gráficos de autocorrelação (ACF) e autocorrelação parcial (PACF). Esse processo exigia um olhar criterioso do analista para determinar quais termos (**AR**, **I**, e **MA**) eram mais adequados para representar os dados. No entanto, com o avanço computacional, métodos automatizados passaram a permitir o **ajuste automático** de um grande número de combinações de parâmetros, comparando os modelos por meio de métricas estatísticas. Vamos conhecer um pouco desse processo.

Ajuste automático de modelos ARIMA

Uma forma de realizarmos esse ajuste automático, que compara diferentes modelos e retorna o melhor entre eles, é com o comando `auto.arima()` da biblioteca `forecast`. Vamos carregá-la. Se precisar, instale-a previamente utilizando a função `install.packages()`.

`library(forecast)`

Agora vamos aplicar o ajuste automático à nossa série de tuberculose:

```
arima_tb_auto ← auto.arima(  
  tb_rio_ts,  
  seasonal = FALSE,  
  # parâmetro que indica se modelos sazonais serão incluídos ou não  
  # (iremos ver sobre isso mais a frente)  
  approximation = FALSE,  
  # parâmetro que determina se aproximações serão utilizadas no  
  # processo de ajuste automático  
  trace = TRUE  
  # o parâmetro trace = T é utilizado apenas para exibir os modelos testados  
)
```

```
#>
#> ARIMA(2,1,2)           with drift      : 2296.159
#> ARIMA(0,1,0)           with drift      : 2434.442
#> ARIMA(1,1,0)           with drift      : 2336.676
#> ARIMA(0,1,1)           with drift      : 2296.235
#> ARIMA(0,1,0)           : 2432.415
#> ARIMA(1,1,2)           with drift      : 2294.289
#> ARIMA(0,1,2)           with drift      : 2294.223
#> ARIMA(0,1,3)           with drift      : Inf
#> ARIMA(1,1,1)           with drift      : 2292.633
#> ARIMA(2,1,1)           with drift      : Inf
#> ARIMA(2,1,0)           with drift      : 2306.826
#> ARIMA(1,1,1)           : 2292.302
#> ARIMA(0,1,1)           : 2297.803
#> ARIMA(1,1,0)           : 2334.732
#> ARIMA(2,1,1)           : 2293.651
#> ARIMA(1,1,2)           : 2294.159
#> ARIMA(0,1,2)           : 2293.957
#> ARIMA(2,1,0)           : 2305.009
#> ARIMA(2,1,2)           : 2295.484
#>
#> Best model: ARIMA(1,1,1)
```

- A função `auto.arima()` é utilizada para ajustar automaticamente um modelo ARIMA à série temporal `tb_rio_ts`. O ajuste é feito testando diferentes combinações de parâmetros até encontrar o melhor modelo com base em critérios de ajuste, como o AIC (Akaike Information Criterion).
- O parâmetro `seasonal = FALSE` especifica que não serão considerados componentes sazonais no modelo.
- O parâmetro `approximation = FALSE` define que não serão utilizadas aproximações durante o processo de ajuste automático, garantindo maior precisão no ajuste, embora possa aumentar o tempo de execução computacional.
- O parâmetro `trace = TRUE` exibe na saída do console os modelos que estão sendo testados, permitindo acompanhar o processo de seleção do modelo.
- O modelo ajustado é armazenado no objeto `arima_tb_auto`.

Pronto! Após comparar vários modelos, o processo de ajuste automático indicou que o modelo *ARIMA*(1, 1, 1), com $p = 1$ (um termo autorregressivo com dependência do valor anterior), $d = 1$ (diferenciação de primeira ordem, para remover a tendência) e $q = 1$ (um termo de média móvel, considerando o erro do valor anterior), é de fato o melhor modelo entre os que testamos.

No entanto, será que esse modelo é o **mais adequado** para nossa análise? Escolher um modelo estatisticamente ótimo não significa que ele seja o mais útil ou confiável para a aplicação prática. Precisamos verificar se ele realmente representa bem os dados e se suas previsões são coerentes com o contexto da série temporal. Para isso, vamos verificar a qualidade do ajuste e a análise dos resíduos, garantindo que o modelo escolhido seja não apenas o melhor em termos numéricos, mas também o mais adequado para descrever a série e gerar previsões confiáveis.

Mas, antes, vamos tratar de um outro ponto importante: e quando a série apresenta **sazonalidade**?

E a sazonalidade?

Como visto no módulo anterior, as séries temporais podem possuir um componente importante para seu entendimento e predição: a **sazonalidade**. Principalmente em vigilância em saúde, é comum lidar com situações que apresentem fluxos sazonais, como casos de influenza com picos no inverno. Essa componente ainda não seria capturada pelos termos do modelo ARIMA que vimos até agora: autorregressivo, diferenciação e médias móveis. Por isso, existe uma **extensão** do modelo ARIMA que chamamos de **SARIMA (Seasonal ARIMA)**.

A diferença do **SARIMA** é que o mesmo possui termos que se referem à dependência temporal **sazonal**: componente autorregressivo sazonal, diferenciação sazonal e médias móveis sazonais. Neste caso, esses termos não dizem respeito às observações imediatamente prévias (como casos nas últimas semanas ou meses), mas sim às observações do último período, na mesma época (ex.: casos no mesmo mês, mas no ano anterior).

Dessa forma, nosso modelo SARIMA possui a seguinte notação: $SARIMA(p, d, q)(P, D, Q)m$, em que, além dos parâmetros já conhecidos, temos:

- m , que se refere ao número de períodos da série. Ou seja, para séries mensais: $m = 12$ e semanais, $m = 52$;
- P , a ordem do termo autorregressivo sazonal;
- D , o número de diferenciações sazonais;
- Q , a ordem de médias móveis sazonais.

Ou seja, agora além dos termos não sazonais (de ordem p , d e q) que conhecíamos temos termos semelhantes, de ordens P , D e Q e que tratam da dependência sazonal presente nos dados.

Nosso primeiro modelo SARIMA

Agora vamos testar modelar os casos de Síndrome Respiratória Aguda Grave (SRAG) no Estado do Paraná.

```
# Carregando o arquivo csv
srag_pr_mes <- read.csv(file = "https://raw.githubusercontent.com/joaohmorais/dados-
mod-temp/main/csv/srag_PR_mes.csv")

# Vendo os dados
head(srag_pr_mes)
```

```
#>   ano mes   n
#> 1 2013   1   54
#> 2 2013   2   66
#> 3 2013   3  108
#> 4 2013   4  291
#> 5 2013   5  718
#> 6 2013   6 1037
```

Assim como feito anteriormente, vamos transformar os dados para a classe ts:

```
# Convertendo os dados para um objeto de classe ts
srag_pr_ts ← ts(
  data = srag_pr_mes$n, # variável que indica o número de casos
  start = c(2013, 1), # início da série - Janeiro de 2013
  frequency = 12 # frequência da série: mensal
)
```

- A função `ts()` está sendo utilizada para converter os dados do objeto `srag_pr_mes` em uma série temporal (`ts`), atribuindo-o ao objeto `srag_pr_ts`.
- `data = srag_pr_mes$n`: define a variável `n` (número de casos) como os valores da série.
- `start = c(2013, 1)`: especifica que a série começa em janeiro de 2013. `frequency = 12`: indica que a série tem periodicidade mensal (12 observações por ano).

Como já vimos nos exemplos dos módulos anteriores, sabemos que essa série possui forte componente sazonal. Vamos, de forma similar à realizada anteriormente, ajustar um modelo ARIMA automático para a série:

```
arima_srag_auto ← auto.arima(
  srag_pr_ts,
  approximation = F,
  trace = T
)
```

```
#>
#> ARIMA(2,0,2)(1,1,1)[12] with drift      : 654.3544
#> ARIMA(0,0,0)(0,1,0)[12] with drift      : 699.8139
#> ARIMA(1,0,0)(1,1,0)[12] with drift      : 647.817
#> ARIMA(0,0,1)(0,1,1)[12] with drift      : 661.9244
#> ARIMA(0,0,0)(0,1,0)[12]                  : 697.7201
#> ARIMA(1,0,0)(0,1,0)[12] with drift      : 654.6888
#> ARIMA(1,0,0)(1,1,1)[12] with drift      : 650.234
#> ARIMA(1,0,0)(0,1,1)[12] with drift      : 648.6853
#> ARIMA(0,0,0)(1,1,0)[12] with drift      : 692.905
#> ARIMA(2,0,0)(1,1,0)[12] with drift      : 648.6875
#> ARIMA(1,0,1)(1,1,0)[12] with drift      : 649.2379
#> ARIMA(0,0,1)(1,1,0)[12] with drift      : 664.935
#> ARIMA(2,0,1)(1,1,0)[12] with drift      : Inf
#> ARIMA(1,0,0)(1,1,0)[12]                  : 645.4911
#> ARIMA(1,0,0)(0,1,0)[12]                  : 652.4155
#> ARIMA(1,0,0)(1,1,1)[12]                  : 647.8075
#> ARIMA(1,0,0)(0,1,1)[12]                  : 646.3493
#> ARIMA(0,0,0)(1,1,0)[12]                  : 690.9641
#> ARIMA(2,0,0)(1,1,0)[12]                  : 646.2652
#> ARIMA(1,0,1)(1,1,0)[12]                  : 646.8091
#> ARIMA(0,0,1)(1,1,0)[12]                  : 662.8793
#> ARIMA(2,0,1)(1,1,0)[12]                  : 647.8045
#>
#> Best model: ARIMA(1,0,0)(1,1,0)[12]
```

- A função `auto.arima()` é utilizada para ajustar automaticamente um modelo ARIMA à série temporal `srag_pr_ts`.
- O parâmetro `approximation = F` desativa o uso de aproximações durante a seleção do modelo, garantindo maior precisão no ajuste.
- O parâmetro `trace = T` habilita a exibição de informações detalhadas no console sobre os diferentes modelos testados durante o processo de seleção automática.

O resultado é um modelo ARIMA otimizado, armazenado no objeto `arima_srag_auto`, baseado no critério de ajuste (como AIC ou BIC) para identificar a melhor combinação de termos autorregressivos (p), diferenciações (d) e de médias móveis (q).

Como esperado, vemos que o melhor modelo encontrado pela função possui termos sazonais: $ARIMA(1,0,0)(1,1,0)$.

Levar em conta a sazonalidade é essencial para capturar as estruturas presentes em diversas séries temporais que envolvem a rotina da vigilância. Agora, antes de tentar obter previsões com os modelos obtidos, precisamos realizar avaliar a **qualidade do ajuste**.

Módulo 6 - Diagnóstico do modelo e avaliação da qualidade do ajuste

Vimos anteriormente que há inúmeras combinações de parâmetros (p, d, q , ou P, D e Q) que podemos assumir no processo de modelagem. Assim, surge a necessidade de comparar modelos e verificar qual ou quais se ajustaram bem aos dados.

Uma das formas mais conhecidas de comparação de modelos (e que estávamos fazendo anteriormente, mas sem explicá-la a fundo) é por meio do **Critério de Informação de Akaike**, conhecido como **AIC**. Formalmente, essa métrica é definida como:

$$AIC = -2\log(L) + 2(p + q + k + 1)$$

Em que L é a verossimilhança do modelo (métrica que quantifica a bondade de ajuste do modelo), p é a ordem da parte autorregressiva, q é a ordem da parte de média móvel e k é definido como o **número de parâmetros** no modelo.

Ao comparar os modelos por meio do AIC, buscamos sempre o modelo que possui o **menor valor** de tal métrica. Ao olhar a fórmula, portanto, podemos notar que a métrica **penaliza** os modelos que possuem maior número de parâmetros (ou seja, que são mais complexos). Embora um modelo com mais parâmetros provavelmente se ajustará melhor aos dados, em alguns casos essa melhora não é relevante o suficiente quando comparada a um modelo mais simples. É esse o objetivo da penalização: optar pela simplicidade quando o ganho em ajuste não é relevante.

