Análise de Séries Temporais em R

Um curso introdutório

Autor e Organizador

Pedro Costa Ferreira

Autores

Daiane Marcolino de Mattos

Ingrid Christyne Luquett de Oliveira

Lucas Farias Lima

Victor Eduardo Leite de Almeida Duca

Conteúdo

Apresentação

Ao longo dos últimos dez anos tenho dedicado boa parte dos meus estudos a Séries Temporais: a) tratei sobre o tema na minha dissertação de mestrado e na minha tese de doutorado; b) Prestei consultoria sobre o assunto em importantes empresas como Light S.A, Operador Nacional do Sistema Elétrico (ONS) e Duratex S.A.; e c) contribui para a geração de importantes resultados em projetos de Pesquisa e Desenvolvimento desenvolvidos para empresas do setor elétrico brasileiro.

Atualmente, além de ministrar a disciplina de Econometria de Séries Temporais e ser revisor de importantes *Journals* como *Journal of Applied Statistics*, coordeno o Núcleo de Métodos Estatísticos e Computacionais (FGV/IBRE) e, basicamente, nosso trabalho envolve Séries Temporais, Big Data e Data Science.

A experiência adquirida, em empresas e academicamente, ao longo desses anos, mostraram-me a importância e a necessidade de se trabalhar com softwares robustos e flexíveis, mas, sobretudo, livres. Tendo tais premissas como princípio, há três anos resolvemos adotar o R como nossa ferramenta de trabalho. E, com muita convicção, acredito que tomamos a decisão correta.

O R, além de ser robusto, flexível, livre, trabalhar muito bem com grandes bases de dados e oferecer excelentes pacotes de séries temporais, surpreendeu-nos positivamente em atividades como *webscraping*, *datamining* e, mais recentemente, com o Shiny (desenvolvido pela RStudio Inc) e todas suas possibilidades na criação de páginas na web, por exemplo. Aliás, recomendo a visita a nossa página sobre séries temporais, toda desenvolvida em Shiny - https://pedroferreira.shinyapps.io/timeseries.

É nesse contexto que esse livro se insere. Ele é a junção de diversos trabalhos que desenvolvemos nos últimos anos no NMEC (FGV/IBRE) e de minhas aulas de Séries Temporais. O livro está muito interessante e, sem dúvida, reflete a competência e a dedicação da equipe do NMEC (FGV/IBRE).

O livro está organizado da seguinte forma: na primeira parte, **Introdução ao R, Estatística**

descritiva e Gráficos, que compreende dois capítulos, fazemos uma breve introdução do software R e abordamos tópicos como a instalação do software, a criação de funções e conceitos básicos de estatística. Além disso, ensinamos como criar importantes gráficos, como boxplot e gráfico de dispersão. Por fim, ao final desta parte introdutória, apresentamos alguns exercícios com respostas no Github (https://github.com/pedrocostaferreira/Analise-de-Series-Temporais-em-R) que ajudarão a fixar os conceitos aprendidos nesses capítulos.

A próxima seção, entitulada **Análise de Séries Temporais: Modelos Univariados** engloba os capítulos 3 ao 6. No capítulo 3 o leitor aprenderá a fazer ajustes e previsões em séries estacionárias, séries com a presença das componentes de tendência e de sazonalidade ou as duas em conjunto. Além disso, o leitor terá acesso a diversas bases de dados disponíveis gratuitamente para fins didáticos. Os modelos apresentados neste capítulo estão separados em NAIVE, Média Móvel, Suavização Exponencial Simples, Suavização Exponencial de Holt e Suavização Exponencial de Holt-Winters. Para cada modelo será discutido a sua construção seguido de exemplos ilustrativos.

No capítulo 4 abordamos o problema da não estacionariedade, quando as propriedades estatísticas não são invariantes por translações no tempo. Começamos o capítulo mostrando as principais fontes de não-estacionariedade. Em seguida, apresentamos o método adequado para tratar de cada uma. São apresentados testes de raiz unitária, além de testes para quebras estruturais, a fim de verificar a necessidade de diferenciação da série.

O capítulo 5 é dedicado à apresentação do modelo SARIMA. Para tal, fez-se uso da série temporal de vendas de passagens aéreas, mais conhecida como *Airline*. Trata-se de uma série temporal mensal que registra o total de passageiros internacionais (em milhares) da linha aérea (Pan Am) no período de janeiro de 1949 a dezembro 1960, nos EUA. Ao longo desse capítulo, discutiremos as características dessa série e os passos para modelá-la utilizando o software R. Ao ler esse capítulo, pretende-se que o leitor esteja apto a modelar uma série temporal "não complexa", seguindo a proposta de Box & Jenkins, utilizando o software R.

