Manual Econometria Aplicada no R

Lucas Augusto Silva Ribeiro
Universidade Federal do Rio Grande do Sul (UFRGS)
Estágio Docência para a matéria Econometria Aplicada 2020/2
Professor Sabino Porto Junior
https://github.com/Utreti/EconometriaAplicada

2021-04-16

Contents

1 Comentários Iniciais **5**

2 Introdução ao R **7**

2.1 Por que o R? 7

2.2 Estrutura de dados no R 8

2.3 Pacotes 11

2.4 Lendo Arquivos 13

2.5 Trabalhando com dados 13

2.6 Gráficos 14

2.7 Referências úteis para ajudar a mexer com o R 16

2.8 Exercícios para treino 16

3 Estatísticas Padrão e Regressão linear **19**

4 Séries Temporais no R **21**

4.1 Introdução a análise de séries temporais no R 21

4.2 Introdução a séries temporais 22

4.3 Modelo ARMA 26

4.4 Análise de uma Série Temporal 28

4.5 Metodologia de Box-Jenkins 32

5 Uso da Metodologia SARIMA **35**

5.1 Princípios básicos 35

5.2 Transformando uma série para ser passível de análise 36

5.3 Modelagem SARIMA 47

5.4 Predição 52

5.5 Exercícios para treino 55

Chapter 1

Comentários Iniciais

Este é um pequeno manual de uso do R para estudos econométricos e análise de dados feito com o Rbookdown. É um trabalho em progresso que tem como objetivo auxiliar os alunos de econometria aplicada da turma 2020/2.

Para entrar em contato sobre erros encontrados, sugestões e dúvidas, estou disponível no e-mail: lucas.augusto@ufrgs.br

Chapter 2

Introdução ao R

2.1 Por que o R?

O uso do software R tem várias vantagens, principalmente no meio acadêmico mas também possui várias vantagens pessoais:

- É gratuito, isto é uma vantagem para que possa ser usado sem necessidade de pagar por uma licença tanto por você quanto pela empresa que te contrate.

- O R por ser uma linguagem de programação ajuda em entender conceitos e ideias de linguagens de programação, facilitando o aprendizado de outras línguas.

- A existência de pacotes facilita em muito o uso de análises, provavelmente o que você quer fazer já foi feito por alguém e não há a necessidade de reinventar a roda.

- Integrabilidade com LaTeXpermite uma ótima fonte para a criação de pesquisas reprodutivas.

- A comunidade do R é bem extensa então erros e problemas encontrados provavelmente terão alguma solução no StackOverflow. Além de ser muito fácil encontrar material prático e teórico.

2.2 Estrutura de dados no R

2.2.1 Vetores

O R possui algumas estruturas principais. Vetores, fatores, matrizes, data frames e listas. Vetores são a estrutura mais básica do R, podem armazenar tanto caracteres como números mas apenas um tipo no mesmo vetor. O tipo dos elementos do vetor permite as operações que se pode fazer com eles, por exemplo não é possível a soma de vetores com caracteres (diferente do Python). Para criar um vetor, é apenas escrever os elementos do vetor dentro da estrutura c() separados por vírgulas.

```
letras1 <- c("a", "b", "c")

letras2 <- c("d", "e", "f")

letras1+letras2
```

```
## Error in letras1 + letras2: non-numeric argument to binary operator
```

Este erro que ocorreu era esperado, justamente por tentarmos fazer uma operação designada apenas a objetos do tipo numérico com objetos do tipo char.

Ao passo que vetores feitos com valores numéricos e com valores lógicos podem ser somados sem problemas. Vetores lógicos podem ser somados pois TRUE tem valor 1 e FALSE tem valor 0. Isto é um recurso muito útil ao passo que podemos assim descobrir quantas variáveis possuímos faltando, ou do tipo que nos interesse.

Perceba que enquanto letras precisam de parênteses para serem alocados dentro de um vetor, números não precisam. Isto acontece pois ao usarmos letras sem parênteses o R procura um objeto. Ao criarmos objetos colocamos ou = ou <- para atribuir seu valor. Vetores possuem apenas uma dimensão então para selecionar um elemento de um vetor se faz vetor[i], onde i é a posição do elemento.

```
numeros1 <- c(1,2,3)

numeros2 <- c(4,5,6)

a <- numeros1+numeros2

a
```

```
## [1] 5 7 9
```

```
a[2]
```

```
## [1] 7
```

2.2.2 Matrizes

Matrizes são uma concatenação de data que aceitam apenas valores numéricos. Percebam que criei um vetor com outros dois vetores para a construção dessa matriz. O formato que eles se encontram é devido ao chamado de uso de matriz, se não eles apenas ficariam concatenados um ao lado do outro.

```
matriz <- matrix(c(numeros1, numeros2), nrow=2, ncol = 3)

print(matriz)
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
print(c(numeros1, numeros2))
```

```
## [1] 1 2 3 4 5 6
```

2.2.3 Data Frames

Data frames são uma concatenação de data que aceitam mais de um tipo de objeto. Cada coluna num data frame é constituído por um tipo de objeto diferente. Um jeito de se pensar em data frames no R é colocar vários vetores um do lado do outro ou um acima do outro.

Data Frames, assim como matrizes possuem duas dimensões então para selecionar um elemento se escreve df[i,j]. Outra maneira é indexar a coluna do df e depois selecionar o elemento da coluna. É o objeto mais utilizado na análise de dados.

É interessante notar que nesta apostila será usado, sempre que possível o uso do pacote tidyverse e suas funcionalidades. Neste sentido, é necessária uma pequena menção ao formato tibble que, de maneira prática, não é muito diferente de um data frame normal porém permite uma função print mais simples e detalhada, permitindo um uso do print no objeto sem muitos problemas.

```
df <-  data.frame(numeros1, letras1, numeros2, letras2)

print(df)
```

```
##   numeros1 letras1 numeros2 letras2
## 1        1       a        4       d
## 2        2       b        5       e
## 3        3       c        6       f
```

```
df[1,3]
```

```
## [1] 4
```

```
df$letras1[1]
```

```
## [1] "a"
```

2.2.4 Listas

Listas são elementos extremamente versáteis e podem ser pensadas como um vetor capaz de concatenar tudo dentro dele. São mais flexíveis e complexos que data frames.

```
lista <-  list(numeros1, letras1, numeros2, letras2, df)

print(lista)
```

```
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] "a" "b" "c"
##
## [[3]]
## [1] 4 5 6
##
## [[4]]
## [1] "d" "e" "f"
##
## [[5]]
##   numeros1 letras1 numeros2 letras2
## 1        1       a        4       d
## 2        2       b        5       e
## 3        3       c        6       f
```

2.3 Pacotes

O uso de pacotes simplifica em muito o uso do R. Simplifica código, aumenta as capacidades de importação de dados, permite o uso de novas funções e

criação de visualizações de dados mais bonitas. Para instalar um pacote, é só usar a função install.packages(). Caso haja alguma dúvida no que a função faz e quais argumentos são usados por ela, é só escrever ?função, serve para base de dados dentro do R também.

Podemos instalar manualmente os pacotes ou instalar vários de uma vez. Interessante a instalação dos seguintes pacotes para a análise de dados e análise de séries temporais e leitura de dados:

- Tidyverse
- Openxlsx
- Foreign
- Tsibble
- Fable
- Feasts
- MTS
- urca
- vars

Uma maneira de instalar todos de uma vez é adicionar um objeto com o nome de todos os pacotes e então instalá-los:

```
pacotes ← c(tidyverse, openxlsx, foreign, tsibble, fable, feasts,
            mts, urca, vars)
install.packages(pacotes)
```

Claro que sempre há novos pacotes surgindo e substituindo antigos. Uma boa estratégia é ver na documentação do pacote (geralmente é só procurar o nome do pacote no google junto de documentation) quando foi sua última atualização e se ele tem sido atualizado periodicamente

2.3.1 Chamando pacotes

Para chamar um pacote para utilizá-lo basta escrever library(nome do pacote), note que ao contrário da instalação que requer aspas no nome do pacote, para chamá-lo isto não é necessário.

```
library(tidyverse)
library(openxlsx)
```

2.4 Lendo Arquivos

Para ler um arquivo no R, basta mandar ler o arquivo com a função apropriada para o tipo de arquivo. Por exemplo, para um arquivo csv usa-se ou read.csv ou read_csv (readr). Para ler xlsx usa-se read.xlsx(openxlsx). Faça isso enquanto atribui este valor a um objeto para que fique guardado nos objetos do R e possa ser manipulado, se não o R apenas irá ler o arquivo e colocar no seu terminal.

```
caminho <- "/home/lucas/Desktop/Mestrado/
estagio docencia/
Datasets/Econometrics with applied methods/
chapter7/xls/xm701inp.xlsx"

dados <- read.xlsx(caminho)

outrosdados <- read.csv("/home/lucas/Documents/bustabit.csv")
```

2.5 Trabalhando com dados

Primeiramente, é necessário explicar o operador pipe, %>%. Ele é o equivalente ao uso de uma função composta mas que deixa mais simples para um humano ler. Enquanto no R se queremos uma função composta faríamos assim como uma sintaxe matemática normal com uma h(g(f(x))), o operador pipe nos permite que façamos f(x)%>%g(x)%>%h(x). Pequenos exemplos serão dados com base de dados simples.

```
library(tidyverse)
carros <- mtcars

mtcars %>%
```

```
  filter(cyl >= 6) %>%
  summarize(media_mpg = mean(mpg))
```

```
##   media_mpg
## 1  16.64762
```

```
mean(carros[carros$cyl >= 6,]$mpg)
```

```
## [1] 16.64762
```

Ambos geram o mesmo resultado porém o segundo é muito mais simples de ser utilizado. Uma última nota sobre uso de sintaxe para análise de dados, para bases muito grandes, existe o pacote data.tables que possui uma sintaxe levemente diferente do padrão do R, e é extremamente rápida. Ao se trabalhar com big data, é melhor usar aquele pacote e aprender a manuseá-lo.