Quando ajustamos um modelo ARIMA no R vemos que ele nos fornece o AIC e, também, outras duas métricas. Voltemos a um exemplo da saída do modelo para a série temporal de SRAG:

arima_srág_auto

```
#> Series: srág_pr_ts
#> ARIMA(1,0,0)(1,1,0)[12]
#>
#> Coefficients:
#>         ar1      sar1
#>     0.7863  -0.4943
#> s.e.  0.0852   0.1408
#>
#> sigma^2 = 33762: log likelihood = -319.47
#> AIC=644.95  AICc=645.49  BIC=650.56
```

- O objeto `arima_srag_auto` armazena o modelo ARIMA ajustado automaticamente para os casos de SRAG no estado do Paraná.

As métricas de qualidade de ajuste aparecem na última linha da saída. Além do AIC, as outras métricas exibidas são o AICc (AIC corrigido) e o BIC (Critério de Informação Bayesiano). Todos seguem a mesma ideia: ao comparar modelos, buscamos diminuir esses valores para termos melhores ajustes. O AICc é similar ao AIC, porém com um termo de correção para casos em que a amostra de dados que temos é reduzida. Assim, penalizam-se modelos muito complexos em casos de pequenas amostras. O BIC também busca aplicar um peso maior a modelos com alta complexidade, tornando a penalização mais severa conforme a amostra aumenta.

Podemos, portanto, nos basear nessas métricas para comparar modelos ajustados à mesma série temporal. Vamos, por exemplo, comparar o ajuste do modelo para SRAG no Paraná com um outro modelo com outro conjunto de parâmetros, como um SARIMA(2,0,1)(1,0,1). Primeiro, vamos ajustá-lo:

```
arima_srag_2 ← Arima(y = srag_pr_ts, order = c(2,0,1), seasonal = c(1,0,1))
```

- A função `Arima()` é utilizada para ajustar um modelo ARIMA sazonal à série temporal `srag_pr_ts`.
 - `order = c(2, 0, 1)` define o modelo não sazonal com:
 - `p = 2`: dois termos autorregressivos.
 - `d = 0`: nenhuma diferenciação.
 - `q = 1`: um termo de médias móveis.
 - `seasonal = c(1, 0, 1)` define o componente sazonal do modelo com:
 - `P = 1`: um termo autorregressivo sazonal.
 - `D = 0`: nenhuma diferenciação sazonal.
 - `Q = 1`: um termo de médias móveis sazonais.
- O modelo ajustado é armazenado no objeto `arima_srag_2`.

Repare que agora usamos a função `Arima()`, com A maiúsculo, ao invés da `arima()` tradicional. A `Arima()` é a função para ajustes de modelos ARIMA do pacote `{forecast}`, e que possui mais opções, como a inclusão de termos sazonais.

Vamos chamar o modelo para exibir o seu resumo:

```
arima_srag_2
```

```
#> Series: srag_pr_ts
#> ARIMA(2,0,1)(1,0,1)[12] with non-zero mean
#>
#> Coefficients:
#>       ar1      ar2      ma1     sar1     sma1     mean
#>       1.4941 -0.7247 -0.4804  0.3610 -0.1979 357.5365
#> s.e.   0.1504   0.1255   0.2008  0.7201  0.7071  56.0912
#>
#> sigma^2 = 28749: log likelihood = -390.97
#> AIC=795.94   AICc=798.1   BIC=810.6
```

Vemos que segundo os três critérios, o modelo com ajuste automático obtido anteriormente possui melhor ajuste, pois possui menor valor tanto para o AIC quanto para AICc ou BIC.

Os processos de **ajuste automático** que utilizamos anteriormente, portanto, seguem essa lógica: realiza-se um conjunto de ajustes de diferentes modelos, e estes são comparados de acordo com uma essas métricas (por padrão, o AIC).

Análise de resíduos

Assim como outros tipos de modelos estatísticos, ao escolher um modelo ARIMA para os dados é necessário verificar se o modelo está **adequado**. Para isso, deve-se atentar se o modelo atende aos **pressupostos** estabelecidos. Neste caso, o pressuposto que temos é de que os resíduos do modelo sejam independentes e distribuídos aproximadamente por uma distribuição normal, de média zero e uma variância constante.



Resíduos de um modelo são valores que representam a **diferença** entre os valores ajustados pelo modelo e os valores observados da série temporal. Aqui, chamamos os resíduos de α_t , onde:

$$\alpha_t = z_t - \hat{z}_t$$

Ou seja, α_t representa a diferença de um valor observado na série no tempo t (z_t) e o que foi ajustado pelo modelo, para esse mesmo momento t (\hat{z}_t).

Assim, tem-se que o pressuposto em relação aos resíduos é:

$$\alpha_i \sim N(0, \sigma_\alpha^2)$$

Queremos que os resíduos (α_i) sigam uma distribuição normal centrada em zero, e com variância constante (σ_α^2). Ou seja, nós esperamos que o modelo **capture toda a estrutura de autocorrelação** dos dados. Ao olhar os resíduos, esperamos não haver nenhuma evidência de autocorrelação restante (chamada de *autocorrelação residual*).

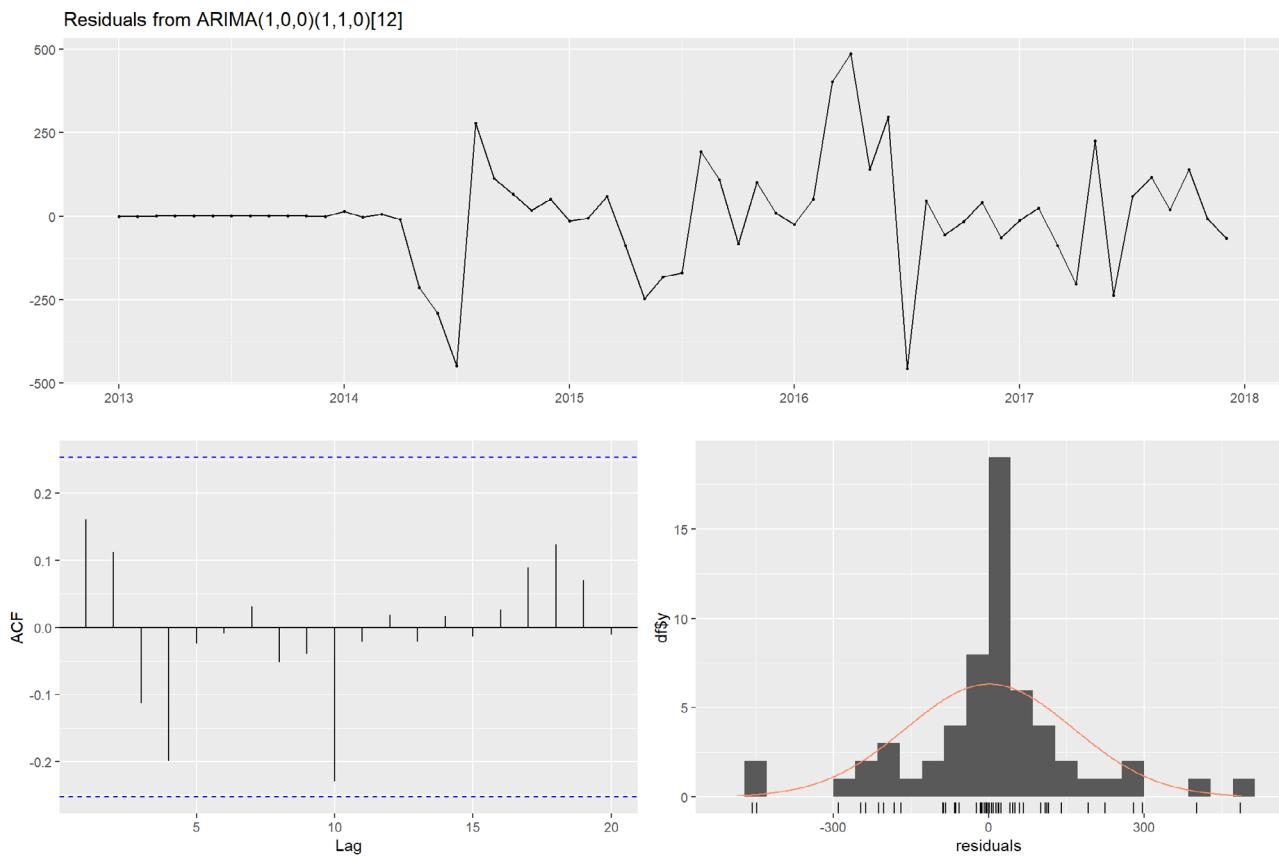
Para analisar esse pressuposto, geralmente seguimos as seguintes etapas:

- Fazemos um gráfico da série dos resíduos α_t , observando se não há uma tendência evidente (ou seja, se é estacionária), e se sua média está em torno de zero.
- Se a série dos resíduos for de fato estacionária, calcula-se funções de autocorrelação (ACF) e autocorrelação parcial (PACF).
- Se não há autocorrelação evidente, há indícios de que o modelo esteja adequadamente ajustado, pois não há nenhuma estrutura temporal “sobrando” nos dados. Caso haja, então é necessário rever e ajustar um outro modelo para a série trabalhada.

Vamos verificar se o modelo ajustado anteriormente para SRAG está adequado. Para isso, há uma função do pacote `{forecast}` que realiza esses passos diretamente, chamada `checkresiduals()`. Veja:

```
checkresiduals(arima_srág_auto)
```

Figura 40: Análise de resíduos de um modelo ARIMA para casos de SRAG no Paraná.



```
#>
#> Ljung-Box test
#>
#> data: Residuals from ARIMA(1,0,0)(1,1,0)[12]
#> Q* = 10.332, df = 10, p-value = 0.4119
#>
#> Model df: 2. Total lags used: 12
```

- A função `checkresiduals()` é usada para diagnosticar os resíduos de um modelo ajustado, neste caso, o modelo armazenado em `arima_srag_auto`. Ela verifica se os resíduos do modelo ARIMA seguem as suposições de independência e normalidade.
- Essa função gera três gráficos principais:
 - A série temporal dos resíduos: Permite verificar se há padrões visíveis nos resíduos ao longo do tempo. O ideal é que eles estejam distribuídos de forma aleatória ao redor de zero e que não apresentem tendências ou variações sistemáticas.
 - Um correlograma (ACF) dos resíduos: Mostra a autocorrelação dos resíduos em diferentes lags. O esperado é que os valores fiquem dentro das linhas pontilhadas. Ou seja, espera-se valores das autocorrelações de, no máximo, 0,25.
 - Um histograma mostrando a distribuição dos resíduos: Permite avaliar se a distribuição dos resíduos se aproxima de uma distribuição normal.
- Além desses gráficos, o teste de Ljung-Box também é aplicado para verificar se há autocorrelação significativa nos resíduos. Um p-valor não significativo (geralmente $p>0,05$) indica que não há autocorrelação residual relevante, reforçando a adequação do modelo.

O objetivo é verificar a adequação do modelo ajustado, garantindo que ele capturou toda a estrutura da série temporal sem deixar padrões significativos nos resíduos.

Caso ainda existam valores com autocorrelação superior a 0.25 ou -0.25, é indício de que o modelo não capturou toda a estrutura dos dados e não apresentou um bom ajuste. Dessa forma, um novo modelo deve ser ajustado. Isso pode envolver:

- Testar outros parâmetros;
- Aplicar transformações adicionais nos dados.

Agora vamos interpretar a Figura 40. Vê-se por meio da série dos resíduos (primeiro gráfico) e da distribuição dos resíduos em um histograma (terceiro gráfico) que sua maior concentração de valores está ao redor de zero. Sua variância também não parece aumentar nem diminuir de amplitude ao longo do tempo, o que indica uma variância estável.

Ao olhar para o segundo gráfico (ACF), que indica a autocorrelação dos resíduos, vê-se que não há autocorrelação relevante (acima ou abaixo da linha de referência) em nenhum lag. O p-valor para o teste de Ljung-Box também não foi significativo ao nível de 5%. Portanto, não há evidência de que tenha restado estrutura de autocorrelação temporal nos dados que não foi incorporada pelo modelo. Podemos dizer que o **modelo está, portanto, adequado**.