Finalizando a parte 2, o capítulo 6 tem como objetivo analisar a sazonalidade presente em séries temporais. Aborda-se como identificar tal componente e removê-la para que as análises a respeito da

trajetória da série temporal sejam feitas adequadamente. O método utilizado para extrair a componente sazonal, isto é, dessazonalisar, é o X-13ARIMA-SEATS desenvolvido pelo U. S. Census Bureau. Este é um programa mundialmente conhecido. Você vai aprender a utilizá-lo no R e a avaliar a qualidade de um ajuste sazonal a partir das ferramentas que o mesmo dispõe.

A parte 3, **Análise de Séries Temporais: Modelos Multivariados**, aborda os principais modelos de séries temporais que levam em conta outras variáveis para explicar a variável de interesse.

O capítulo 7 apresenta como acrescentar outras variáveis aos modelos desenvolvidos por Box & Jenkins. A metodologia é conhecida como Box & Jenkins com Função de Transferência e as variáveis inseridas, geralmente, possuem relação causal com a variável resposta, o que pode aperfeiçoar a capacidade preditiva do modelo. Apresentamos o método originalmente desenvolvido para definir a defasagem das variáveis auxiliares que serão inseridas no modelo e como diagnosticar o modelo final.

O capítulo 8 explora a aplicação do modelo clássico de regressão em variáveis observadas no tempo, abordando as implicações sobre seus pressupostos. Discutimos, em particular, a presença de correlação serial nos resíduos e apresentamos maneiras de contorná-la, tanto pela inclusão de estrutura autorregressiva nos erros, quanto pelo uso de modelos autorregressivos com defasagens distribuídas (ADL). Discutimos os problemas associados a regressões espúrias, expondo o modelo de correção de erro como possível solução.

Neste capítulo 9, apresentamos os modelos VAR e VECM. Começamos com a definição do modelo e as limitações que levaram à sua criação. Em seguida, tratamos de sua estabilidade, que traça um paralelo com o problema de não-estacionariedade para modelos univariados. Uma vez estimado, apresentamos algumas ferramentas de análise estrutural e de previsão. Em seguida, abordamos os modelos estruturais e a cointegração no contexto multivariado, que leva à metodologia de Johansen para inferência da quantidade e estimação dos vetores de cointegração.

Esperamos que aproveitem e gostem do conteúdo apresentado no livro e que ao final dessa viagem pelo "Mundo das Séries de Tempo"você saiba os passos e os cuidados necessários para uma boa modelagem e previsão.

Autores

Pedro Guilherme Costa Ferreira é Doutor em Engenharia Elétrica - (Decision Support Methods) e Mestre em Economia. Co-autor dos livros "Planejamento da Operação de Sistemas Hidrotérmicos no Brasil"e "Análise de Séries Temporais em R: um curso introdutório". É o primeiro pesquisador da América Latina a ser recomendado pela empresa RStudio Inc. Atuou em projetos de Pesquisa e Desenvolvimento (P&D) no setor elétrico nas empresas Light S.A. (e.g. estudo de contingências judiciais), Cemig S.A, Duke Energy S.A, entre outras. Ministrou cursos de estatística e séries temporais na PUC-Rio e IBMEC e em empresas como o Operador Nacional do Setor Elétrico (ONS), Petrobras e CPFL S.A. Atualmente é professor de Econometria de Séries Temporais e Estatística (IBMEC-RJ), cientista chefe do Núcleo de Métodos Estatísticos e Computacionais (FGV|IBRE) e coordenador dos cursos Economia Descomplicada (FGV|IDE) e Big Data e Data Science (FGV|IDE). É também revisor de importantes journals, como Energy Policy e Journal of Applied Statistics. Principais estudos são em modelos Econométricos, Setor Elétrico, Incerteza Econômica, Preços, R software e Business Cycle.

Daiane Marcolino de Mattos é mestranda em Engenharia Elétrica (Métodos de Apoio à Decisão) pela PUC-Rio e graduada em Ciências Estatísticas pela Escola Nacional de Ciências Estatísticas (ENCE/IBGE). Atualmente trabalha como pesquisadora no Núcleo de Métodos Estatísticos e Computacionais (NMEC) no Instituto Brasileiro de Economia (FGV/IBRE) com foco na melhoria do ajuste sazonal das séries de Sondagens do IBRE e na extração da tendência da inflação; utiliza o software R como principal programa estatístico nas tarefas diárias desde as análises até a apresentação dos resultados.