2.6 Gráficos

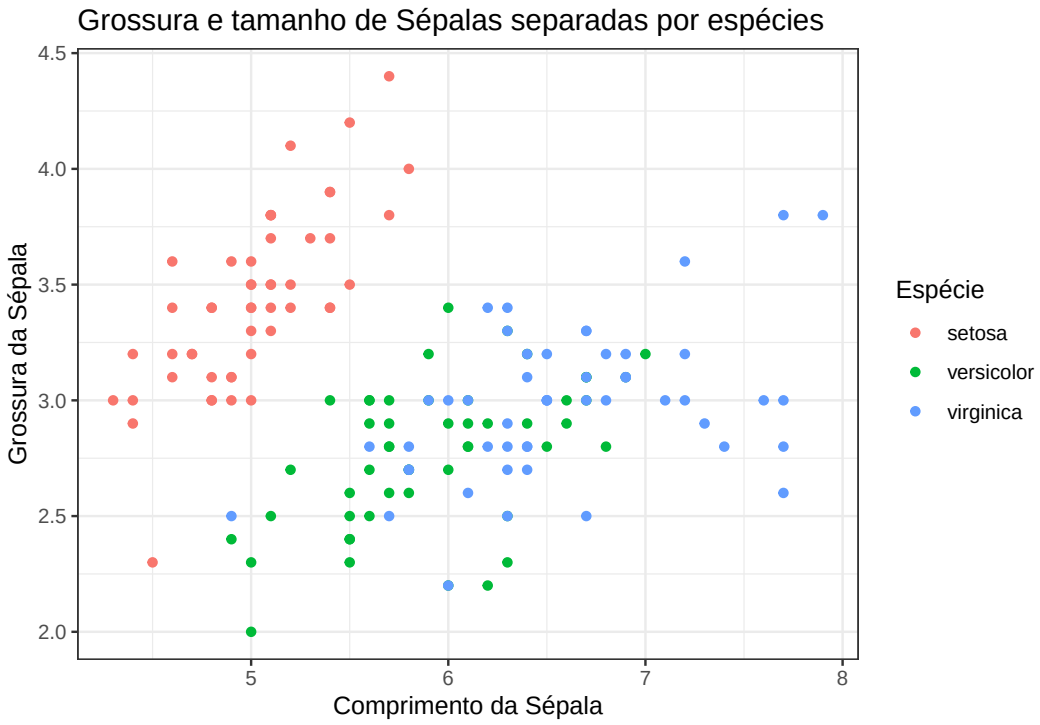
Demonstrar graficamente o que está acontecendo é importante tanto para o leitor quanto para quem está escrevendo algo, logo, bons gráficos ajudam não só nossa análise como a interpretação dela por terceiros. E agora uma verdadeira última nota sobre análise de dados: podemos ter base de dados no formato wide ou no formato long.
O formato long nos permite colocar gráficos de maneira mais simples com o ggplot usando alguma característica como um separador. Em geral, os dados que trabalhamos em economia estão no formato wide.

```
flores <- as_tibble(iris)

print(flores)
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
```

```
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # ... with 140 more rows
```

```
ggplot(data = flores, aes(x=Sepal.Length, y = Sepal.Width, color = Species))+
  geom_point()+
  theme_bw()+
  labs(x= "Comprimento da Sépala", y= "Grossura da Sépala", color = "Espécie")+
  ggtitle("Grossura e tamanho de Sépalas separadas por espécies")
```



A sintaxe do ggplot permite que sejam feitas várias modificações na maneira que se apresentam os dados, tentem mexer no argumento e camada que coloquei e terão diferentes gráficos. Cada camada é escrito na linha seguinte após o símbolo de +. A chamada padrao do ggplot é a que se encontra na primeira linha.

2.7 Referências úteis para ajudar a mexer com o R

- Data Typesand Structures - Programming with R. https://swcarpentry.github.io/r-novice-inflammation/13-supp-data-structures/
- Data analytics for beginners. https://data-flair.training/blogs/data-analytics-tutorial/
- Livro: R for Data Science https://r4ds.had.co.nz
- Livro: R Graphics Cookbook, 2nd edition https://r-graphics.org/
- Livro: Efficient R Programming https://csgillespie.github.io/efficientR/
- Livro R Programming for Data Science https://bookdown.org/rdpeng/rprogdatascience/
- Hands on programming with R https://rstudio-education.github.io/hopr/
- Livro: Pratical Data Science with R https://www.amazon.com.br/Practical-Data-Science-Nina-Zumel/dp/1617291560
- Site: Dicas e macetes no R https://tinyurl.com/rcjsdre

Obviamente, não é necessário ler todos os livros, alguns são mais completos que outros, escolha o que parecer mais interessante a você e siga com ele, alguns são ótimos manuais de bolso para consultar caso possua alguma dúvida. O graphics cookbook, por exemplo, é um bom manual quando houver a vontade de entender melhor o poder do pacote ggplot que não será abordado em aula.

2.8 Exercícios para treino

2.8.1 Algumas dicas antes dos exercícios

Como o tidyverse é enorme, ficam algumas funções que vocês podem encontrar no dplyr que ajudam a trabalhar com base de dados:

- filter() - permite filtrar e fazer uma seleção de dados que retornem TRUE em um operador lógico dado: idade>10, sexo==“homem,” etc.
- group_by() - é uma função que permite agrupar um determinado grupo que retorne TRUE em um operador lógico e faça o calculo para cada grupo.
- summarise() - em conjunto com o group_by() permite cálculos rápidos por grupo como média, soma, contagem, etc.
- mutate() - permite criar novas váriaveis num dataframe, que são funções de variáveis já existentes
- arrange() - modifica a ordem das linhas
- pivot_Longer(), pivot_wider() - transforma dados no formato long(wide), regra geral: long é bom para gráficos, wide para se analisar.

Ler a documentação do pacote ajuda a aprender as possibilidades que temos na hora de manipular os dados, procurar no google também, ele é seu melhor amigo.

2.8.2 Onde achar os pacotes de dados necessários para realizar os exercícios?

As bases de dados e tabela para comparação podem ser encontrados em:

https://www.dieese.org.br/analiseped/mensalBSB.html (tabela)
https://www.dieese.org.br/analiseped/microdadosBSB.html (microdados)

As bases de dados para o exercício 2 e 3 podem ser baixados também pelo meu github no endereço:
https://github.com/Utreti/EconometriaAplicada

2.8.3 Agora sim! Exercícios

1. Procure uma base de dados qualquer presente no R, faça um filtro e crie um gráfico no R.

2. Usando o pacote dbc (leia sobre!), leia os dados de 2019 no RS que podem ser encontrados aqui:
   http://www2.datasus.gov.br/DATASUS/index.php?area=0901&item=1&acao=26

Leia a documentação e calcule a quantidade de mortes por ocorrência, segundo grupo do CID-10 no Rio Grande do Sul, separe entre homem e mulher. Compare com a tabela geral que é possível produzir no próprio site. Desafio: Baixe todos os dados, transforme tudo em um só data frame e faça uma comparação gráfica dos óbitos por estado.

3. Desafio: Usando o pacote haven para ler o arquivo do tipo sav da PED-DF (pesquisa de emprego e desemprego do DIEESE), tentar conseguir fazer com que valores totais da população ativa de 14 anos ou mais na tabela dos resultados mensais no mês de Agosto de 2019 seja o mesmo que o seu. (O valor é 2447)

Uma pequena palavra de cautela é ao trabalhar com dados, checar se não há NA que é o equivalente do R de missing values. Somas e outras operações feitas com NA retorna NA, exceto expressamente dito o contrário. (Em geral, é boa prática limpar a base desses valores faltando.)

Um site interessante para manusear dados voltado para datascience e tópicos mais avançados como Machine Learning é o https://www.kaggle.com/

Chapter 3

Estatísticas Padrão e Regressão linear

Esta sessão será feita após o término da parte de séries temporais e dados de painel. Devido a como o curso está progredindo no momento.

VEMAÍ

Chapter 4

Séries Temporais no R

4.1 Introdução a análise de séries temporais no R

O software R, sendo extremamente útil para análises estatísticas, possui uma ampla gama de opções, pacotes e métodos para se trabalhar com séries temporais. Trabalharemos principalmente com objetos do formato tipo tsibble, mas, quando necessário, outros tipos de objetos serão usados. Afinal, se o R é bem maleável, devemos explorar esta versatilidade dele. Os principais pacotes usados para o R na análise de séries temporais são, entre outros: fable, feasts, tsibble, urca, MTS, vars, zoo, etc.

4.1.1 Objetos temporais e pacotes no R

Para se fazer análise temporal é necessário, obviamente, trabalhar com séries temporais. Objetos assim precisam possuir um indexador de tempo e valores observados. Pacotes como lubridate e tsibble permitem que mexamos com tipos de data POSIXlt para que sejam mais convenientes para trabalharmos. O pacote tsibble nos permite, também, converter um arquivo no R para um arquivo do tipo ts (time séries). Outras maneiras existem, como funções de coerção como do próprio R base como o as.ts e o do pacote zoo, o as.zoo. Em geral, toda função as."objeto" é uma função que faz uma coerção de um tipo de objeto para outro tipo.

4.2 Introdução a séries temporais

4.2.1 Conceitos básicos de séries temporais

Ao aprendermos a análise de séries temporais aprendemos alguns conceitos fundamentais que são usados para a análise e criação de modelos. São com base neles que podemos analisar, predizer e testar nossos modelos. É necessário que esses conceitos estejam muito bem definidos e claros para o nosso uso deles. São eles:

- Estacionariedade
- Ruído Branco
- Sazonalidade
- Tendência
- Autocorrelação

4.2.2 Estacionariedade Fraca

O conceito de estacionariedade em séries temporais pode se referir tanto a estacionariedade estrita ou estacionariedade fraca. Ao trabalharmos com séries temporais, trabalhamos com o conceito de estacionariedade fraca.

Uma série, com variância finita, será fracamente estacionaria se:

- Sua média é constante ao longo do tempo
- $\gamma_{t,t-k} = \gamma_{0,k}$ $\forall$ tempo t e lag k

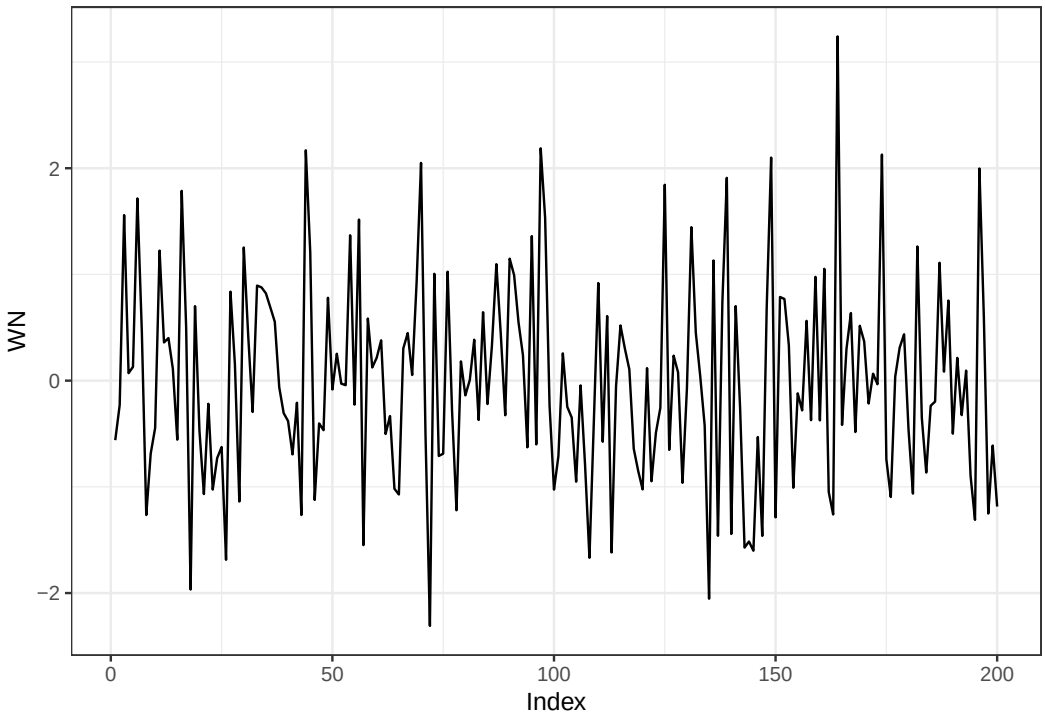
Onde $\gamma_{t,t-k}$ é sua autocovariância entre o tempo t e o tempo t-k.