Contudo, se padrões diferentes do esperado fossem observados nos resíduos, ou se persistisse a presença de autocorrelação ao olhar o gráfico de ACF, um novo modelo que incorporasse essas estruturas seria necessário.

Com o modelo adequado, podemos seguir para realizar previsões com o modelo!

Modelos de previsão

Uma vez que ajustamos um modelo ARIMA e verificamos a qualidade de seu ajuste para uma série temporal, podemos utilizá-lo para realizar previsões para a série em um determinado número de “passos” (dias, semanas, meses) adiante. As previsões se dão com base nas estruturas modeladas (os termos de autocorrelação, de médias móveis, tendência e sazonalidade dos dados).

Vamos seguir com nosso exemplo de SRAG no Paraná. Anteriormente ajustamos um modelo ARIMA(1,0,0)(1,1,0) que pareceu adequado após a análise de resíduos, portanto vamos utilizá-lo para uma previsão de 12 meses à frente com a função `predict()`:

```
pred_srag ← predict(arima_srag_auto, n.ahead = 12, se.fit = T)
pred_srag ← data.frame(pred_srag)
pred_srag
```

```
#>          pred      se
#> 1  103.8709 183.7453
#> 2  149.4024 233.7406
#> 3  472.6541 259.8811
#> 4  777.6522 274.8005
#> 5  995.3029 283.6316
#> 6  1013.8055 288.9561
#> 7  588.6373 292.1993
#> 8  482.1176 294.1864
#> 9  345.9675 295.4081
#> 10 294.8000 296.1609
#> 11 260.4514 296.6254
#> 12 139.0994 296.9121
```

- A função `predict()` é utilizada para gerar previsões baseadas no modelo ARIMA ajustado armazenado em `arima_srag_auto`. O argumento `n.ahead = 12` indica que estão sendo previstas 12 observações à frente da série temporal original. O argumento `se.fit = T` também solicita os erros padrão das previsões.
- O objeto `pred_srag` armazena o tibble resultante, que contém as previsões e seus respectivos erros padrão. Quando `pred_srag` é chamado, ele exibe as previsões geradas pelo modelo ARIMA para os próximos 12 períodos e outras informações associadas.
- A função `data.frame()` transforma o objeto `pred_srag` em um data frame.

Obtemos um `data.frame` com duas variáveis: as estimativas pontuais para os 12 passos à frente, e o erro padrão associado a essas estimativas. A partir delas, podemos calcular o intervalo de confiança de 95% para a previsão:

```
pred_srag$inf ← pred_srag$pred - 1.96 * pred_srag$se
pred_srag$sup ← pred_srag$pred + 1.96 * pred_srag$se
```

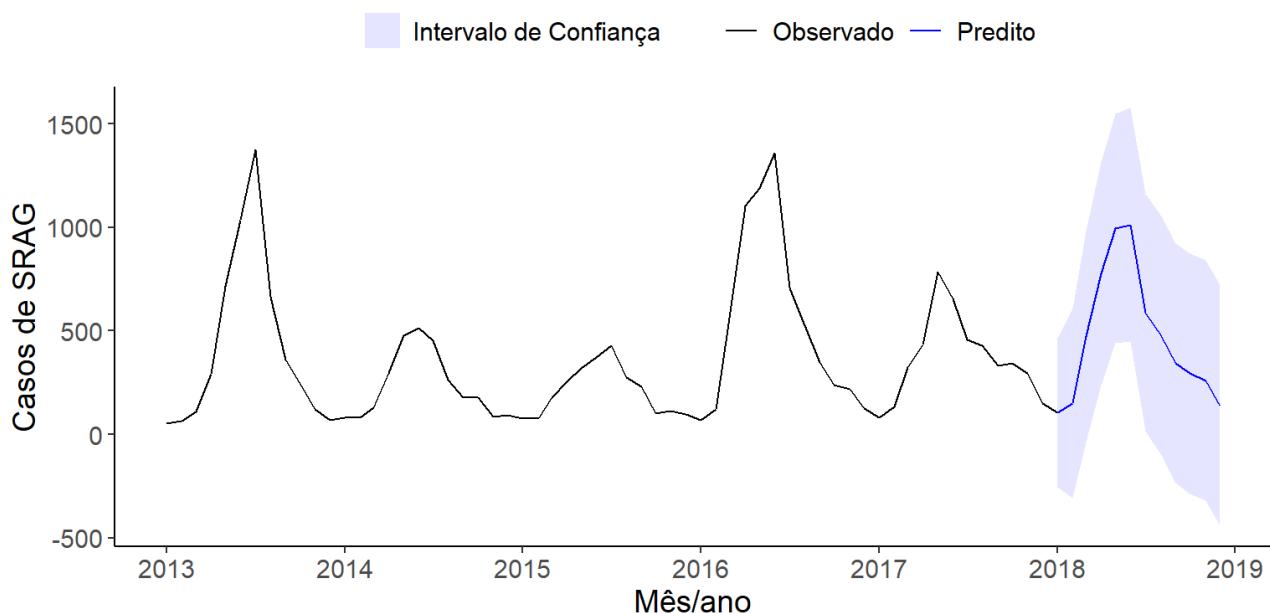
- Estas linhas de comando estão calculando os intervalos de confiança de 95% para as previsões armazenadas no objeto `pred_srag`.
 - `pred_srag$inf`: Calcula o limite inferior do intervalo de confiança, subtraindo 1,96 vezes o erro padrão (`se`) da previsão (`pred`).
 - `pred_srag$sup`: Calcula o limite superior do intervalo de confiança, somando 1,96 vezes o erro padrão (`se`) à previsão (`pred`).



Aqui, utilizamos os valores -1,96 e 1,96 pois são os valores dos percentis 2,5% e 97,5% de uma **distribuição normal padrão**. Ou seja, a partir deles delimitamos um intervalo de 95% de confiança para nossas estimativas.

Vejamos como a previsão se comporta no gráfico de série temporal na Figura 41.

Figura 41: Série temporal de SRAG no Paraná com previsão para um ano.



Vemos que o número esperado de casos para 2018 representa um padrão semelhante e aceitável em relação ao que foi visto nos anos anteriores, com um pico no inverno. É interessante perceber que o modelo estima um pico ainda maior que o ano anterior (2017). Contudo, os intervalos de confiança da previsão são amplos devida à alta incerteza associada.

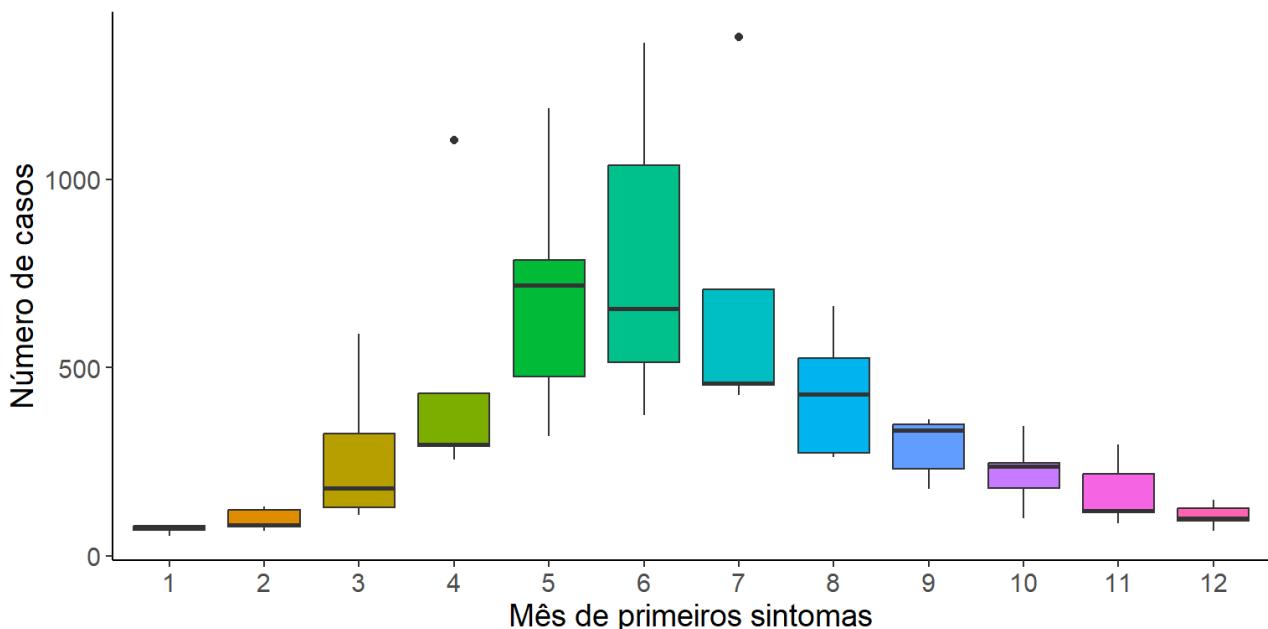
Módulo 7 - Modelando séries temporais com GAM

Agora vamos ver outra classe de modelos que também é capaz de capturar algumas estruturas de uma série temporal: os **Modelos Aditivos Generalizados (GAMs)**.

Em um modelo linear generalizado (GLM), como uma regressão linear ou uma regressão logística, estamos sujeitos a explicar relações lineares entre os dados. O **modelo aditivo generalizado (GAM)** é uma extensão dessa classe de modelos que permite a incorporação de termos não lineares, chamados de **funções suaves**.

Por exemplo, em nossa série de SRAG no Paraná, vamos observar o efeito da variável mês em relação ao número de casos na Figura 42.

**Figura 42: Distribuição do número de casos de SRAG
no PR por mês de primeiros sintomas, 2013-2017.**



Essa é, como vimos nos módulos anteriores, uma boa técnica para observar a sazonalidade dos dados. Vemos um aumento de casos nos meses intermediários do ano, principalmente nos meses de outono/inverno. Contudo, percebemos que a relação entre casos e meses **não é linear**, pois apresenta um aumento e depois uma diminuição, conforme avançamos os meses.

Mesmo assim, vamos tentar modelar essa relação por meio de um modelo linear generalizado (GLM), o modelo de Poisson:

```
mod ← glm(n ~ mes, data=srag_pr_mes, family = "poisson")
summary(mod)
```

```
#>
#> Call:
#> glm(formula = n ~ mes, family = "poisson", data = srag_pr_mes)
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept) 5.951726  0.014318 415.686 < 2e-16 ***
#> mes         -0.010897  0.001976 -5.514 3.51e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for poisson family taken to be 1)
#>
#> Null deviance: 14492  on 59  degrees of freedom
#> Residual deviance: 14461  on 58  degrees of freedom
#> AIC: 14907
#>
#> Number of Fisher Scoring iterations: 5
```

- A função `glm()` ajusta um modelo de regressão linear generalizado, neste caso em particular será levado em consideração os dados com distribuição de Poisson.
- O modelo tem a variável resposta `n` (o número de casos de SRAG) sendo explicada pela variável preditora `mes` (mês de primeiros sintomas) no conjunto de dados `srag_pr_mes`. O argumento `family = "poisson"` indica que o modelo deve usar a distribuição de Poisson, adequada para contagens.
- A função `summary(mod)` exibe um resumo do modelo ajustado, incluindo coeficientes estimados, significância estatística, desvio residual e outras métricas relevantes para avaliar o ajuste do modelo.