Ingrid Christyne Luquett de Oliveira é Mestre e graduada em Estatística pela Universidade Federal do Rio de Janeiro (UFRJ). Atualmente pesquisadora do Núcleo de Métodos Estatísticos e Computacionais (NMEC) no Instituto Brasileiro de Economia (FGV/IBRE) com foco em modelagem de séries econômicas, utilizando o R como principal ferramenta computacional. Principais estudos em modelagem de séries temporais, estatística espacial e inferência bayesiana.

Lucas Farias Lima é mestrando em Matemática pela PUC-Rio e graduado em Ciências Econômicas pelo IBMEC-MG, onde atuou como assistente de pesquisa - função que exerceu também na Fundação João Pinheiro, do governo do estado de Minas Gerais. Atualmente é pesquisador do Núcleo de Métodos Estatísticos e Computacionais (NMEC) no Instituto Brasileiro de Economia (FGV/IBRE). Possui experiência em econometria de séries tmeporais e modelos estatísticos multivariados.

Victor Eduardo Leite de Almeida Duca Graduou-se em Estatística pela Universidade Federal Fluminense (UFF). Mestre em Engenharia Elétrica (Métodos de Apoio à Decisão) pela PUC-Rio. Atualmente é aluno de doutorado em Estatística na Universidade Federal do Rio de Janeiro (UFRJ). Trabalhou como professor substituto no Departamento de Estatística da Universidade Federal Fluminense (UFF). Tem experiência na área de Probabilidade e Estatística atuando principalmente em Séries Temporais e Simulação.

Análise de Séries Temporais em R

Um curso introdutório

Parte I

Introdução ao R, Estatística Descritiva + Gráficos

Introdução ao R

Daiane Marcolino de Mattos

Pedro Costa Ferreira

Introdução: O que é o R e RStudio?

Nesse primeiro capítulo vamos introduzir os *softwares* R e RStudio. Esses *softwares* serão utilizados durante todo o livro e irão proporcionar uma formar muito agradável e prática de aprender Estatística.

O R é uma linguagem e um ambiente para cálculos estatísticos e elaboração de gráficos. Foi criado originalmente por Ross Ihaka e por Robert Gentleman no Departamento de Estatística da Universidade de Auckland, Nova Zelândia. Hoje conta com um esforço colaborativo de pessoas de vários locais do mundo. O R é gratuito e livre sob uma licença referida no Cran. Funciona nas versões de sistemas operacionais: Unix, Windows e Macintosh e está disponível em http://cran.r-project.org/. Nesse site também é possível encontrar dicas para a utilização do programa e um fórum onde você pode tirar suas dúvidas fazendo perguntas ou analisando dúvidas de outros usuários já respondidas.

O programa é bastante utilizado por profissionais que usam ferramentas estatísticas no seu dia-a-dia, como estatísticos e *data miners*. Permite realizar operações matemáticas simples e métodos mais complexos como, por exemplo, análises e manipulações de dados, análises de séries temporais, modelagens lineares e não lineares, testes estatísticos, técnicas de amostragem, elaboração de gráficos, simulações de redes neurais, entre outras coisas.

O RStudio é um ambiente de desenvolvimento integrado e tem uma ampla gama de recursos que permitem aumentar a produtividade do usuário no uso do R. O RStudio também é executado em todas as principais plataformas. Para utilizá-lo, é preciso que o usuário já tenha instalado previamente o R de versão 2.11.1 ou superior.

Como baixar o R e o RStudio

Após acessar o site http://cran.r-project.org/, escolha o sistema operacional de seu computador: Linux, (Mac) OS X ou Windows e clique na opção *base* ou *install R for first time*. Em seguida, baixe o arquivo em sua versão mais atualizada. Após a instalação do programa, inicie o R em seu computador. Quando o programa estiver pronto para o uso, aparecerá o símbolo > em vermelho no console e você já poderá escrever sua programação.

A fim de aprimorar sua produtividade e facilitar o uso do R, indicamos que você baixe o RStudio acessando o site http://www.rstudio.com/ e clicando na opção *Download RStudio*. Também indicamos o canal da FGV no Youtube para você aprender um pouco mais sobre o R por meio

dos vídeos produzidos pelo Núcleo de Métodos Estatísticos e Computacionais. Acesse o canal em https://www.youtube.com/fgv e pesquise por Estatística com R.