4.2.3 Ruído Branco(RB)

O ruído branco é uma das séries estacionárias mais importantes pois muitos processos podem ser construídos a partir de um ruído branco. Como o ruído branco é uma série fracamente estacionária, temos:

- $E(\varepsilon_t) = 0 \forall t$
- $E(\varepsilon_t^2) = \sigma^2 \forall t$
- $E(\varepsilon_t \varepsilon_{t-j}) = 0, \forall j \neq 0$

Em particular, se o ruído branco segue a distribuição $\varepsilon_t \sim N(0, \sigma^2)$ este é chamado de ruído branco gaussiano.



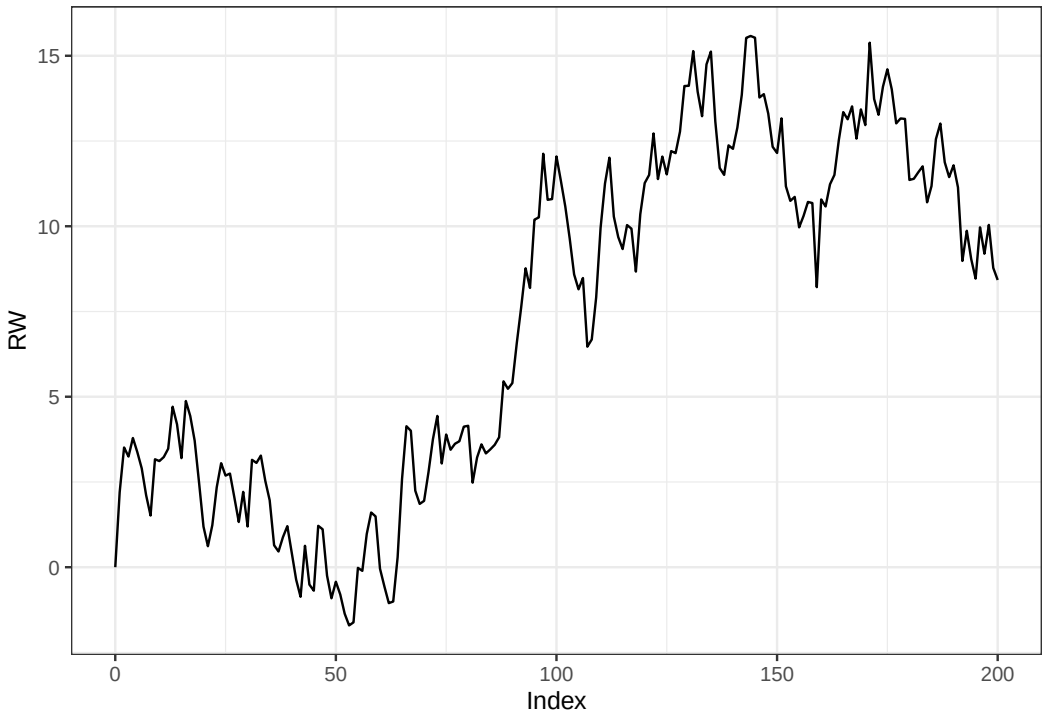
4.2.4 Passeio Aleatório

Suponha agora que temos uma sequência de variáveis aleatórias identicamente e independentemente distribuídas de média 0 e variância σ_e^2 construída como:

$$\begin{cases} Y_1 = e_1 \\ Y_2 = e_1 + e_2 \\ \vdots \\ Y_t = e_1 + e_2 + \ldots + e_t \end{cases} \qquad \text{Alternativamente: } Y_t = Y_{t-1} + e_t \tag{4.1}$$

Este processo é denominado passeio aleatório e possui as seguintes propriedades:

- $E(y_t) = 0$
- $E(y_t^2) = t\sigma^2$
- $\gamma t, s = cov(Y_t, Y_s) = t\sigma^2, \forall j \neq 0$



Compare este gráfico com o anterior, perceba que a média não é mais estável no tempo, esta série é não-estacionária.

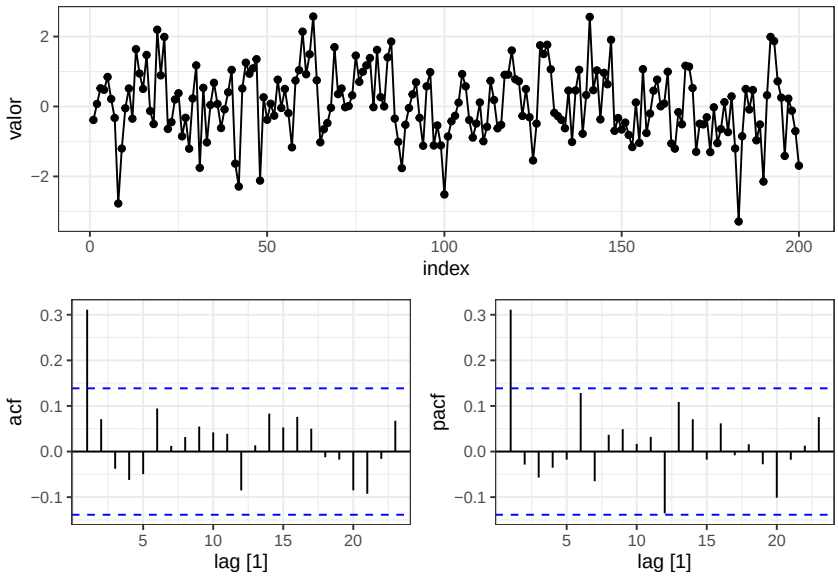
4.2.5 Processo Autorregressivo (AR), FAC e FACP

A Função de Autocorrelação (FAC) é o gráfico da autocorrelação contra a defasagem. A FAC de um processo MA(q) é significante até q lags e decai exponencialmente no gráfico do FACP. A FACP consiste, *a grosso modo* de

regredir y_t contra y_{t-1} e y_{t-2} e avaliar os coeficientes em um gráfico contra as ordens de defasagem.

Considere, agora o seguinte processo estocástico:

$$y_t = c + \phi yt - 1 + \varepsilon_t$$



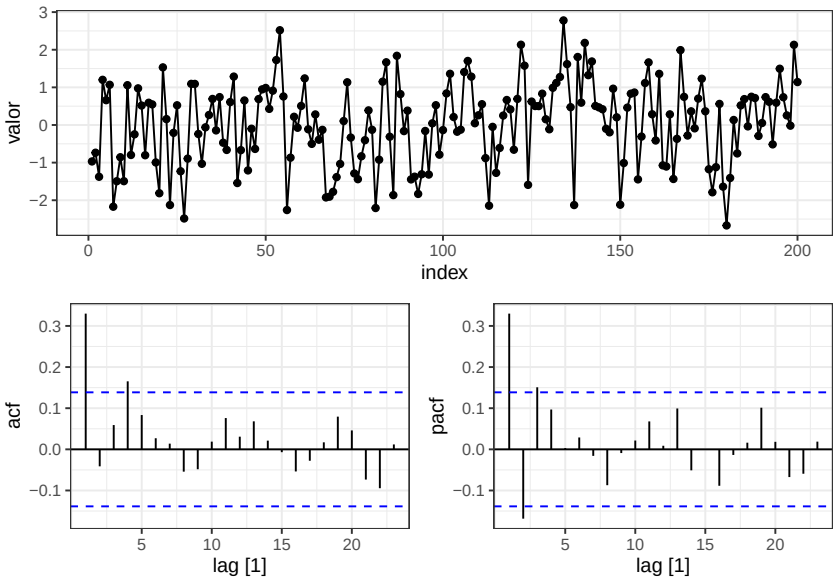
Onde ε_t é um RB. Este processo é denominado um processo Autorregressivo de ordem 1, ou AR(1). A FAC de um processo AR(p) cai exponencialmente a cada p lag e se mantém significante até o lag p no gráfico do FACP. Selecionamos a ordem de defasagem p do AR baseado em quantos lags significantes temos no FACP.

4.2.6 Médias Móveis(MA), FAC e FACP

Considere o processo estocástico:

$$y_t = \mu + \varepsilon_t + \theta\varepsilon t$$

Onde ε_t é um RB. Este processo é chamado de médias móveis. Como ele depende do erro contemporâneo e o erro imediatamente passado, é denotado como um MA(1).

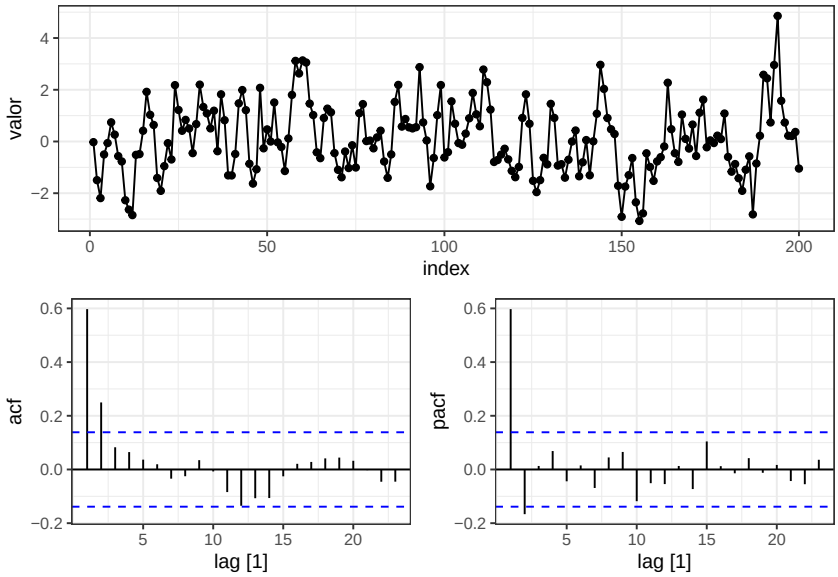


A FACP de um processo MA(q) cai exponencialmente no q lag e se mantém significante até o lag q no gráfico do FAC. Selecionamos a ordem de defasagem q do MA baseado em quantos lags significantes temos no FAC.

Note que no gráfico do ACF, o primeiro valor é dado no 0, ignora-se esse e se contam os lags significativos depois do 0.

4.3 Modelo ARMA

Um processo ARMA, então será um processo que possui tanto um componente Autorregressivo quanto um de médias móveis. Caso ele seja estacionário, podemos fazer sua análise.