Vemos que o mês de início de sintomas teve uma associação inversa estatisticamente significativa com o número de casos. Porém, vamos verificar os valores ajustados pelo modelo na Figura 43:

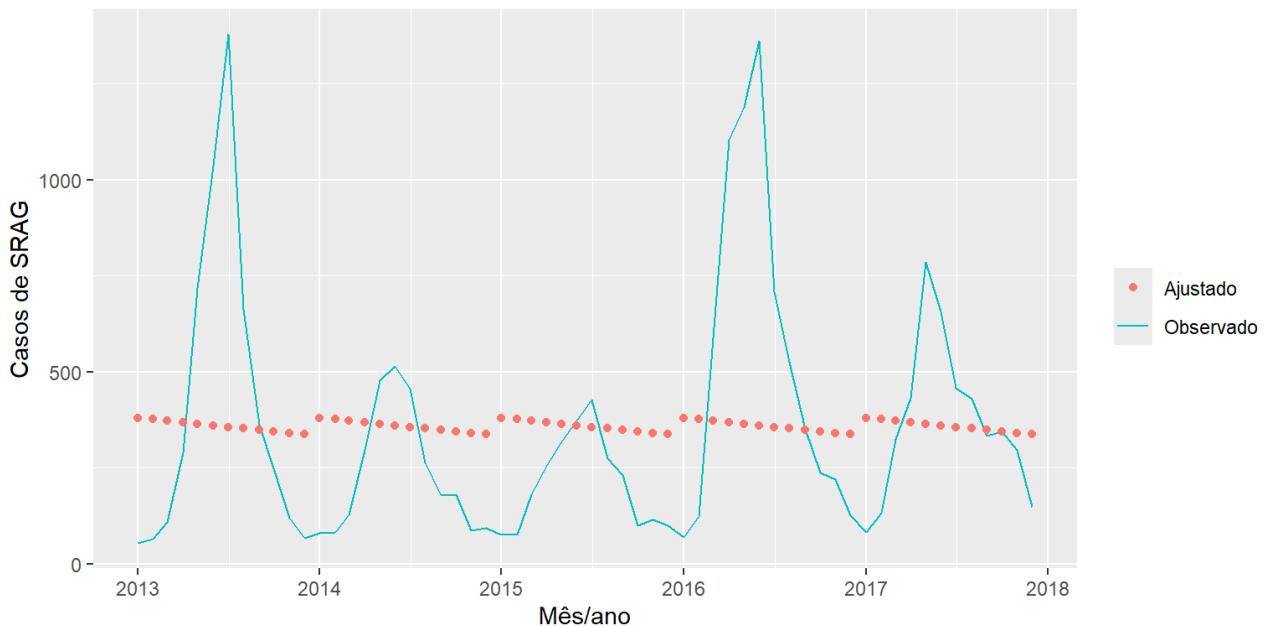
```

srag_pr_mes <- srag_pr_mes %>%
  mutate(dt_mes = ymd(paste0(ano, " - ", mes, "-01")))

srag_pr_mes$est_poisson <- fitted.values(mod)

ggplot(srag_pr_mes, aes(x=dt_mes)) +
  geom_line(aes(y=n, color="Observado")) +
  geom_point(aes(y=est_poisson, color = "Ajustado")) +
  scale_x_date(date_breaks = "1 year",
               date_labels = "%Y") +
  labs(x="Mês/ano", y = "Casos de SRAG", color = "")
```

Figura 43: Casos de SRAG no Paraná observados versus ajustados por um modelo linear generalizado.



O script cria um gráfico utilizando a biblioteca `ggplot2` para comparar os valores observados de casos de Síndrome Respiratória Aguda Grave (SRAG) ao longo do tempo com os valores ajustados por um modelo linear generalizado:

- `srag_pr_mes$est_poisson ← fitted.values(mod)`: Adiciona uma nova coluna ao dataset `srag_pr_mes` chamada `est_poisson`, que armazena os valores ajustados pelo modelo linear generalizado armazenado em `mod`.
- `ggplot(srag_pr_mes, aes(x=dt_mes))`: Inicia a criação de um gráfico com `ggplot2`, definindo o eixo x como `dt_mes`, que representa as datas (mês/ano).
- `geom_line(aes(y=n, color="Observado"))`: Adiciona uma linha ao gráfico para os valores observados (variável `n`), atribuindo a cor "Observado".
- `geom_point(aes(y=est_poisson, color="Ajustado"))`: Adiciona pontos representando os valores ajustados (variável `est_poisson`), atribuindo a cor "Ajustado".
- `scale_x_date()`: Define que o eixo x será formatado com intervalos anuais (`date_breaks = "1 year"`) e rótulos no formato de ano (`date_labels = "%Y"`).
- `labs(x="Mês/ano", y="Casos de SRAG", color="")`: Adiciona rótulos aos eixos e remove o título da legenda de cores.

Veja que o modelo linear generalizado **não foi capaz** de capturar o efeito dos meses sobre os dados, justamente porque esse efeito é **não linear**, como vimos no gráfico de boxplot anterior.

É aí que entra o modelo aditivo generalizado (GAM): ele permite que incluamos termos não lineares por meio de funções suaves cuja complexidade conseguimos controlar. Isso é útil para modelar relações como essa (entre mês e número de casos - sazonalidade), para modelar a tendência nos dados (seja ela linear ou não), e modelar o efeito de outras variáveis sobre nosso desfecho de interesse.

Vamos realizar o mesmo exemplo anterior com os dados de SRAG no Paraná, mas agora com um modelo GAM. Para isso, vamos utilizar a função `gam()` da biblioteca `mgcv`. Instale-a primeiro, se for preciso, utilizando o código `install.packages("mgcv")`. Agora, em vez de especificar o modelo como `n ~ mes`, utilizamos `n ~ s(mes)` para indicar que uma função suave não linear será aplicada à variável.

Library(`mgcv`)

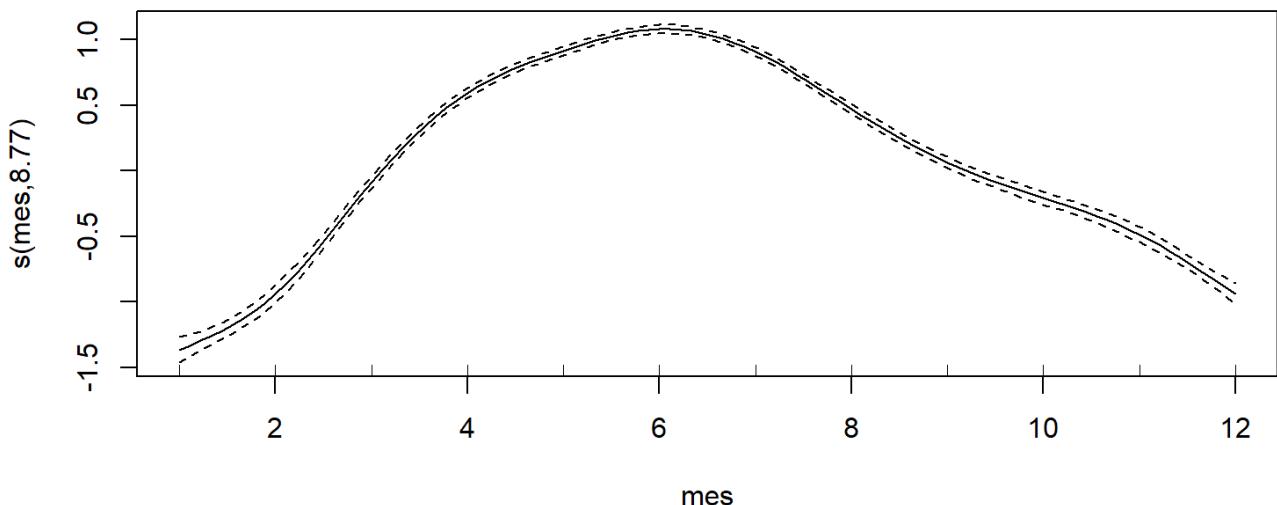
```
mod_gam ← gam(n ~ s(mes), data=srag_pr_mes, family = "poisson")
```

- A biblioteca `mgcv` é carregada para possibilitar o uso de Modelos Aditivos Generalizados (GAM), que permitem modelar relações não lineares entre variáveis.
- A função `gam()` ajusta um modelo GAM, onde:
 - A variável resposta `n` (número de casos de SRAG) é modelada como uma função suave do termo `mes` (mês), representado por `s(mes)`. Isso permite capturar relações não lineares entre o mês e os casos.
 - O parâmetro `family = "poisson"` indica que os dados seguem uma distribuição de Poisson, comumente utilizada para modelar contagens.

Pronto! Agora vamos plotar o resultado da relação entre mês e casos de SRAG na Figura 44.

```
plot(mod_gam, select = 1)
```

Figura 44: Relação entre mês de primeiros sintomas e casos de SRAG no Paraná estimada com um modelo GAM.



- A função `plot()` gera um gráfico para visualizar a relação não linear estimada entre o mês (`mes`) e os casos de SRAG (`n`). O argumento `select = 1` garante que a suavização do primeiro termo seja exibida no gráfico.

Veja só! Agora o modelo, ainda que simples, conseguiu capturar melhor a relação entre o mês de primeiros sintomas e o número de casos de SRAG. Percebe-se que o aumento de casos, com ápice no mês 6, assim como o menor número no começo e final do ano, foram capturados pelo modelo.

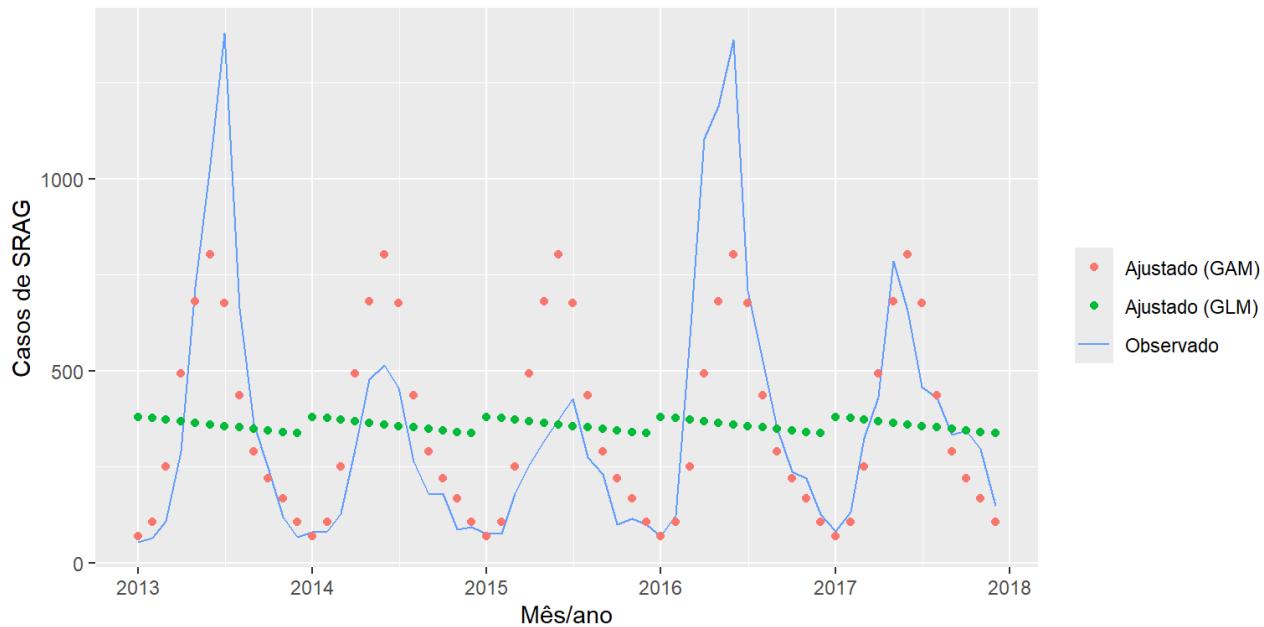
Vamos na Figura 45 comparar os valores ajustados pelos modelos GLM e GAM com a série observada:

```

srag_pr_mes$est_gam <- fitted.values(mod_gam)

ggplot(srag_pr_mes, aes(x=dt_mes)) +
  geom_line(aes(y=n, color="Observado")) +
  geom_point(aes(y=est_poisson, color = "Ajustado (GLM)")) +
  geom_point(aes(y=est_gam, color = "Ajustado (GAM)")) +
  scale_x_date(date_breaks = "1 year",
               date_labels = "%Y") +
  labs(x="Mês/ano", y = "Casos de SRAG", color = "")
```

Figura 45. Casos de SRAG no Paraná observados comparado com valores ajustados por modelo GAM e por modelo GLM.