Iniciando o R

Primeiros Passos

• Operações Aritméticas

O R pode ser utilizado como uma calculadora de grande capacidade para realizar diferentes operações com números, vetores ou matrizes. Veja a Tabela 1.1 a seguir com alguns dos principais operadores e funções.

Operação	**Comando no R**
Soma	`+`
Subtração	`-`
Multiplicação	`*`
Divisão	`/`
Quociente da divisão	`%/%`
Resto da divisão	`%%`
Potenciação	`^` ou `**`
Raiz quadrada	`sqrt(x)`
Logaritmo	`log(x, base)`
Exponencial	`exp(x)`
Fatorial	`factorial(x)`
Combinação	`choose(n, x)`

Tabela 1.1: Operações matemáticas no R

```
> 5+2              # soma: +
> 5-2              # subtração: -
> 5*2              # multiplicação: *
> 5/2              # divisão: /
> 5%/%2            # quociente da divisão: %/%
> 5%%2             # resto da divisão: %%
> 5^2              # potência: ^
> 5**2             # potência: **
> sqrt(5)          # raiz quadrada
> exp(5)           # exponencial
> log(5, base = 2) # logaritmo
> factorial(5)     # fatorial
> choose(5,2)      # combinação de 5, 2 a 2
```

Dica: use o símbolo # para fazer comentários junto a sua programação. Esse símbolo não influencia na execução dos comandos e é muito útil para organizar o seu código.

• **Criando objetos**

Um objeto pode ser criado com a operação de atribuição `<-`. Criando um objeto, você pode guardar um resultado para utilizá-lo novamente em seguida, por exemplo:

```
> x <- 5  # objeto x recebe valor 5
> x
[1] 5
```

Importante: o R é uma linguagem case sensitive, ou seja, diferencia letras maiúsculas de minúsculas. Nomes de objetos precisam iniciar com letras.

Você pode criar um objeto e, ao mesmo tempo, observar o resultado sem ter que digitar o nome do objeto novamente. Para isso coloque parênteses no início e no final da linha de comando como é feito a seguir.

```
> (a <- 5 + 27^2)  # cria o objeto 'a' e o mostra em seguida
[1] 734
```

A função `ls()` mostra os objetos criados por você e que estão, no momento, armazenados na memória do R.

```
> ls()  # lista os objetos existentes
[1] "a" "x"
```

Caso você queira excluir objetos, use a função `remove()` ou `rm()`. Exemplo:

```
> a <- 1
> b <- 6
> c <- 9  # cria os objetos a, b, c
> ls()  # lista os objetos existentes
[1] "a" "b" "c"
> rm(b) # remove o objeto b
> ls()  # lista os objetos existentes
[1] "a" "c"
```

• **Criando vetores**

Vetores são objetos que armazenam mais de um valor. O R usa a função `c()` para criar vetores.

```
> vetor1 <- c(1,2,5,6,7,25,0,-3) # vetor com números
> vetor2 <- c("verde","amarelo") # vetor com textos
> vetor1
[1]  1  2  5  6  7 25  0 -3
> vetor2
```

```
[1] "verde"   "amarelo"
```

Com o R é possível gerar sequências e números aleatórios. Abaixo seguem exemplos de como gerar diversas sequências.

```
> # sequência de 1 a 5
> (a <- 1:5)
 [1]  1  2  3  4  5
> # sequência de 1 a 10 de 2 em 2
> (b <- seq(from = 1, to = 10, by = 2))
 [1]  1  3  5  7  9
> # sequência de tamanho 9
> (c <- seq(length = 9, from = 1, to = 5))
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> rep(1,5)        # repete o nº 1 5 vezes
 [1] 1 1 1 1 1
> sequence(3)      # gera a sequência de 1 até 3
[1] 1 2 3
> sequence(c(3,5)) # executa duas sequências ao mesmo tempo, neste caso,
123 e 12345
[1] 1 2 3 1 2 3 4 5
```

• Criando matrizes

Para criar matrizes utilize a função `matrix()`. Aprenderemos como utilizar essa função com os exemplos a seguir. No primeiro exemplo cria-se uma matriz quadrada de ordem 3 composta apenas por zeros.

```
> # criando uma matriz de zeros 3x3
> A <- matrix(0, nrow = 3, ncol = 3)
> A
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
[3,]    0    0    0
```