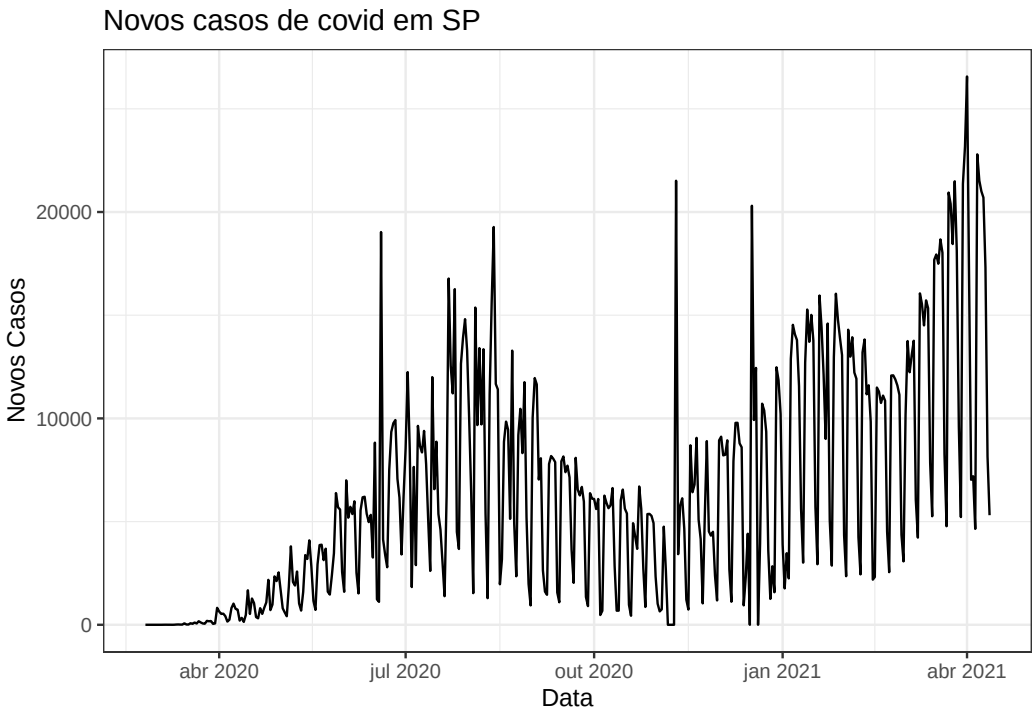
4.3.1 Componentes de uma Série Temporal

Toda série temporal mais complexa que trabalhamos possui pelo menos um dos seguintes fatores:

- Sazonalidade
- Ciclo
- Tendência

Estes componentes fazem com que nossa série não seja mais fracamente estacionária e precisamos, então, trabalhá-la para que possamos fazer alguma análise.

```
covid_sp <- readRDS("covid.RDS")

 ggplot(data = covid_sp, aes(x=date, y= newCases))+
    geom_line()+
    labs(x = "Data", y = "Novos Casos")+
    ggtitle("Novos casos de covid em SP")
```



Esta série dos dados do covid possui tanto uma tendência positiva quanto uma sazonalidade a cada sete dias devido a forma que os dados são colocados no sistema. Modelos do tipo ARIMA não são bons para esse tipo de série mas caso fossem utilizados seria necessário corrigir a sazonalidade a cada 7 dias, tendência e a volatilidade crescente antes de qualquer análise.

4.4 Análise de uma Série Temporal

```
us_retail_employment <- us_employment %>%
  filter(year(Month) >= 1990, Title == "Vendas de Varejo") %>%
  select(-Series_ID)

dcmp <- us_retail_employment %>%
  model(stl = STL(Employed))
```

```
## Error in `[[.default`(mdl, 1): subscript out of bounds
```

```
components(dcmp) %>%
  autoplot()+
  ggtitle("Decomposição de uma série temporal")
```

```
## Error in components(dcmp): object 'dcmp' not found
```

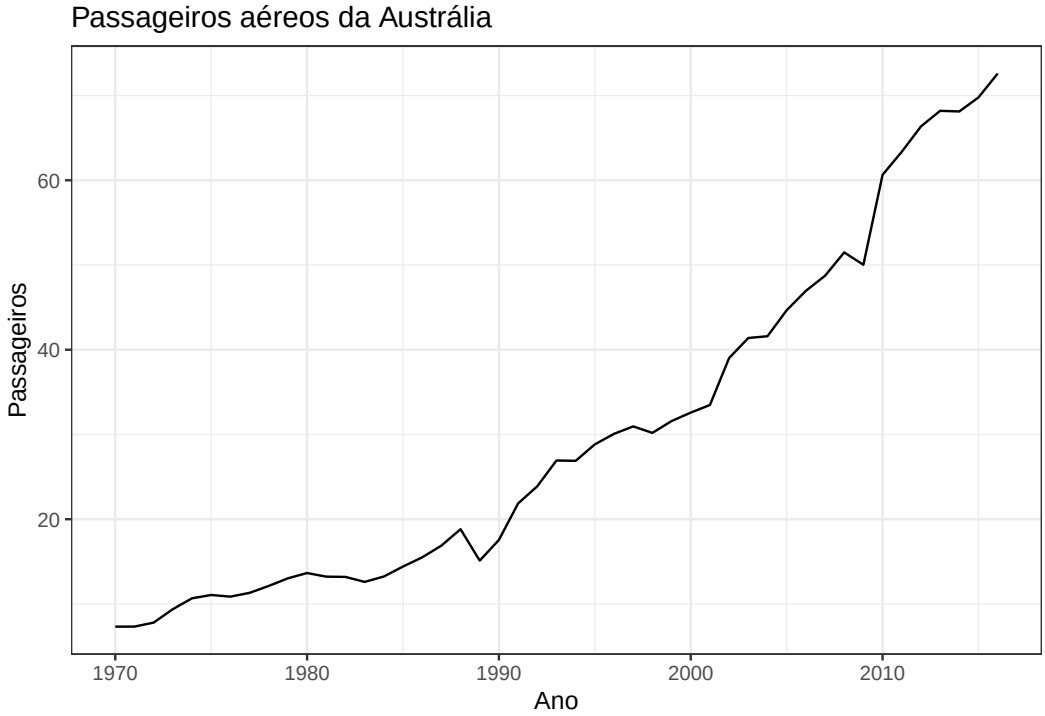
4.4.1 Metodologia Análise de uma Série Temporal por SARIMA

O primeiro passo ao analisar uma série temporal é fazer seu gráfico e analisá-lo, ver quais características ele possui e, a partir disso, ir trabalhando cada um de seus componentes para que o que sobre a nós seja algo estacionário que possamos analisar.

4.4.2 Tratando tendência

Uma série possui tendência quando há um crescimento ou decaimento de longo prazo no nível dos dados. Não precisa ser linear, pode ser visto como uma "mudança de direção."

```
autoplot(as_tsibble(aus_airpassengers))+
  ggtitle("Passageiros aéreos da Austrália")+
  labs(x="Ano", y = "Passageiros")
```



Quando uma série possui tendência ela não é estacionária e muito provavelmente possui uma raiz unitária. Precisamos, então, testar para a existência de uma raiz unitária e caso exista, lidar com ela.
O processo normalmente usado para lidar com a existência de uma raiz unitária é o uso de diferenças na séria. Num ARIMA(p,d,q) enquanto p e q se referem a ordem das partes do AR e MA, respectivamente, o d é referente a integração. Ou seja, é o número de quantas vezes precisamos tirar a diferença da série para que ela se torne estacionária. Alguns dos testes utilizados são o ADF – Augmented Dickey Fueller, Phillips-Person(PP) e o Kwiatkowski-Phillips-Schmidt-Shin (KPSS).

4.4.3 Tratando Volatilidade

Muitas vezes uma tendência nos traz um aumento de volatilidade, ou seja, nossa variância não é mais constante no tempo. Quando usamos modelos SARIMA, temos como hipótese uma variância constante pelo tempo pois são processos advindos de ruídos brancos. Se a variância não está constante em relação ao tempo, necessitamos, então lidar com ele. Um método muito usado

é trabalhar com o log dos valores a serem trabalhados. Uma generalização para esse problema é o uso de uma transformação Box-Cox.

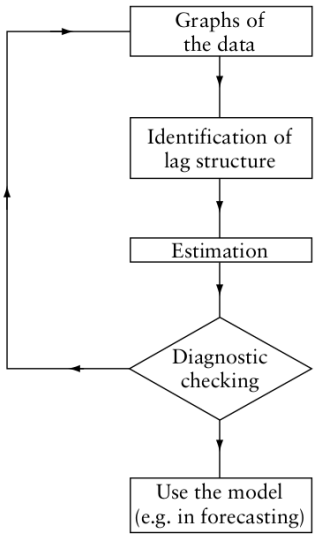
$$w_t = \begin{cases} log(y_t), \text{se } \lambda = 0 \\ \frac{y_t^{\lambda}-1}{\lambda}, \text{se } \lambda \neq 0 \end{cases} \tag{4.2}$$

O pacote feasts possui um modo, guerrero, que procura o melhor λ que minimiza a volatilidade da série.

4.4.4 Tratando Sazonalidade

Quando a série apresenta um componente sazonal precisamos mudar nosso ARIMA(p,d,q) para um SARIMA(p,d,q)X(P,D,Q)m. Ficamos assim com uma parte não sazonal e uma parte sazonal, ambas com diferentes parâmetros autorregressivos, de médias móveis e integração.

4.5 Metodologia de Box-Jenkins



Esta imagem dá uma ideia bem geral de como se cria e analisa um modelo de acordo com a metodologia de Box-Jenkins. Nesta metodologia podemos chegar a parâmetros diferentes dos modelos ou então ao mesmo modelo por caminhos diferentes. Em diversos livros é encontrado imagens parecidas com a mesma ideia de como fazer o passo a passo ao lidar com o modelo

4.5.1 Estimando o modelo

Como dito na imagem, é necessário estimar o modelo mas como se faz isso? Existem diversos passos que podemos tomar neste momento. Uma rotina padrão é analisar o critério de informação, seguido de uma análise dos erros e uma predição retirando uma quantidade de tamanho p das observações e usando $t - p$ para se fazer uma predição e comparar o predito e o realizado.

4.5.2 Critério de Informação

Critério de Informação	Fórmula	Detalhes
BIC(p,q)	$ln\hat{\sigma^2} + n\frac{lnT}{T}$	Parcimonioso e assintoticamente consistente
AIC(p,q)	$ln\hat{\sigma^2} + n\frac{2}{T}$	Melhor para pequenas amostras, tende a escolher modelos sobreparametrizados
HQ(p,q)	$ln\hat{\sigma^2} + n\frac{2}{T}lnlnT$	Assintoticamente consistente, menos forte que o BIC

4.5.3 Análise dos Resíduos

Após usar os gráficos de ACF e PACF para chegar aos parâmetros do modelo, queremos que os resíduos possuam apenas um ruído branco, ou seja, extraímos todas as informações relevantes para o modelo e temos apenas agora um erro i.i.d de média 0 e variância σ^2. Alguns testes que podemos realizar são:

- FAC e FACP nos resíduos
- Teste de Ljung-Box – ausência de autocorrelação entre os resíduos
- Teste para normalidade (Shapiro-Wilk, Jarque-Bera)

4.5.4 Predição

Um último passo é a parte de predição onde pegamos a série até um momento h no tempo, h<t, fazemos uma predição com o modelo que obtemos e analisamos como ele se comporta quando comparado a série original. Tanto em questão de valor quanto em questão de tendências, direção, etc.

4.5.5 Material útil

- Livro: Forecasting, Principles and Practice 3ª edição (fpp3). https://otexts.com/fpp3/non-seasonal-arima.html

- Livro: Econometria de séries temporais,2ª edição https://tinyurl.com/yp32769m
- Livro: Econometric methods with applications in business and economics https://tinyurl.com/em93ykpt
- Livro: Introduction to Econometrics with R https://www.econometrics-with-r.org/index.html
- Livro: Principles of Econometrics with R https://bookdown.org/ccolonescu/RPoE4/
- Livro: Análise de séries temporais https://tinyurl.com/2tmdhru9

Os tinyURL são apenas os links para a Amazon encurtados por motivos estéticos.