Este trecho de código está criando um gráfico de linhas e pontos utilizando o pacote `ggplot2` no R para comparar dados observados e ajustados em um conjunto de dados sobre casos de Síndrome Respiratória Aguda Grave (SRAG), contidos no objeto `srag_pr_mes`. A seguir, uma explicação dos elementos:

- `srag_pr_mes$est_gam ← fitted.values(mod_gam)`: Adiciona uma coluna chamada `est_gam` ao conjunto de dados `srag_pr_mes`, contendo os valores ajustados do modelo GAM armazenado em `mod_gam`.
- `ggplot(srag_pr_mes, aes(x=dt_mes))`: Inicializa um gráfico com o eixo x representando os meses (`dt_mes`).
- `geom_line(aes(y=n, color="Observado"))`: Adiciona uma linha ao gráfico representando os valores observados de SRAG (`n`), com a cor definida como "Observado".
- `geom_point(aes(y=est_poisson, color = "Ajustado (GLM)"))`: Adiciona pontos ao gráfico para os valores ajustados pelo modelo GLM, presentes na coluna `est_poisson`. A cor é definida como "Ajustado (GLM)".
- `geom_point(aes(y=est_gam, color = "Ajustado (GAM)"))`: Adiciona pontos ao gráfico para os valores ajustados pelo modelo GAM, presentes na coluna `est_gam`. A cor é definida como "Ajustado (GAM)".
- `scale_x_date(date_breaks = "1 year", date_labels = "%Y")`: Configura o eixo x para exibir marcas de data com intervalos de 1 ano, formatadas para mostrar apenas o ano.
- `labs(x="Mês/ano", y = "Casos de SRAG", color = "")`: Personaliza os rótulos dos eixos `x` e `y`, além da legenda de cores (color).

Resumindo, o gráfico resultante mostra a evolução dos casos de SRAG ao longo do tempo, comparando os dados observados e os valores ajustados pelos modelos GLM e GAM.

Agora vemos claramente que os valores ajustados do modelo GAM se aproximam muito mais dos valores observados do que os valores ajustados do modelo GLM. Note que esse modelo GAM ainda é um modelo muito simples, sem incorporação da tendência ou qualquer outro componente, apenas para exemplificar a incorporação de **termos não-lineares** a um modelo. Vamos prosseguir para uma definição mais completa de um modelo GAM.

Definição de um modelo GAM

Um modelo GAM é, portanto, uma extensão do modelo linear generalizado (GLM) que permite a inclusão destas **funções suaves não-lineares** para capturar as relações presentes nos dados. Geralmente essas funções são funções de alisamento, como kernel, loess e splines. Neste curso não é nosso objetivo nos aprofundarmos sobre esse tipo de modelo, mas sim apenas contextualizar seu uso dentro do campo de séries temporais. Formalizando, para uma série temporal simples, temos:

$$\eta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_j x_j + f_1(x_{j+1}) + f_2(x_{j+2}) + \dots$$



Aqui escrevemos a equação em termos de η , que chamamos de preditor linear do modelo. Fazemos isso pois, a depender do tipo de modelo (linear, logístico, Poisson) haverá uma função de ligação entre o parâmetro modelado e a equação. No caso de modelos Poisson, essa função é o log, portanto: $\eta = \log(\lambda)$, onde λ é a média da distribuição que costumamente se modela.

Vê-se que agora temos j termos (referentes às variáveis x_1, \dots, x_j) modelados de forma linear, como costumamos fazer em um GLM; seguidos de outros termos (referentes a x_{j+1}, x_{j+2}, \dots) modelados através de funções f_1, f_2, \dots que introduzem a não-linearidade à equação.

Essas funções são representadas no R através da função `s()`, que possui três parâmetros principais:

- `s(var)`: o primeiro parâmetro a se especificar é sobre **qual(is) variável(is)** será aplicada a função de suavização. No nosso exemplo anterior essa variável era o mês de primeiros sintomas: `s(mes)`;
- `s(var, bs="tp")`: `bs` informa o **tipo de função suave** a ser usada. Mais detalhes sobre essas funções serão dados abaixo;
- `s(var, bs="tp", k=6)`: `k` é o número de nós, ou a **dimensão** utilizada para se ajustar o termo. Quanto menos nós, **mais simples e parcimoniosa** nossa função; quanto mais nós, **mais rigidamente** ela se ajustará aos dados.

Tipos de Splines

Há diversas implementações de funções de suavização, sendo as mais comuns:

- thin plate (`bs="tp"`),
- cubic regression (`bs="cr"`),
- cyclic cubic regression (`bs="cc"`),
- B-splines (`bs="bs"`), e
- P-splines, ou splines penalizadas (`bs="ps"`).

De uma forma simplificada:

Thin plate – Baseia-se na obtenção do menor erro quadrático, e penaliza a spline conforme sua curvatura, o que evita complexidade nas curvas e o sobreajuste. Não precisa de muitos parâmetros, mas pode ser mais computacionalmente custosa.

Cubic regression spline – são baseadas em polinômios de grau 3 ajustados aos intervalos dos dados definidos pelos nós distribuídos regularmente em toda a amplitude dos dados. Tende a gerar transições suaves e sem curvas bruscas. No entanto, não se baseia em um estimador ótimo com as thin plate splines.

Cyclic cubic regression spline – semelhantes à spline anterior, há agora a restrição de ter o mesmo valor e mesmas derivadas no início e final, garantindo a ciclicidade. É interessante para séries com termos sazonais.

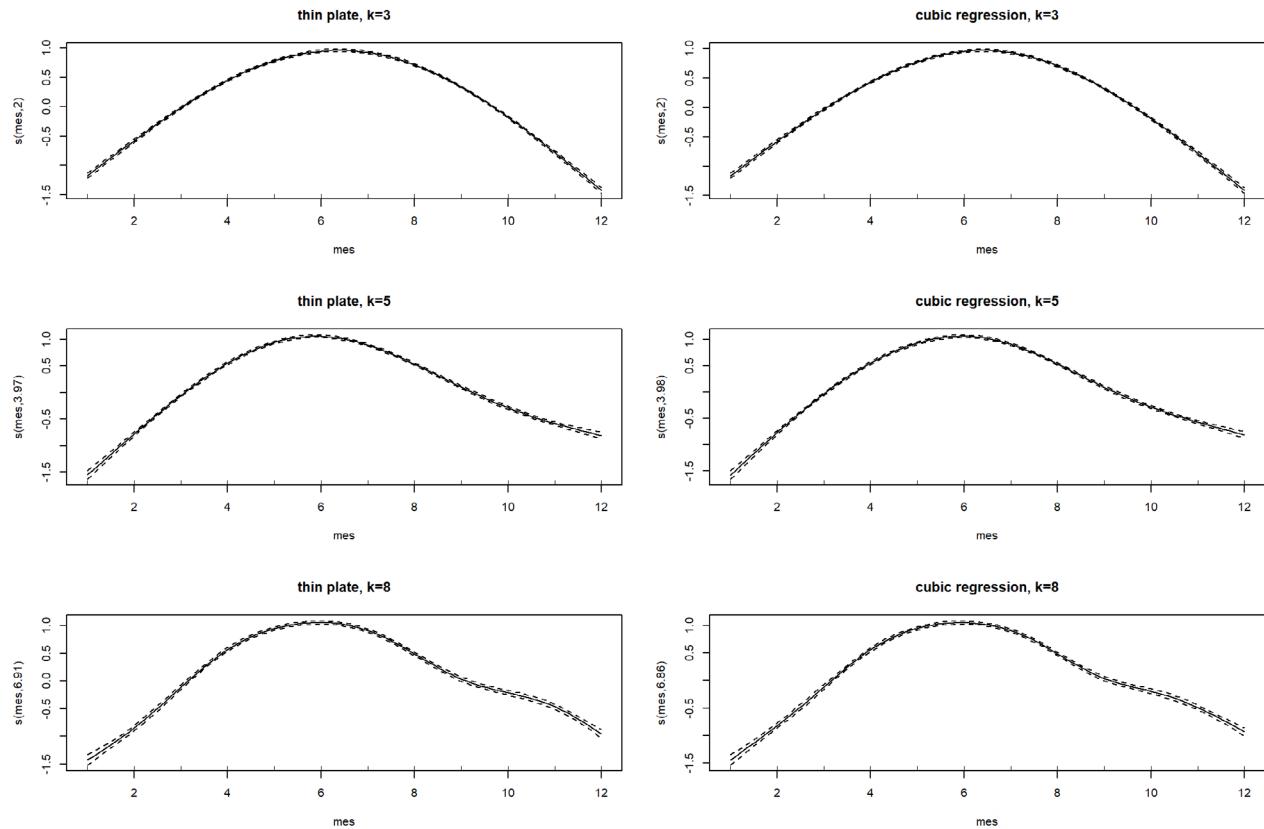
B-splines: Essas splines baseiam-se em funções-base locais que agem cada uma sobre uma parcela dos dados. Em cada segmento entre os nós especificados são ajustadas funções simples (como as funções cúbicas, mas não necessariamente). Geralmente possuem ajustes eficientes, e menos sensíveis a outliers.

P-splines – Baseiam-se na B-spline, porém aplicam uma penalização diferencial entre os termos adjacentes para controlar possíveis variações bruscas e suavizar a continuidade dos termos. Ideais para dados com variação complexa.

Mais detalhes sobre tais funções e modelos aditivos generalizados podem ser encontrados em [Ross \(2019\)](#).

Voltando ao exemplo das splines para o mês de primeiros sintomas na série de SRAG, vamos comparar as diferentes funções (`bs`) e dimensões (`k`) na Figura 46:

Figura 46: Relação entre mês de primeiros sintomas e casos de SRAG no Paraná estimada com diferentes funções e dimensões de modelo GAM.



Módulo 8 - GAM em Séries Temporais

Bom, já vimos como incorporar um termo que represente a sazonalidade da série de SRAG em relação aos meses do ano (`s(mes)`). Como vimos anteriormente, outro componente frequentemente presente nas séries temporais é a tendência. O efeito da tendência pode ser incorporado por um termo que capture a variação da média da série temporal em um médio/longo prazo. Note que o quanto médio ou longo será esse efeito é determinado pela quantidade de nós (o parâmetro `k` da função) que vimos anteriormente. Vamos testar a entrada do termo de tendência, variando alguns valores de `k`.

Voltemos aos dados de SRAG no Paraná e verifiquemos as primeiras 15 observações do banco:

ano	mes	n	dt_mes
2013	1	54	2013-01-01
2013	2	66	2013-02-01
2013	3	108	2013-03-01
2013	4	291	2013-04-01
2013	5	718	2013-05-01

Vemos que nosso banco de dados tem a variável mês (sobre a qual incluímos o termo de sazonalidade) e a variável ano. Para inserir o efeito da tendência, vamos criar uma variável adicional que inclua informações do mês e ano simultaneamente.

```
srug_pr_mes <- srug_pr_mes %>%
  mutate(mes_ano = ano + (mes-1)/12)
```

Os comandos estão realizando as seguintes ações:

- `srag_pr_mes %>%`: Utiliza o operador pipe (`%>%`) do pacote `dplyr` para aplicar transformações no objeto `srag_pr_mes`.
- `mutate(mes_ano = ano + (mes-1)/12)`: Também do pacote `dplyr`, cria uma nova coluna chamada `mes_ano` no conjunto de dados `srag_pr_mes`. Essa nova coluna representa o mês/ano em formato decimal, calculado como:

O ano (`ano`) mais a fração do mês (`mes-1`) dividido por 12. Isso transforma o valor do mês em uma fração do ano.

Exemplo de cálculo:

- Para `ano = 2023` e `mes = 4`:
- Cálculo: $2023 + (4-1)/12 = 2023 + 3/12 = 2023.25$
- Resultado: O valor 2023.25 representa abril de 2023 em formato decimal.