Chapter 5

Uso da Metodologia SARIMA

5.1 Princípios básicos

Ao começarmos a usar um modelo ARIMA (na verdade, qualquer modelo de série temporal) vale a pena termos a metodologia de Box-Jenkins em mente adaptado de forma condizente com o nosso modelo escolhido. Embora ela tenha sido apresentada há pouco, entremos em melhores detalhes do passo a passo:

1. Faça gráficos dos dados e transformações. Isto nos dá uma primeira impressão da série para sabermos os possíveis elementos presentes nela para nos guiar nos próximos passos.
2. Fazer as modificações para a série se tornar estacionária, corrigir sazonalidade e tendência.
3. Escolha de lags a partir dos testes da FAC e FACP para decidirmos os valores de (p, q). Isto é, decidirmos a ordem do processo AR(p) e do MA(q) no nosso modelo.
4. Com estes valores decididos, fazemos então a estimação dos parâmetros do modelo.
5. Teste de diagnóstico, neste momento usamos testes de autocorrelação nos resíduos (queremos que sejam um ruído branco ou o mais perto disso) e testamos para ver o quão bom é o nosso modelo para prever o futuro (em uma base temporal com N observações, retiramos t, usando o modelo nas N-t observações e comparamos o predito e o realizado).

6. Melhorar o modelo se necessário, caso ele não seja satisfatório, repetimos os passos anteriores para tentar achar um modelo melhor. Isto pode significar, inclusive, selecionarmos um modelo diferente do que tínhamos em mente inicialmente.
7. Após estes passos feitos e resultados e testes satisfatórios, podemos então usar o modelo.

5.2 Transformando uma série para ser passível de análise

5.2.1 Bibliotecas necessárias

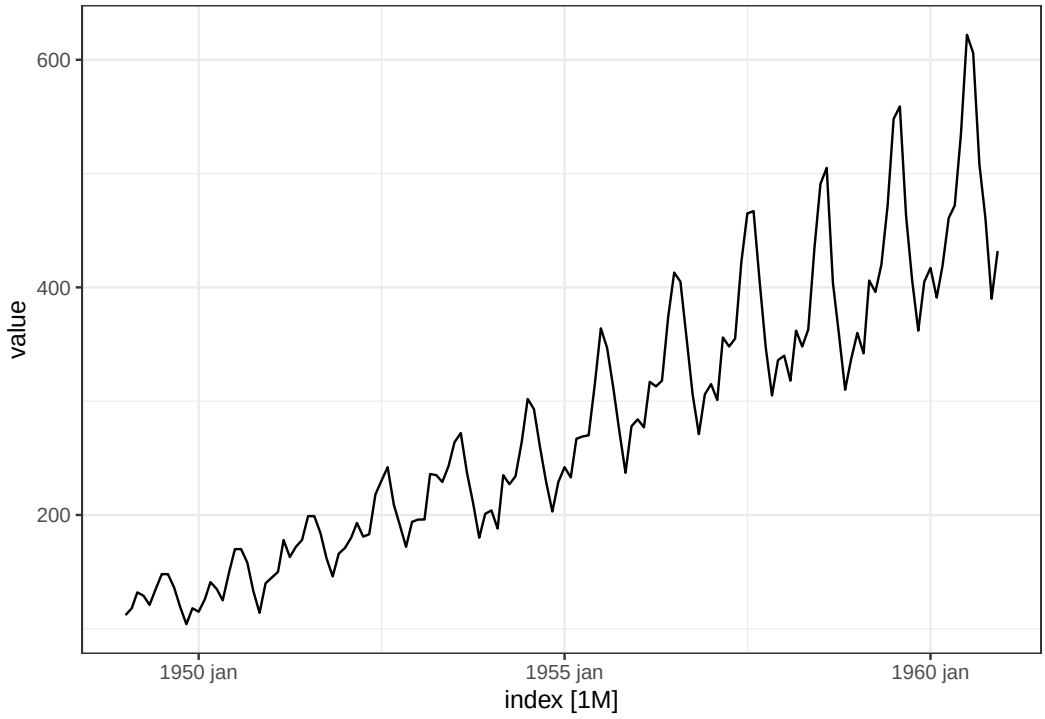
```
# Manipulação de dados
library(tidyverse)

# Manipulação de Séries Temporais
library(tsibble)

# Funções de Previsão
library(fable)

# Gráficos e Estatísticas de Séries Temporais
library(feasts)

# Séries Temporais Tidy
library(tsibbledata)

# Todos os itens acima e mais
library(fpp3)

#Pacote para testes de normalidade
library(normtest)

#Pacotes complementares de séries temporais
library(aTSA)
```

Usaremos a clássica série de aviões de Box & Jenkins para fazer uma análise com o modelo SARIMA que consiste dos totais de passageiros mensais entre 1949 a 1960. Um pequeno ?AirPassengers nos gera mais informações dentro do próprio R. Alguns dos passos dados aqui tem um fim mais didático sendo possível faze-los de maneira mais truncada.

Primeiramente, analisamos a série gráficamente e nos perguntamos quais elementos ela possui.

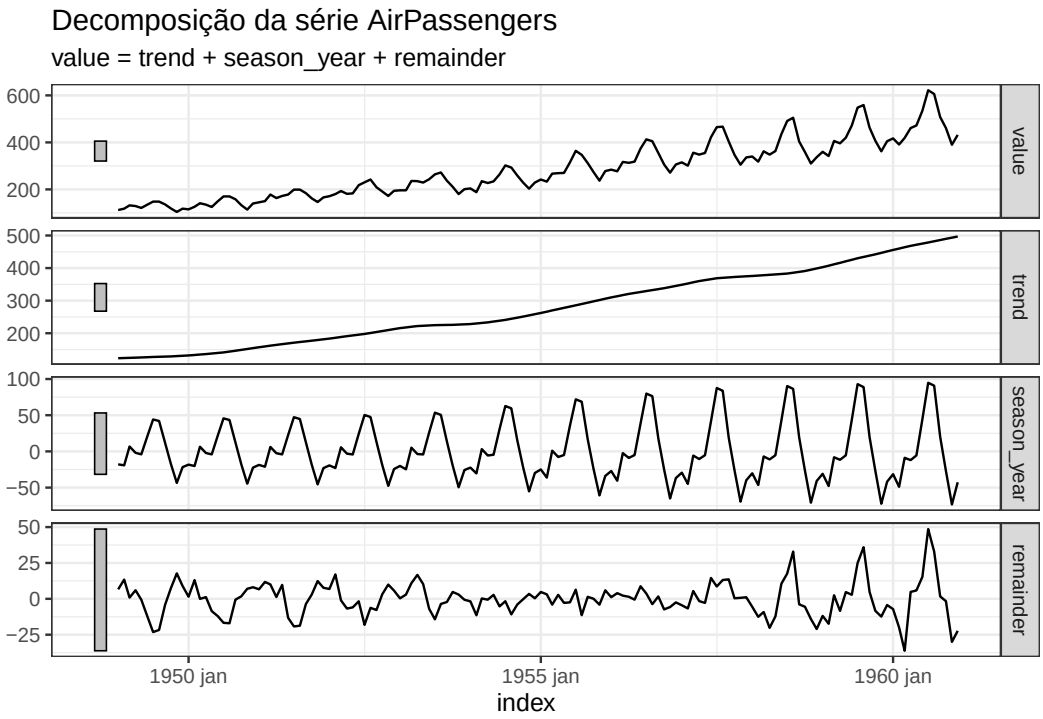
```
aviao <- AirPassengers %>% as_tsibble()
autoplot(aviao)
```

```
## Plot variable not specified, automatically selected `.vars = value`
```



Percebemos que ela possui tanto uma tendência positiva, como uma sazonalidade bem aparente e, também, uma variância não constante. Temos que tratar estes problemas antes de podermos modelar a série de acordo com um modelo SARIMA.

```
aviao_decomposto <- aviao %>%
  model(stl = STL(value))

components(aviao_decomposto) %>%
  autoplot()+
  ggtitle("Decomposição da série AirPassengers")
```



Estes elementos se tornam bem claros com a decomposição da série.

5.2.2 Corrigindo tendência

Em geral, quando há uma tendência numa série, tiramos a diferença entre y_t e y_{t-1} para corrigir a tendência, desta forma, temos então:

$$y'_t = y_t - y_{t-1}$$

Algumas vezes será necessário fazer diferenças em segunda ordem para chegarmos a série estacionária:

$$y''_t = y_t - 2y_{t-1} + y_{t-2}$$

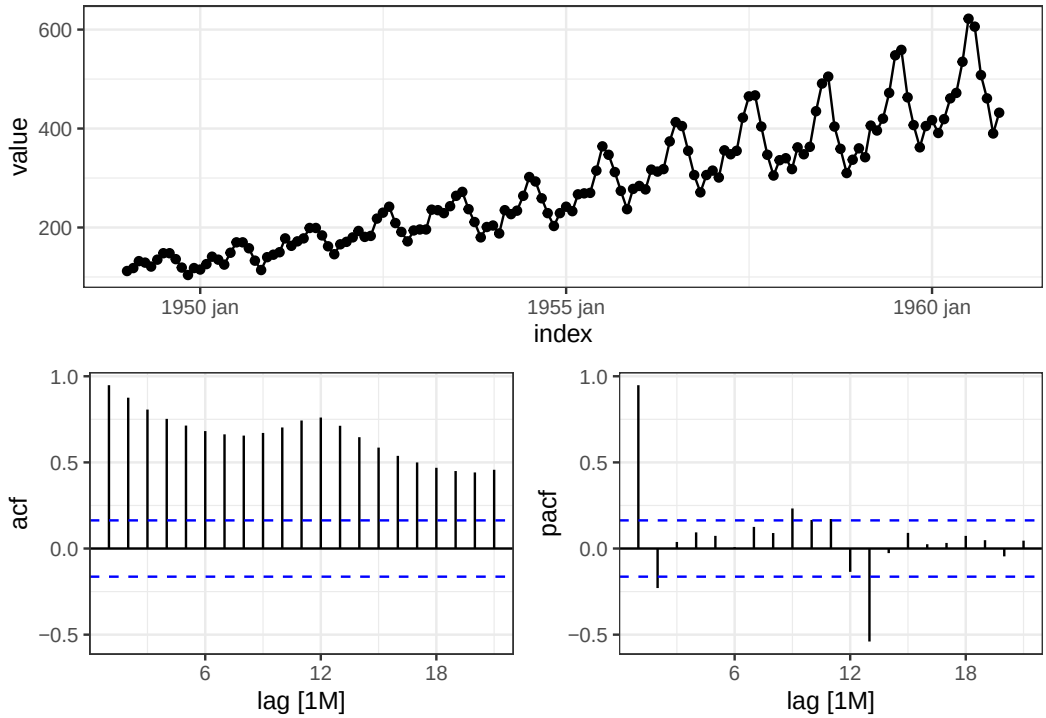
A maneira de testar a necessidade do uso de diferenças para controlar tendência é fazer o teste de raiz unitária. Existem vários testes para isto como o KSS, o ADF e o PP.

No KSS, a hipótese nula é de que a série é estacionária, queremos evidência que ela é falsa, logo, esperamos um p-valor pequeno.