Primeiras 5 observações do banco:

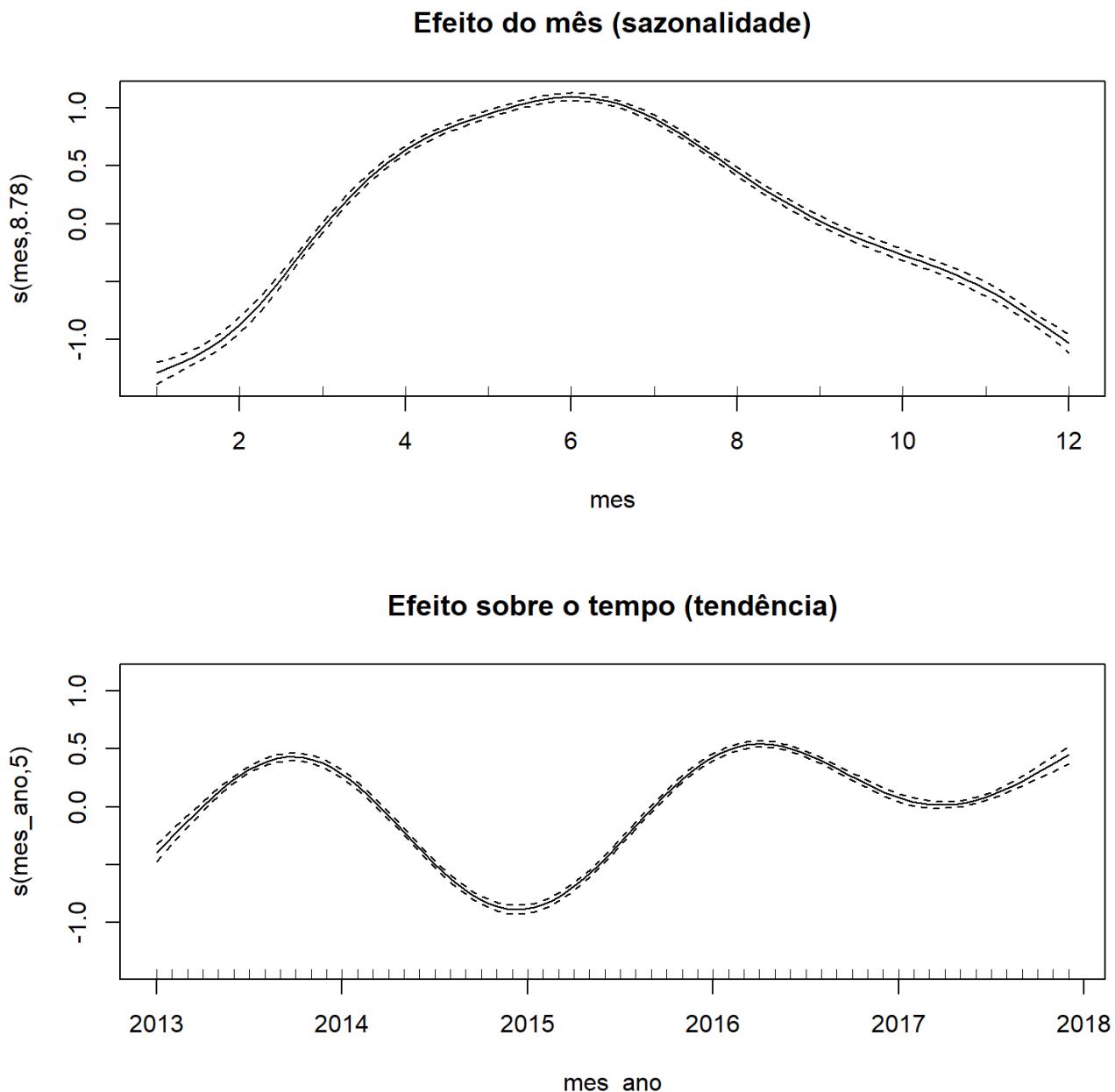
ano	mes	n	dt_mes	mes_ano
2013	1	54	2013-01-01	2013.000
2013	2	66	2013-02-01	2013.083
2013	3	108	2013-03-01	2013.167
2013	4	291	2013-04-01	2013.250
2013	5	718	2013-05-01	2013.333

Vamos agora inserir uma função suave sobre essa variável criada e visualizar os efeitos estimados (Figura 47):

```
mod_gam2 ← gam(n ~ s(mes) + s(mes_ano, k=6), data=srag_pr_mes, family = "poisson")
par(mfrow=c(2,1))

plot(mod_gam2, select = 1, main = "Efeito do mês (sazonalidade)")
plot(mod_gam2, select = 2, main = "Efeito sobre o tempo (tendência)")
```

Figura 47. Efeitos não lineares da sazonalidade e da tendência temporal estimados por modelo GAM aos dados de SRAG no Paraná.

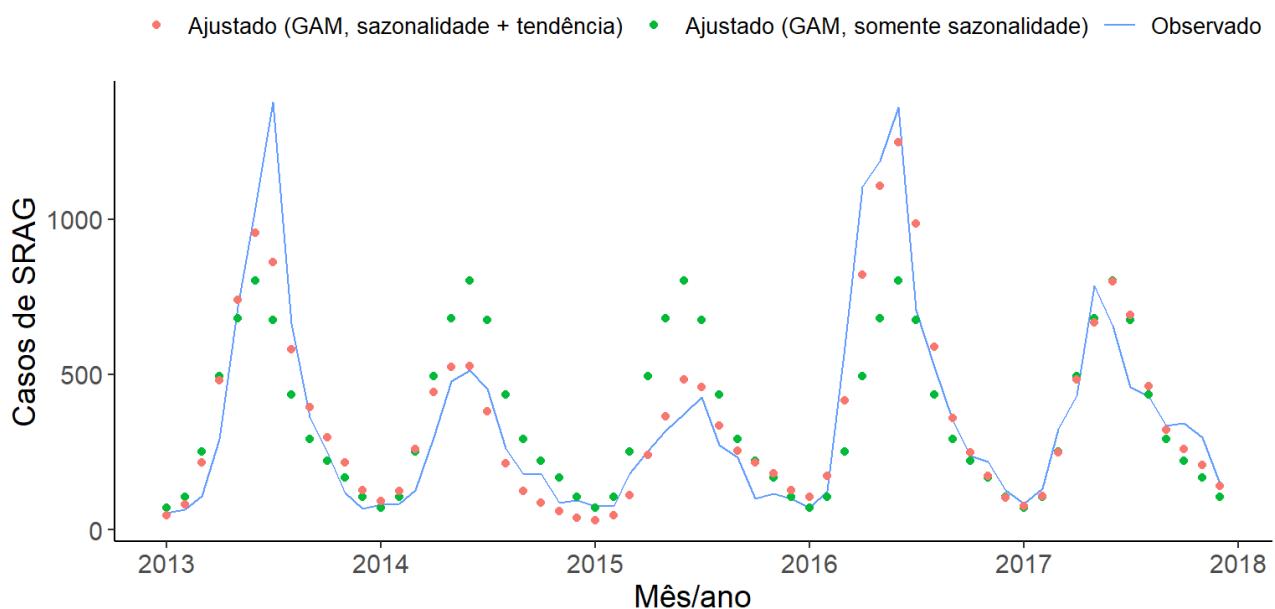




- Este script está ajustando o modelo GAM através do comando `mod_gam2 ← gam(n ~ s(mes) + s(mes_ano, k=6), data=srag_pr_mes, family = "poisson")` com suavizações para capturar os efeitos não lineares da sazonalidade mensal (`s(mes)`) e da tendência temporal ao longo dos anos (`s(mes_ano, k=6)`), utilizando distribuição de Poisson para modelar contagens.
- Para a configuração dos gráficos, está sendo utilizada a função `par(mfrow=c(2,1))`, que divide a janela gráfica em duas áreas verticais para exibir dois gráficos.
- E para visualização dos efeitos, está sendo utilizado:
 - `plot(mod_gam2, select = 1, main = "Efeito do mês (sazonalidade)")`: Mostra o efeito da sazonalidade mensal sobre os casos.
 - `plot(mod_gam2, select = 2, main = "Efeito sobre o tempo (tendência)")`: Mostra a tendência de longo prazo nos casos ao longo dos anos.

Vamos também ver os valores ajustados (Figura 48):

Figura 48: Casos de SRAG no Paraná observados comparado com valores ajustados por modelos GAM com e sem tendência.



Da mesma forma que testamos com a sazonalidade, podemos testar diferentes valores de **k** para o termo da tendência:

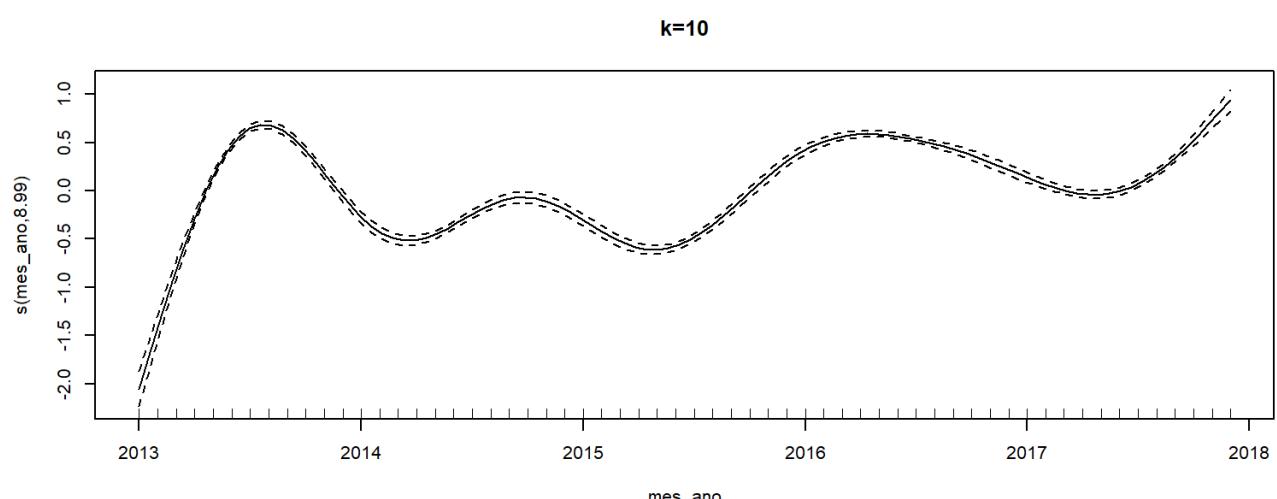
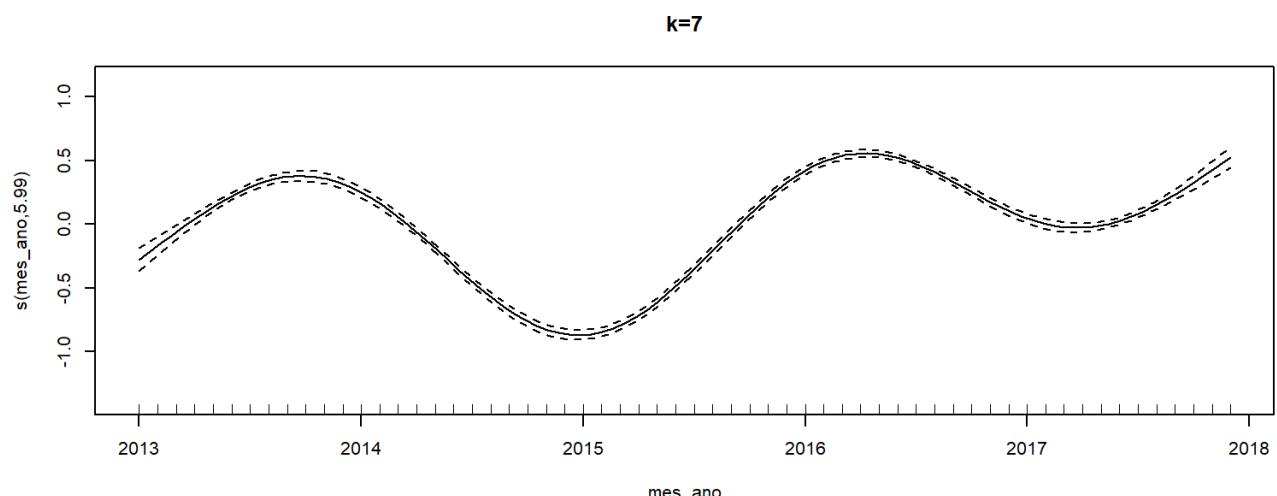
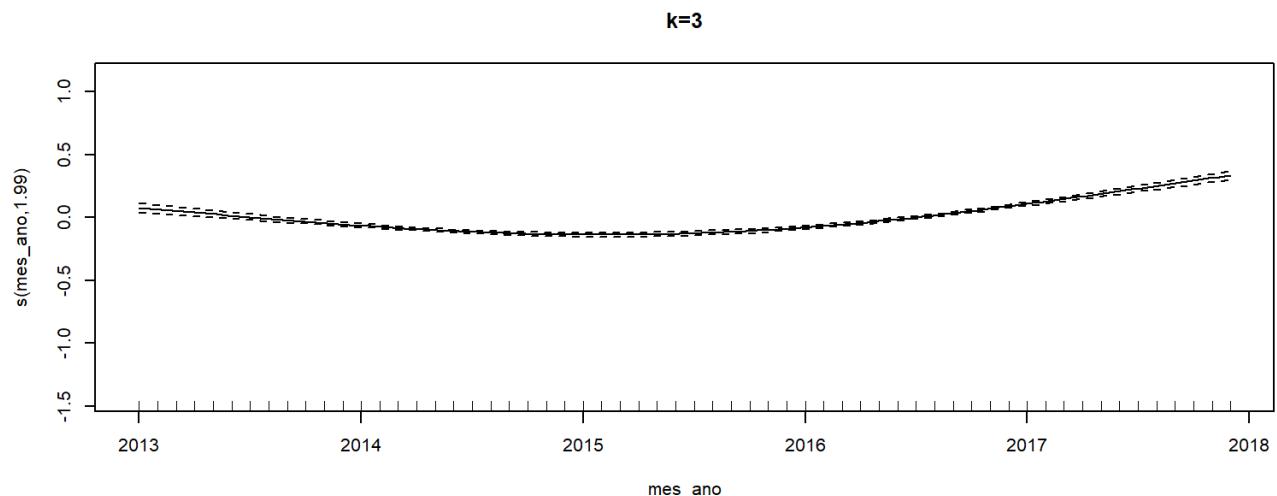
```
gam7 <- gam(n ~ s(mes) + s(mes_ano, k=3), data=srag_pr_mes, family = "poisson")
gam8 <- gam(n ~ s(mes) + s(mes_ano, k=7), data=srag_pr_mes, family = "poisson")
gam9 <- gam(n ~ s(mes) + s(mes_ano, k=10), data=srag_pr_mes, family = "poisson")

par(mfrow=c(3,1))

plot(gam7, select=2, main="k=3")
plot(gam8, select=2, main="k=7")
plot(gam9, select=2, main="k=10")
```