Vamos, então, ao passo a passo de lidar com tendência. Lembrando que o que está sendo mostrado aqui é bem mais lento do que geralmente podemos fazer no R para fins didáticos.

Podemos, inicialmente, ver o gráfico da FAC da série. Repare que ele decai muito lentamente e tem uma ondulação que acontece a um intervalo constante, isso indica que a série possui tanto uma raiz unitária e não é estacionária quanto possui uma parte sazonal. Usamos esta função que é bem útil para vermos tanto a série, seu gráfico de FAC e FACP.

```
aviao %>%
  gg_tsdisplay(value, plot_type = "partial")
```



Feito isto, podemos então fazer o teste para raiz unitária, aqui usaremos o teste KSS.

```
aviao %>%
  features(value, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1      2.74        0.01
```

Temos um p-valor de 0,01 (provavelmente menor mas o R mostrará apenas no máximo 0,01) assim rejeitamos a H_0 = a série ser estacionária. Ou seja, estamos trabalhando com uma série temporal não-estacionária.

Dado a não estacionariedade da série, podemos agora usar um teste para sabermos o número de diferenças necessárias para ela.

```
aviao %>%
  features(value, unitroot_ndiffs)
```

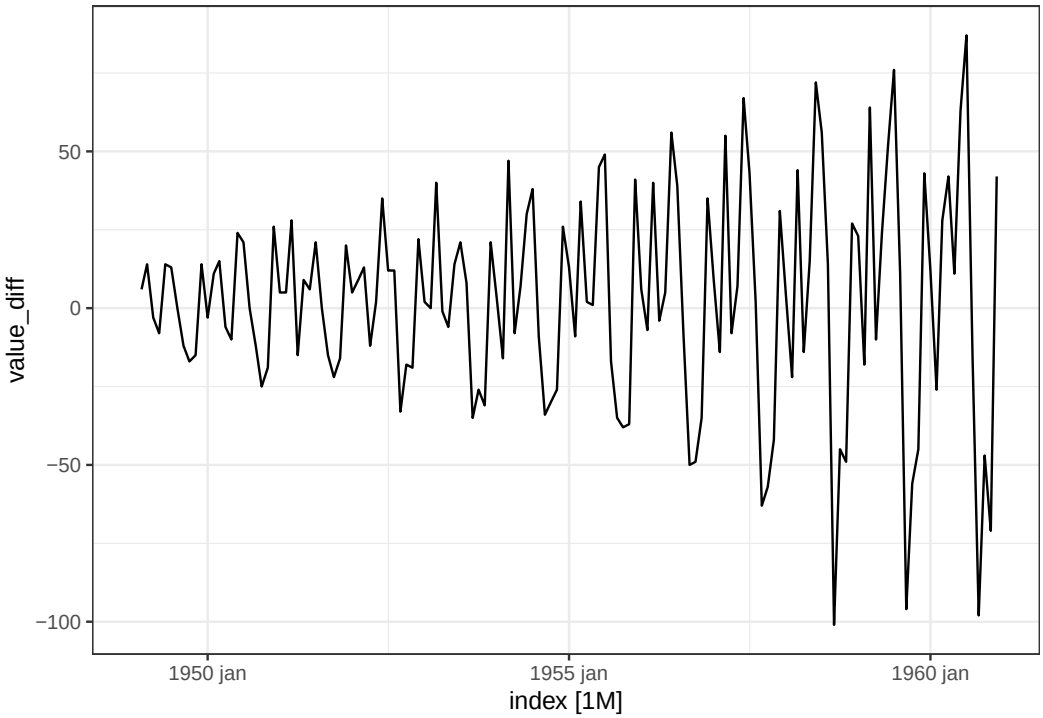
```
## # A tibble: 1 x 1
##   ndiffs
##    <int>
## 1      1
```

Que nos diz que apenas uma diferença é o suficiente para corrigirmos a tendência da série.

Vamos então, ver o gráfico da série com o uso de uma diferença:

```
aviao_diff <- aviao %>%
  mutate(value_diff = difference(value))

aviao_diff %>%
  autoplot(value_diff)
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```



É claro que a série com uma diferença não possui variância constante, assim temos que passar para nosso próximo problema da lista a ser resolvido.

- ☒ Tendência
- ☐ Variância não constante
- ☐ Sazonalidade

5.2.3 Corrigindo variância não constante

Como dito no capítulo passado, um método padrão a se usar é o de box-cox, dada pela equação: 4.2

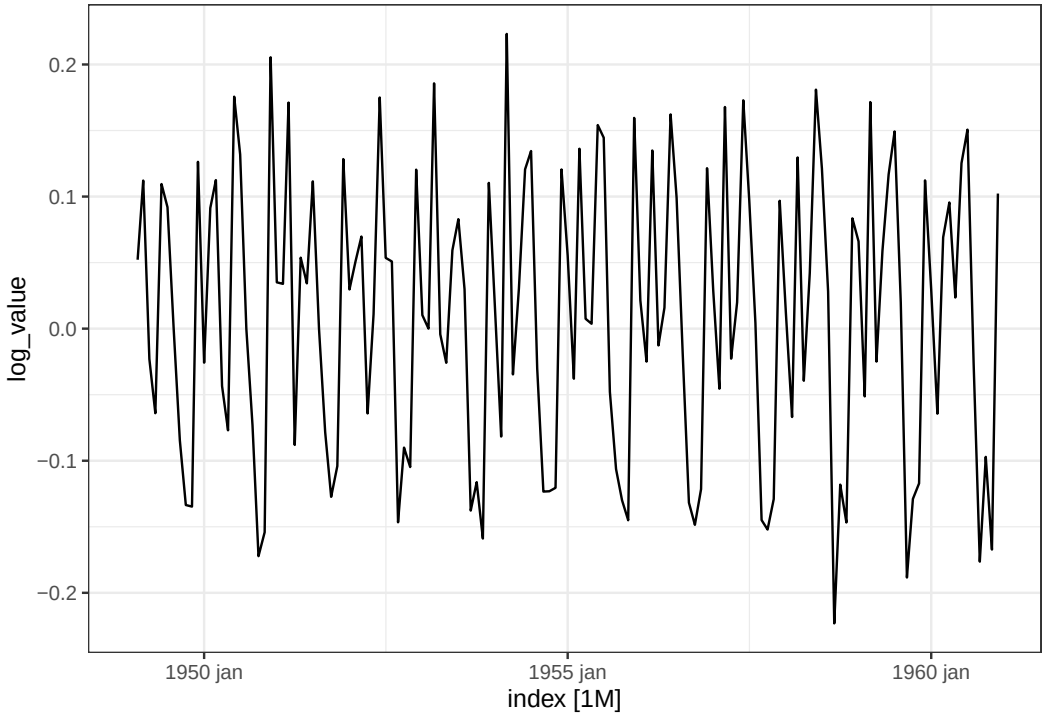
Essa transformação serve apenas para valores positivos. Essa transformação ajuda a termos como verdade as suposições que

$$y \sim N(X\beta, \sigma^2 I_n)$$

. Nem todos os dados podem ser transformados numa normal mas os valores usuais de λ levam a distribuições que respeitam algumas restrições de interesse até o quarto momento.

Seguindo esta ideia, então, usamos um log na série. Perceba que não funciona tirar a diferença e depois o log. Isso pode gerar vários erros nos valores (faça o teste!).

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```



Temos, agora, uma série com variância constante. Nosso segundo item de nossa pequena listinha também pode ser riscado!

- ☒ Tendência
- ☒ Variância não constante
- ☐ Sazonalidade

5.2.4 Corrigindo sazonalidade

Uma diferença sazonal é a diferença do valor dado em y_t e em t-m períodos anteriores, isto é, y_{t-m}, equivalentemente:

$$y'_t = y_t - y_{t-m}$$

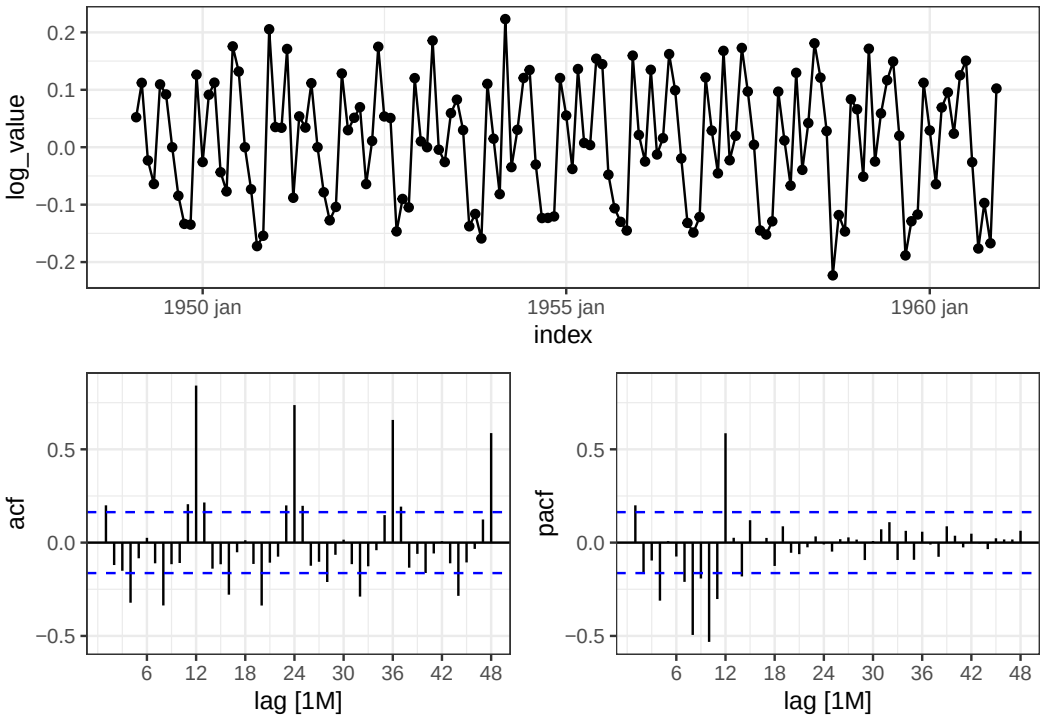
Assim, estamos corrigindo para um padrão que ocorre em um intervalo constante de tempo, por exemplo: alta de temperatura no verão, ou então um momento de contabilização de dados, etc.

Vamos agora, novamente com o auxílio do gg_tsdisplay olhar para a nossa série junto de seu FAC e FACP:

```
aviao_diff %>%
  gg_tsdisplay(log_value, plot_type = "partial", lag = 48)
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