Figura 49: Relação entre mês de primeiros sintomas e casos de SRAG no Paraná estimada com diferentes dimensões de tendência de modelo GAM.



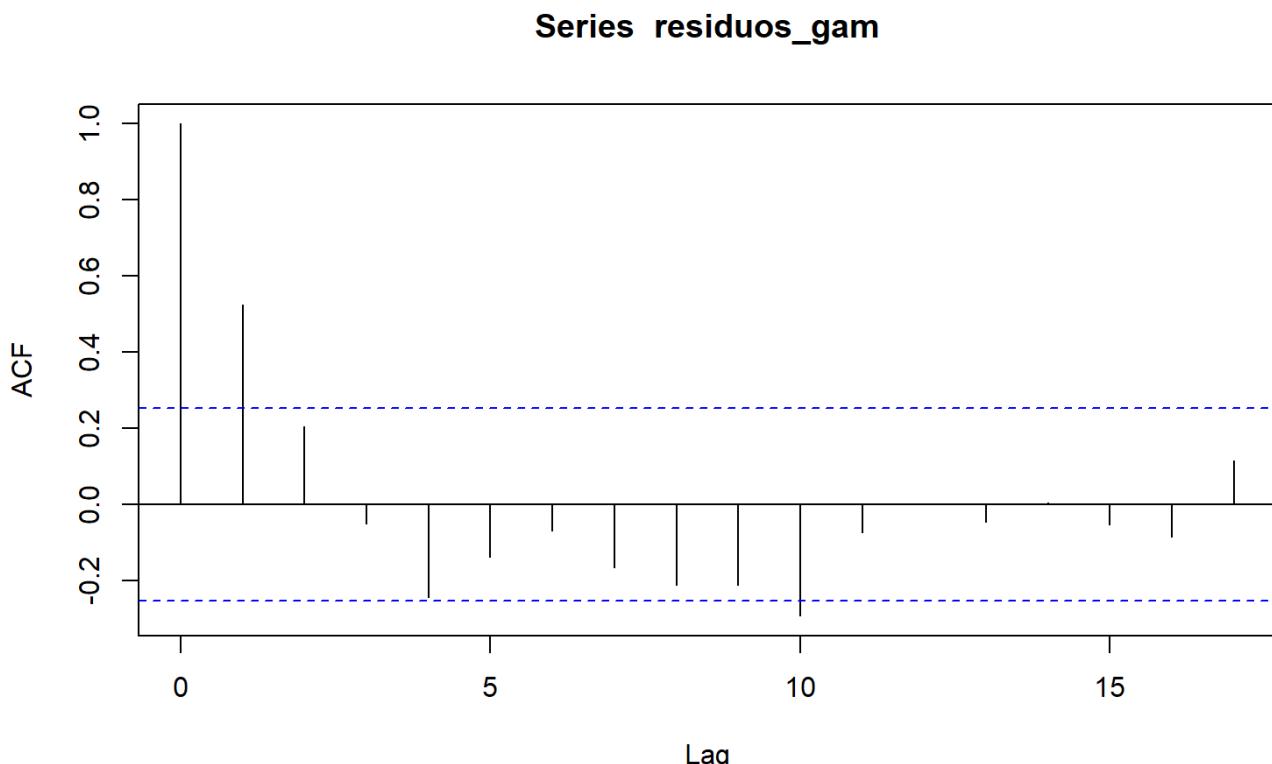
- O script está ajustando três modelos aditivos generalizados (GAM) para a variável `n` (casos de SRAG) no conjunto de dados `srag_pr_mes`, variando o número de nós (`k`) para suavização da tendência temporal (`mes_ano`). Em seguida, ele exibe gráficos comparando o efeito da suavização com diferentes valores de `k`.
- O Ajuste de modelos GAM com diferentes níveis de suavização:
 - `gam7`: Suavização da tendência ao longo do tempo com `k=3` (suavização mais rígida, menos flexível),.
 - `gam8`: Suavização com `k=7` (mais flexível que `k=3`).
 - `gam9`: Suavização com `k=10` (ainda mais flexível).
- Configuração de múltiplos gráficos:
 - `par(mfrow=c(3,1))`: Configura a área gráfica para exibir três gráficos empilhados (3 linhas, 1 coluna).
- Visualização do efeito da tendência temporal `s(mes_ano)`:
 - `plot(gam7, select=2, main="k=3")`: Mostra o efeito da tendência para o modelo com `k=3`.
 - `plot(gam8, select=2, main="k=7")`: Mostra o efeito da tendência para o modelo com `k=7`.
 - `plot(gam9, select=2, main="k=10")`: Mostra o efeito da tendência para o modelo com `k=10`.
- Resumindo, este script visa ajustar modelos GAM com diferentes graus de suavização para a tendência temporal e compara visualmente o efeito da escolha do número de nós (`k`) na flexibilidade das curvas de tendência.

Veja na Figura 49 que um valor de `k` pequeno captura uma tendência de mais **longo prazo** e, conforme aumentamos o valor de `k`, temos mais **especificidade** sobre as variações ao longo do tempo.

De forma similar à feita com modelos ARIMA, podemos verificar se há algum padrão restante nos resíduos (como autocorrelação):

```
residuos_gam ← residuals(mod_gam2)
acf(residuos_gam)
```

Figura 50: Função de autocorrelação dos resíduos do modelo GAM.

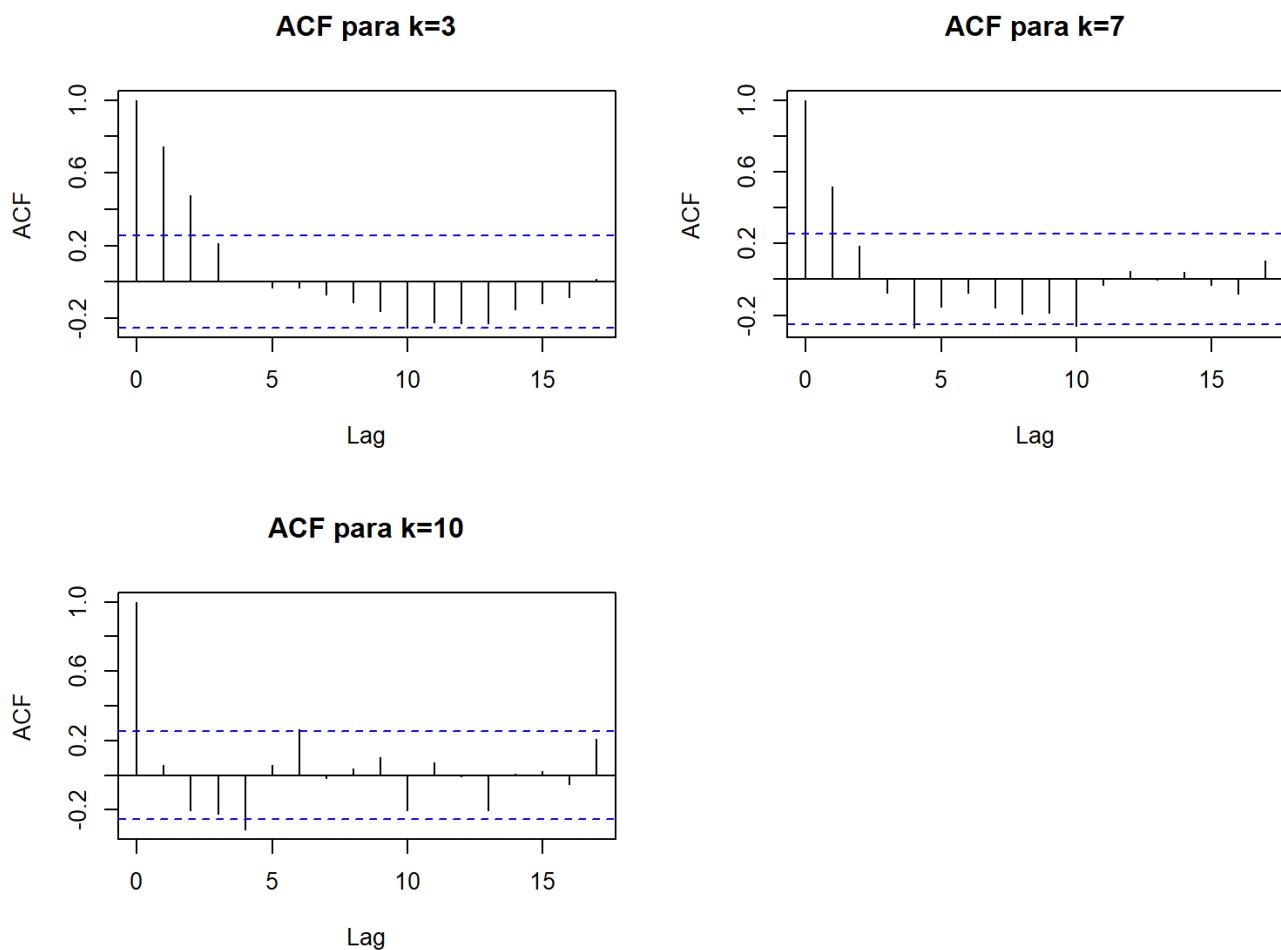


- `residuos_gam ← residuals(mod_gam2)`: Calcula os resíduos do modelo GAM ajustado `mod_gam2` e os armazena no objeto `residuos_gam`.
- `acf(residuos_gam)`: Gera o gráfico da função de autocorrelação (ACF) dos resíduos, avaliando se há dependência temporal (autocorrelação) entre eles.