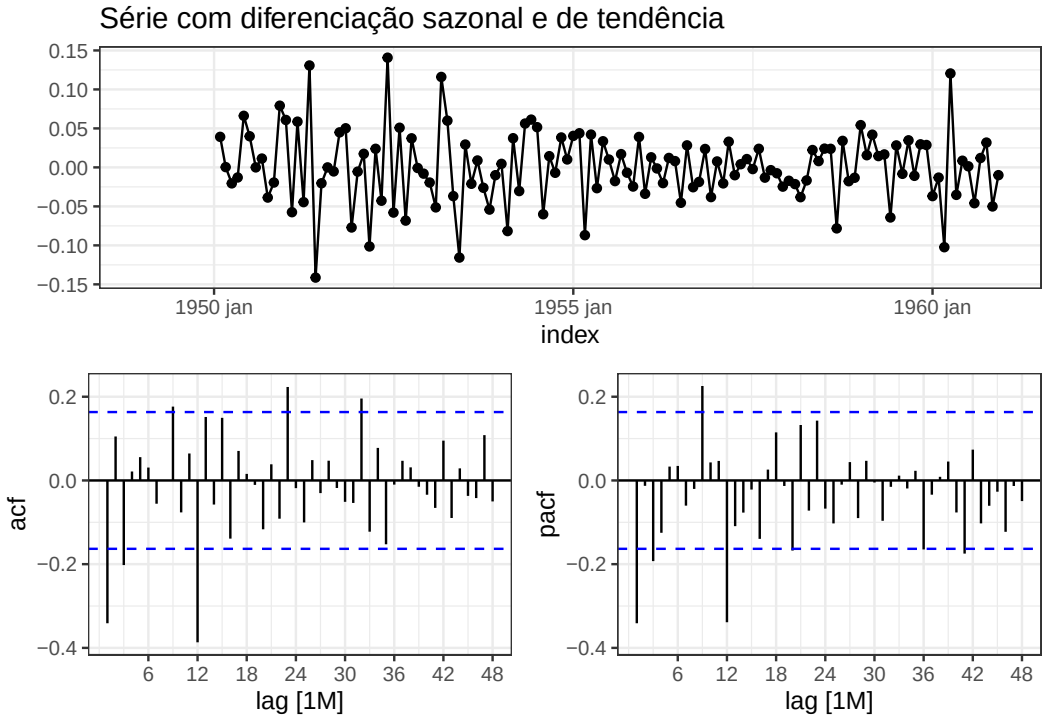
Vemos que essa série possui a cada 12 meses um pico na FAC, o que nos leva a pensar que temos uma sazonalidade anual de ordem MA(1). Mas não só

isso! Esta série está cheia de surpresas e sua FAC agora nos revela que sua parte sazonal também possui uma raiz unitária devido ao lento decaimento na FAC. Temos uma série com uma parte sazonal e não sazonal, ambas não estacionárias.

```
aviao_final <- aviao_diff %>%
  mutate(log_value_12 = difference(difference(log(value), 1),12))

aviao_final %>%
  gg_tsdisplay(log_value_12, plot_type = "partial", lag = 48) +
  labs(title = "Série com diferenciação sazonal e de tendência", y="")
```

```
## Warning: Removed 13 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 13 rows containing missing values (geom_point).
```



Atente pro código usado. Usamos uma diferença de lag 1 para lidar com a não estacionariedade da parte sazonal e um de lag 12 para corrigir a parte sazonal de periodicidade de um ano.

Podemos, fazer um teste de raiz unitária para testarmos quanto a existêcia desta após os tratamentos implementados.

```
aviao_final %>%
  features(log_value_12, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1    0.0844         0.1
```

O valor de teste é menor que o p-valor, assim não podemos rejeitar a hipótese nula de que a série que estamos trabalhando agora é estacionária.

Feito isto, *voilá* temos nossa série devidamente tratada para sua análise segundo um modelo do tipo SARIMA e a nossa pequena listinha está finalmente preenchida.

- ☒ Tendência
- ☒ Variância não constante
- ☒ Sazonalidade

Fica para o leitor o conselho de uma extensão: Uso dos testes PP e ADF do pacote aTSA e interpretar seus resultados, comparar com o teste KSS e comparar a diferença na implementação do código.

5.2.5 Qual correção devemos fazer primeiro?

Em geral, não existe uma ordem correta para o que fazer primeiro, em geral, as dicas encontradas na literatura é para começar com o ajuste sazonal pois algumas vezes só de corrigi-lo, a tendência também é corrigida. Caso a volatilidade seja bem aparente, começar usando o log ou uma correção da variância usando o método de box-cox funciona também.

Às vezes a nossa série tem uma tendência tão forte que nada mais é visível direito devido a ela, nestes casos, começar tratando a tendência é a nossa única opção. No fim, não há um tratamento que se deva começar sempre obrigatoriamente e, em teoria, o destino final será o mesmo independente do caminho (ou, pelo menos, os finais serão bem próximos um do outro).

5.3 Modelagem SARIMA

5.3.1 Escolha do modelo

Vamos, agora, para o momento mais aguardado desde o início deste capítulo: O uso do que analisamos ali atrás para fazer uma modelagem dos nossos queridos dados seguindo um SARIMA do tipo (p,d,q)X(P,D,Q)m. Podemos, ou manualmente colocar o modelo que queremos que seja usado para o ARIMA (neste caso SARIMA), ou deixar que nossa função faça o trabalho pra gente através de um algoritmo. Caso haja interesse, essa página explica a maneira como essa função trabalha caso deixada no automático: https://otexts.com/fpp3/arima-r.html (em inglês).

Agora, fazemos três modelos ao mesmo tempo, um baseado nas análises anteriores de diferenciação e leitura das FACs e FACPs. Perceba que isto está longe de ser algo objetivo, uma mesma pessoa pode ter leituras diferentes da mesma série em momentos diferentes no tempo, imagine então pessoas diferentes! Fica, inclusive, a dica de contestar este que lhos escreve e fazer esse processo com um modelo que é diferente e que faça mais sentido com os gráficos de FAC e FACP na opinião de vocês e comparar com os dados aqui.

```
fit <- aviao %>%
  model(
    arima111111 = ARIMA(value ~ pdq(1,1,1)+ PDQ(1,1,1)),
    arima011111 = ARIMA(value ~ pdq(0,1,1)+ PDQ(1,1,1)),
    auto = ARIMA(value, stepwise = FALSE, approx = FALSE)
  )
```

A função arima com stepwise e approx como falsos demora bem mais para ser feita, mas como estamos tratando apenas com uma única série temporal, esta demora a mais não será um grande problema. Caso fosse necessária a análise de muitas outras séries, é aconselhável deixar estes argumentos como verdadeiros.

5.3.2 Comparação entre os modelos

Como dito no capítulo 4, usamos os critérios de informação para a análise dos modelos, existem outras maneiras de analisar mas esta é a principal. Podemos

analisar todos de uma vez com as seguintes funções:

```
fit %>% pivot_longer(everything(), names_to = "Nome do Modelo",
                     values_to = "Ordem")
```

```
## # A mable: 3 x 2
## # Key:     Nome do Modelo [3]
##   `Nome do Modelo`                      Ordem
##   <chr>                                <model>
## 1 arima111111      <ARIMA(1,1,1)(1,1,1)[12]>
## 2 arima011111      <ARIMA(0,1,1)(1,1,1)[12]>
## 3 auto             <ARIMA(2,1,1)(0,1,0)[12]>
```

```
glance(fit) %>% arrange(AICc) %>% select(.model:BIC)
```

```
## # A tibble: 3 x 6
##   .model      sigma2 log_lik   AIC  AICc   BIC
##   <chr>        <dbl>   <dbl> <dbl> <dbl> <dbl>
## 1 auto          132.   -505. 1018. 1018. 1029.
## 2 arima011111   134.   -506. 1020. 1021. 1032.
## 3 arima111111   135.   -506. 1022. 1023. 1037.
```

Podemos olhar também os valores dos coeficientes, seus erros padrão e o quão significativos eles são e ir interagindo com o modelo na busca de um que seja melhor, retirando os coeficientes não significativos e mantendo os que são.

```
coef(fit)
```

```
## # A tibble: 10 x 6
##    .model      term  estimate std.error statistic  p.value
##    <chr>       <chr>    <dbl>     <dbl>     <dbl>    <dbl>
##  1 arima111111 ar1     -0.128    0.615     -0.208 8.36e- 1
##  2 arima111111 ma1     -0.214    0.640     -0.335 7.38e- 1
##  3 arima111111 sar1    -0.927    0.236     -3.93  1.35e- 4
##  4 arima111111 sma1     0.839    0.352      2.39  1.84e- 2
##  5 arima011111 ma1     -0.344    0.0880    -3.91  1.48e- 4
```

```
##  6 arima011111 sar1    -0.936    0.215     -4.36  2.60e- 5
##  7 arima011111 sma1     0.852    0.332      2.57  1.13e- 2
##  8 auto        ar1      0.596    0.0888     6.71  5.31e-10
##  9 auto        ar2      0.214    0.0880     2.44  1.62e- 2
## 10 auto        ma1     -0.982    0.0292   -33.6   2.74e-66
```

```
tidy(fit)
```

```
## # A tibble: 10 x 6
##    .model      term  estimate std.error statistic  p.value
##    <chr>       <chr>    <dbl>     <dbl>     <dbl>    <dbl>
##  1 arima111111 ar1     -0.128    0.615     -0.208 8.36e- 1
##  2 arima111111 ma1     -0.214    0.640     -0.335 7.38e- 1
##  3 arima111111 sar1    -0.927    0.236     -3.93  1.35e- 4
##  4 arima111111 sma1     0.839    0.352      2.39  1.84e- 2
##  5 arima011111 ma1     -0.344    0.0880    -3.91  1.48e- 4
##  6 arima011111 sar1    -0.936    0.215     -4.36  2.60e- 5
##  7 arima011111 sma1     0.852    0.332      2.57  1.13e- 2
##  8 auto        ar1      0.596    0.0888     6.71  5.31e-10
##  9 auto        ar2      0.214    0.0880     2.44  1.62e- 2
## 10 auto        ma1     -0.982    0.0292   -33.6   2.74e-66
```

Ambas funções retornam o mesmo resultado neste caso.

5.3.3 Análise dos Resíduos

A penúltima parte do nosso passo a passo consiste na análise dos resíduos que temos do nosso modelo. Como dito anteriormente, queremos que os resíduos sejam um ruído branco. Lembrando que como nosso erro não observável é assumido como sendo um ruído branco, se os resíduos também o são, significa que conseguimos extrair todas as informações relevantes da série e a colocamos na nossa equação explicativa.

O teste comumente achado na literatura para isso é o Ljung-Box, existem críticas a ele na literatura dado ao fato de que ele tende a ser enviesado assintoticamente quando analisamos uma série temporal e algumas pessoas indicam o uso do teste de Breusch–Godfrey no lugar. Como dentro dos pacotes fable,

feasts e tsibble que estamos usando não há a opção de usar o teste BG seguiremos a análise com o Ljung-Box mesmo sabendo que assintoticamente ele é enviesado. Em Hayashi, aqui https://stats.stackexchange.com/questions/6455/how-many-lags-to-use-in-the-ljung-box-test-of-a-time-series e aqui https://stats.stackexchange.com/questions/148004/testing-for-autocorrelation-ljung-box-versus-breusch-godfrey temos uma explicação mais detalhada do assunto. (ambos os links são em inglês e pelo meu conhecimento, o Hayashi não tem versão traduzida).

```
augment(fit) %>% features(.innov, ljung_box, lag=24, dof=4)
```

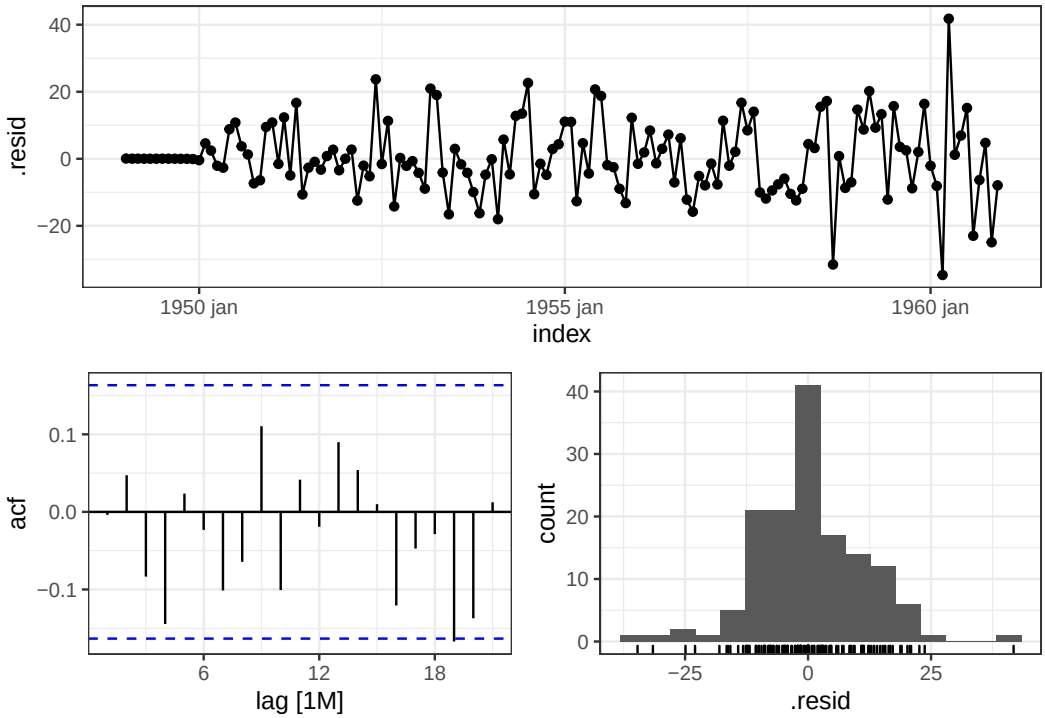
```
## # A tibble: 3 x 3
##   .model      lb_stat lb_pvalue
##   <chr>         <dbl>     <dbl>
## 1 arima011111    39.3   0.00616
## 2 arima111111    38.8   0.00701
## 3 auto           37.8   0.00941
```

Usamos um lag de 24 devido à sazonalidade da série, dof são os graus de liberdade, dados por $lag - (p + q + P + Q)$. Neste caso usei o valor 4 pois foi feito o teste de vários modelos com parâmetros diferentes ao mesmo tempo.

O ideal seria para cada um deles fazer o teste com a quantidade correta no dof. De toda maneira,não rejeitamos a hipótese nula de não autocorrelação dos resíduos.

Podemos, também, fazer o teste de normalidade dos resíduos, primeiramente é possível visualizá-los para fazer uma análise visual dos gráficos:

```
fit %>%
  select(arima011111) %>%
  gg_tsresiduals()
```



Junto disso, podemos retirar os resíduos e transformá-los em um vetor e utilizar o teste Jarque-Bera de normalidade nos resíduos:

```
residuos <-  fit %>%
  select(arima011111) %>%
  augment() %>%
  select(.innov)

normtest::ajb.norm.test(residuos$.innov)
```

```
##
##  Adjusted Jarque-Bera test for normality
##
## data:  residuos$.innov
## AJB = 14.86, p-value = 0.0105
```

Como não há uma função de teste de normalidade nos pacotes fable e feasts, fazemos essa pequena volta para transformar os valores em vetores e depois

então realizarmos o teste. Um procedimento semelhante pode ser feito para se realizar o teste de Breusch–Godfrey, também conhecido como teste LM.

Finalizada esta parte de análise de adequação do modelo à nossa série temporal, analisando os resíduos, vendo os critérios de informação, analisando os coeficientes e se são estatisticamente significantes, podemos passar para nossa última etapa (finalmente) da análise do modelo: A predição.

5.4 Predição

5.4.1 Overfit

É comum, no contesto de modelagens estatísticas, que surja um modelo dentre os que cogitamos como possíveis modelos explicativos da série que se sobressai em todos ou grande parte dos testes estatísticos que analisamos mas ao colocarmos para predizer o futuro, o modelo se mostra sem o brilho que outrora apresentara.

Este é um caso de overfit, o modelo se adequou muito bem apenas para os nossos dados usados para treinamento do modelo mas quando colocado a prova real vê-se que seus poderes preditivos são pífios. Para evitar que isto ocorra com um uso real do modelo, é comum utilizarmos nosso modelo escolhido na série temporal utilizando $T - h$ observações temporais.

Estas h observações retiradas das observações totais são usadas então para a comparação entre o que foi observado e o que o modelo previu. Isto é um processo bem útil e essa ideia pode ser utilizada em vários outros modelos, como, por exemplo, modelos de predição de Machine Learning e afins. Explicada a ideia do que faremos agora, sem mais delongas, vamos a nossa etapa final da implementação de um modelo SARIMA(p,d,q)X(P,D,Q)m.

5.4.2 Predição da Série Temporal

Primeiramente, removemos a nossa série deixando apenas as observações até novembro de 1959 e a denominamos como série de treino. Deixamos, assim, 13 observações de fora na nossa base de dados de treino.

```
treino <- aviao %>%
  filter_index(.~ "1959 nov")
```

Após feito isto, usamos o nosso modelo na nossa série de treino e visualizamos seus coeficientes e suas estatísticas.

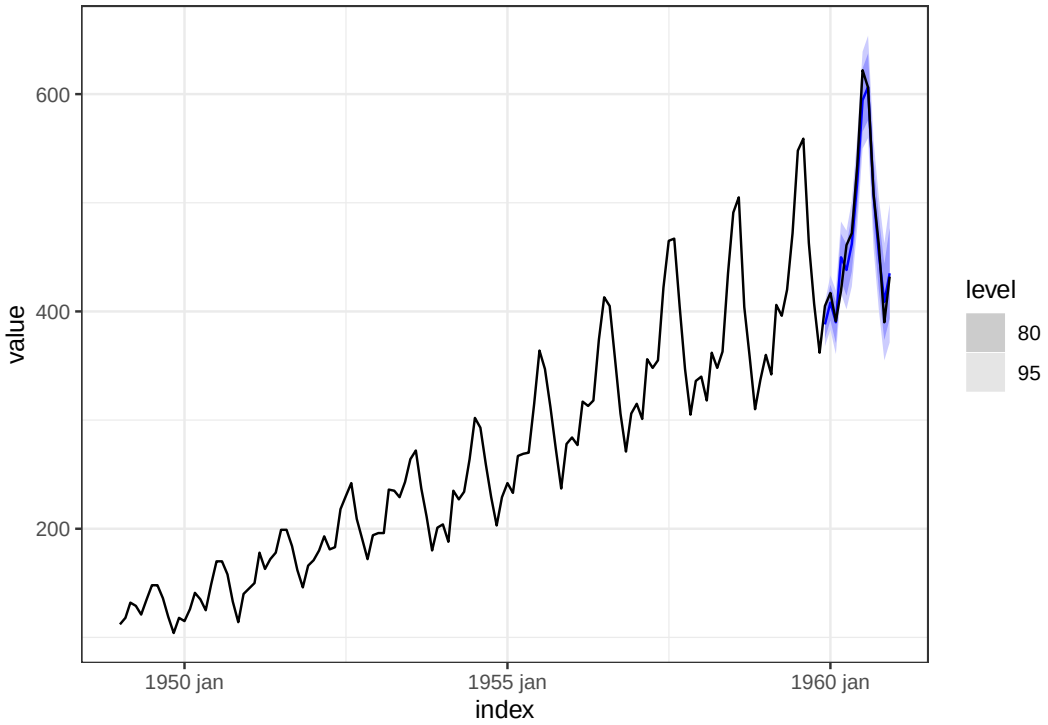
```
aviao_fit <- treino%>%
  model(ARIMA(value ~ pdq(0,1,1)+ PDQ(1,1,1)))

report(aviao_fit)
```

```
## Series: value
## Model: ARIMA(0,1,1)(1,1,1)[12]
##
## Coefficients:
##           ma1     sar1    sma1
##       -0.2479  -0.9977  0.9759
## s.e.   0.0894   0.0216  0.1127
##
## sigma^2 estimated as 100.8:  log likelihood=-441.72
## AIC=891.43   AICc=891.78   BIC=902.51
```

Agora, podemos usar a função forecast do fabletools com h=13 para ver o que o modelo previa. Junto disto, colocamos o gráfico do predito junto do gráfico do que se realizou. Podemos assim, comparar o quão bom esse modelo previu os valores seguintes nessa série teste.

```
aviao_fit %>%
  fabletools::forecast(h=13) %>%
  autoplot+
  autolayer(aviao)
```

```
## Plot variable not specified, automatically selected `.vars = value`
```



Por fim, analisamos os valores de erro entre nossa predição e o que realmente se realizou.

```
aviao_fit %>% accuracy()
```

```
## # A tibble: 1 x 10
##   .model          .type    ME  RMSE   MAE    MPE  MAPE  MASE RMSSE    ACF1
##   <chr>           <chr> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 ARIMA(value ~ ~ Trai~ 0.467  9.41  7.15 0.0882  2.75 0.237 0.276 -0.0156
```

Lista de siglas para as medidas de erro

- ME – Margem de Erro
- RMSE – Raiz Quadrada do Erro Médio
- MAE – Erro Médio Absoluto
- MPE – Erro Percentual Médio
- MAPE – Erro Percentual Médio Absoluto
- MASE – Erro Médio Absoluto de Escala

- RMSSE – Raiz Quadrada do Erro Médio Ajustada
- ACF1 – AutoCorrelação dos Erros no lag 1

5.4.3 Um pequeno adendo

Utilizamos como terminologia a palavra "teste" como sendo o conjunto de observações menos uma quantidade h. E "treino" como sendo as h observações restantes em que comparamos a performance do modelo com valores observados, isto é, comparamos os valores reais que ocorreram com os valores que nosso modelo previu.

Existe uma ampla discussão em qual o tamanho ideal que h deve ter, dentro da literatura, as regras mais comuns são a 70/30 - 70% da série usada como treino e 30% como teste, 80/20. Mas não existe uma regra do tamanho correto a se tomar.

5.5 Exercícios para treino

1. Exercício proposto durante as explicações do passo a passo pedindo o uso dos testes de ADF e PP na série temporal e interpretação de seus resultados.

2. Modelar, com base no passo a passo a série que foi apresentada(AirPassagers), procurando fazer a leitura da ACF e PACF para escolha dos valores para um SARIMA(p,d,q)X(P,D,Q)12 - ao invés de apenas confiar na função auto do modelo.

3. Fazer a leitura e modelagem da série ST.RDS fazendo cada um dos passos apresentados nesta seção.

4. Escolher uma base de dados do pacote tsibbledata e fazer a modelagem apresentada aqui.

5. Utilização do teste de Breusch–Godfrey nos resíduos de um modelo dos modelos e sua comparação com o teste de Ljung-Box. Pode ser do modelo que você fez.