O objetivo é verificar se os resíduos do modelo são independentes, como esperado para um bom ajuste de modelo.

Vemos que ainda há certa autocorrelação no lag 1. Podemos comparar como a autocorrelação dos resíduos se comporta entre os diferentes valores de `k` testados:

Figura 51: Função de autocorrelação dos resíduos de modelos GAM com diferentes dimensões de tendência.



Observamos na Figura 51 que os resíduos dos modelos para $k=3$ e $k=7$ possuem autocorrelação relevante nos primeiros lags. O modelo com $k=10$ se comporta melhor, mas ainda com autocorrelação presente no lag 4.



Note que modelar a tendência da série é uma forma **simplificada** de entender os componentes da série, e pode ser utilizado de maneira exploratória em diversos casos. Quando há dependência temporal entre as observações (**autocorrelação**), a forma correta de modelar a série é por meio de termos de autocorrelação (AR) ou de Médias Móveis (MA). Para os modelos GAM, há uma extensão chamada de **Modelos Aditivos Generalizados Mistos (GAMM)**, que possibilita a incorporação de termos de AR ou MA ao modelo que poderão tratar a autocorrelação de maneira mais adequada. Para maior compreensão sobre esse tipo de modelos, recomendamos a leitura de [Pedersen et. al \(2019\)](#).

```
srag_pr_futuro ← srag_pr_mes %>%
  select(1:3) %>%
  add_row(tibble(ano = 2018, mes = 1:12)) %>%
  mutate(
    dt_mes = as.Date(paste0(ano, “-”, mes, “-01”)),
    mes_ano = ano + (mes-1)/12
  )

previsao_gam ← mod_gam2 %>% predict(newdata = srag_pr_futuro, se.fit = T)
previsao_gam$inf ← previsao_gam$pred - 1.96 * previsao_gam$se
previsao_gam$sup ← previsao_gam$pred + 1.96 * previsao_gam$se
```

Este script está preparando dados futuros para projeção e realizando previsões com o modelo GAM ajustado, incluindo intervalos de confiança.

1. Criação de um conjunto de dados para previsão futura

- `select(1:3)`: Seleciona as três primeiras colunas do conjunto de dados `srag_pr_mes`.
- `add_row(tibble(ano = 2018, mes = 1:12))`: Adiciona 12 novas linhas ao conjunto de dados, representando os meses de janeiro a dezembro de 2018.
- `mutate(...)`: Cria/transforma colunas:
 - `dt_mes`: Converte as combinações de ano e mes em datas no formato "YYYY-MM-DD" (sempre o dia 1).
 - `mes_ano`: Converte ano e mês em uma escala contínua para facilitar cálculos temporais.

2. Previsão utilizando o modelo GAM

- `predict()`: Gera previsões a partir do modelo `mod_gam2`, utilizando o conjunto de dados `srag_pr_futuro`.
- `newdata = srag_pr_futuro`: Aplica as previsões aos dados futuros.
- `se.fit = T`: Retorna também o erro padrão das previsões (standard error).

3. Cálculo do intervalo de confiança

- Calcula os limites do intervalo de confiança de 95% para as previsões:
 - `inf`: Limite inferior = previsão (pred) - 1,96 × erro padrão (se).
 - `sup`: Limite superior = previsão (pred) + 1,96 × erro padrão (se).

Resumindo, as linhas de comando acima tem por objetivo preparar o conjunto de dados para previsão futura (adicionando novos dados), realizar previsões com o modelo GAM ajustado e calcular intervalos de confiança de 95% para as previsões.



Conclusões

Além dos modelos apresentados neste curso, como os modelos clássicos ARIMA e SARIMA, existe uma ampla gama de modelos de séries temporais, que podem ser utilizados em diferentes contextos. Entre eles, destacam-se:

- Modelos de Suavização Exponencial: Antecessores dos modelos ARIMA, oferecem uma abordagem intuitiva e computacionalmente eficiente. O método de **Holt-Winters**, por exemplo, é particularmente adequado para séries com tendência e sazonalidade, apresentando uma implementação mais simples em relação aos modelos aqui apresentados.

Algumas referências adicionais sobre o assunto:

- Análise de Séries Temporais. MORETTIN, Pedro Alberto; TOLOI, Clélia de P. 2. ed. São Paulo: Blucher, 2006.
- Análise de Séries Temporais em R: Curso Introdutório. FERREIRA, Pedro Costa. GEN | Atlas, 2020.
- Forecasting: Principles and Practice. HYNDMAN, R. J.; ATHANASOPOULOS, G. 3. ed. OTexts, 2021. Disponível em: <https://otexts.com/fpp3/>. Acesso em: 30 dez. 2024.
- Modelos de Espaço de Estados: Uma classe de modelos mais complexa, porém extremamente versátil. A dinâmica da série temporal é modelada através de um estado latente não observado, permitindo capturar uma variedade de padrões, como tendências, sazonalidades, componentes cíclicos e efeitos de intervenção. O Filtro de Kalman é um algoritmo fundamental para estimação e previsão nesses modelos.

Algumas referências sobre o assunto:

- Time Series Analysis by State Space Methods. DURBIN, James; KOOPMAN, Siem Jan. Time Series Analysis by State Space Methods. 2. ed. Oxford: Oxford University Press, 2012.
- State-Space Methods for Time Series Analysis. CASALS, José; GARCÍA-HIERNaux, Alfredo; JEREZ, Miguel; SOTOCA, Sonia; TRINDADE, Alexandre. State-Space Methods for Time Series Analysis. CRC Press, 2017
- Time Series Analysis for the State-Space Model with R/Stan. HAGIWARA, Junichiro. Time Series Analysis for the State-Space Model with R/Stan. Springer, 2021.
- Modelos Bayesianos em Séries Temporais: Oferecem uma abordagem probabilística completa, incorporando incerteza e conhecimento prévio (informação *a priori*) no processo de modelagem. Ao contrário dos métodos chamados de frequentistas, que tratam os parâmetros como valores fixos, os modelos Bayesianos tratam os parâmetros como variáveis que possuem também distribuições de probabilidade. A inferência é realizada por meio do Teorema de Bayes, que leva a uma distribuição chamada de posteriori dos parâmetros. Métodos computacionais avançados, como MCMC (simulação de Monte Carlo via Cadeias de Markov), são frequentemente empregados para realizar a inferência. Essa abordagem permite quantificar a incerteza de forma mais completa e lidar com dados faltantes de maneira natural.

Algumas referências sobre o assunto:

- Modelos bayesianos univariados aplicados à previsão de séries econômicas: uma aplicação à previsão da produção industrial brasileira. COSTA, Evandro; MIGON, Hélio N. Revista Brasileira de Economia, v. 47, n. 4, p. 527-550, 1993. Disponível em: <https://periodicos.fgv.br/bre/article/download/2983/1877/5255>. Acesso em: 30 dez. 2024.
- Bayesian Forecasting and Dynamic Models. WEST, Mike; HARRISON, Jeff. 2nd ed. New York: Springer, 1997.
- Modelos bayesianos zero-modificados para séries temporais de contagem. 2020. ASSIS, Caroline Mendes de. Tese (Doutorado em Estatística) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2020. Disponível em: https://www.teses.usp.br/teses/disponiveis/104/104131/tde-24072020-093829/publico/CarolineAssis_2020_revisada.pdf. Acesso em: 30 dez. 2024.

- Bayesian Time Series Models. BARBER, David (Ed.); CEMGIL, A. Taylan (Ed.); CHIAP-PA, Silvia (Ed.). New ed. Cambridge: Cambridge University Press, 2011.
- Dynamic Time Series Models Using R-INLA: An Applied Perspective. RAVISHANKER, Nalini; RAMAN, Balaji; SOYER, Refik. Boca Raton: CRC Press, 2022.

No contexto da Epidemiologia, alguns modelos merecem destaque:

- Modelos Joinpoint (Análise de Regressão de Pontos de Junção ou Regressão segmentada): Utilizados para analisar tendências de longo prazo em dados epidemiológicos, identificando pontos no tempo onde as taxas de variação (inclinações) mudam significativamente. Esses pontos de junção indicam potenciais mudanças nos fatores de risco, intervenções de saúde pública ou outros eventos relevantes. O modelo estima a Taxa de Variação Anual Percentual (AAPC) para cada segmento entre os pontos de junção, permitindo quantificar as mudanças nas tendências.

Algumas referências sobre o assunto:

- Permutation tests for joinpoint regression with applications to cancer rates. KIM, H. J. et al. Statistics in Medicine, v. 19, n. 3, p. 335-351, 2000.
- MUGGeo, V. M. R. Segmented: An R package to fit regression models with broken-line relationships. R News, v. 8, n. 1, p. 20-25, 2008.
- SILVA, Paulo Victor de Oliveira e; et al. Tuberculose no sistema prisional brasileiro: cenários via Joinpoint entre 2007 e 2019. Cadernos de Saúde Pública, Rio de Janeiro, v. 39, n. 9, e00166722, 2023. Disponível em: <https://doi.org/10.1590/0102-311XPT166722>. Acesso em: 30 dez. 2024.
- NATIONAL CANCER INSTITUTE. Joinpoint Trend Analysis Software. Version 5.3.0. Division of Cancer Control & Population Sciences, 2024. Disponível em: <https://surveillance.cancer.gov/joinpoint/>. Acesso em: 30 dez. 2024.
- TAVARES, Ana Carolina Barreto. Análise de regressão joinpoint: tendência de mortalidade de hospitalizados por COVID-19 nas regiões do Brasil. 2022. Trabalho de Conclusão de Curso (Graduação em Estatística) – Universidade Federal de Pernambuco, Recife, 2022. Disponível em: <https://repositorio.ufpe.br/handle/123456789/53529>. Acesso em: 30 dez. 2024.

- Modelos de Séries Temporais Interrompidas (STI): Especificamente projetados para avaliar o impacto de intervenções ou eventos discretos em uma série temporal. A análise busca determinar se a intervenção causou uma mudança estatisticamente significativa no nível ou na inclinação da série. Modelos ARIMA com variáveis dummy ou modelos de regressão segmentada são frequentemente utilizados nesse tipo de análise. As STIs são cruciais para avaliar a eficácia de políticas públicas, campanhas de saúde e outras intervenções. É importante ressaltar que a escolha do modelo mais adequado depende do objetivo da análise, das características dos dados e da pergunta de pesquisa. A combinação de diferentes modelos e abordagens pode fornecer uma compreensão mais completa do fenômeno em estudo.

Algumas referências sobre o assunto:

- Interrupted time series regression for the evaluation of public health interventions: a tutorial. BERNAL, James Lopez; CUMMINS, Steven; GASPARINI, Antonio. International Journal of Epidemiology, v. 46, n. 1, p. 348-355, 2017. DOI: 10.1093/ije/dyw098. Disponível em: <https://doi.org/10.1093/ije/dyw098>. Acesso em: 30 dez. 2024.
- Data Science for Public Service: Regression Tools for Program Evaluation and Applied Research. LECY, Jesse; FUSI, Federica. Disponível em: <https://ds4ps.org/pe4ps-textbook/docs/index.html>. Acesso em: 30 dez. 2024.
- Interrupted time series designs in health technology assessment: lessons from two systematic reviews of behavior change strategies. RAMSAY, Craig R.; MATOWE, Lloyd; GRILLI, Roberto; GRIMSHAW, Jeremy M.; THOMAS, Ruth E. International Journal of Technology Assessment in Health Care, v. 19, n. 4, p. 613-623, 2003. DOI: 10.1017/S0266462303000576. Disponível em: <https://pubmed.ncbi.nlm.nih.gov/15095767/>. Acesso em: 30 dez. 2024.

O conteúdo apresentado neste curso visa apresentar os principais métodos para análise de dados sequenciais, abordando a dependência entre observações ao longo do tempo. Vimos que a introdução de estruturas de autocorrelação permite descrever os padrões temporais presentes nos dados, como tendências, sazonalidades e ciclos, e, consequentemente, realizar previsões. Os modelos exemplificados aqui como ARIMA, SARIMA, e GAM representam algumas das técnicas mais empregadas na área da saúde para análise de dados na dimensão temporal.