Veja agora como preencher a matriz `A` de duas maneiras: por linha e por coluna.

```
> # preenchendo a matriz por linha
> A[1,] <- c(1,2,3)
> A[2,] <- c(7,8,9)
> A[3,] <- c(-10,-4,-6)
> A
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    7    8    9
[3,]  -10   -4   -6
```

```
> # preenchendo a matriz por coluna
> A[,1] <- c(1,2,3)
> A[,2] <- c(7,8,9)
> A[,3] <- c(-10,-4,-6)
> A
     [,1] [,2] [,3]
[1,]    1    7  -10
[2,]    2    8   -4
[3,]    3    9   -6
```

A seguir, em vez de criar uma matriz de zeros e depois preenchê-la, criaremos uma matriz já totalmente preenchida.

```
> # criando uma matriz preenchida
> B <- matrix(c(1,2,3,8,9,10), nrow = 2, ncol = 3, byrow = T)
> B
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    8    9   10
```

Caso o argumento lógico `byrow` fosse alterado para `FALSE`, a matriz seria preechida por coluna. Veja como a matriz `B` ficaria com essa alteração a seguir.

```
> # criando uma matriz preenchida
> (B <- matrix(c(1,2,3,8,9,10), nrow = 2, ncol = 3, byrow = F))
     [,1] [,2] [,3]
[1,]    1    3    9
[2,]    2    8   10
```

• Criando data frame

Data frames são objetos usados para guardar tabelas de dados. É a classe de objeto mais utilizada, principalmente por ser esse o formato que o R lê para as tabelas importadas de outros *softwares*. Para criar um *data frame* no R, utilize a função `data.frame()`. Vamos criar um *data frame* com duas colunas (Nome e Idade) e com três elementos em cada uma delas.

```
> # criando um data frame
> tabela <- data.frame(Nome = c("André", "João", "Tiago"),
+                      Idade = c(20, 58, 22))
> tabela
   Nome Idade
1 André    20
2  João    58
3 Tiago    22
```

Caso você queira criar uma nova coluna, basta apenas digitar o nome do objeto seguido de $ e o nome da sua nova coluna. Como exemplo vamos criar uma coluna chamada Salário.

```
> # adicionando uma nova coluna ao data frame
> tabela$Salário <- c(1200, 5700, 800)
> tabela
   Nome Idade Salário
1 André    20    1200
2  João    58    5700
3 Tiago    22     800
```

Veja outra opção de como adicionar mais valores ao *data frame* usando as funções `cbind` e `rbind` que agregam por coluna e linha, respectivamente.

```
> tabela <- data.frame(Nome = c("André", "João", "Tiago"),
                       Idade = c(20, 58, 22))
> tabela <- cbind(tabela, data.frame(Salario = c(1200, 5700, 800)))
> tabela
   Nome Idade Salario
1 André    20    1200
2  João    58    5700
3 Tiago    22     800

> tabela <- rbind(tabela, data.frame(Nome = "Joana", Idade = 30,
                                     Salario = 3100))
> tabela
   Nome Idade Salario
1 André    20    1200
2  João    58    5700
3 Tiago    22     800
4 Joana    30    3100
```

Em um *data frame*, os dados costumam vir em dois tipos de formatos: amplo (*wide*) ou longo (*long*). No formato amplo, há uma linha para cada observação e cada coluna representa uma variável. Já no formato longo, há mais de uma linha para cada observação e uma coluna para várias variáveis. Há uma função no R que permite a conversão entre esses dois formatos: `reshape()`. Os principais argumentos da função `reshape()` são:

- `data`: um objeto de classe *data frame*;
- `direction`: formato da base de dados que podem ser `"wide"` (amplo) ou `"long"` (longo);
- `idvar`: nome(s) da(s) variável(is) que identifica(m) o individuo. Podem estar no formato longo ou amplo;
- `timevar`: variável ou variáveis no formato longo que diferenciam vários registros do mesmo individuo;
- `v.names`: nome da variável no formato longo que corresponde às várias variáveis do formato amplo;

- `varying`: nomes dos conjuntos de variáveis no formato amplo que correspondem ao `timevar` do formato longo;
- `time`: valores do récem criado `timevar`.

Como exemplo de uso da função `reshape()`, vamos manipular os dados de `Indometh`: um *data frame* de 66 linhas e 3 colunas de dados sobre a farmacocinética de indometacina já disponível no R. As seis primeiras linhas de `Indometh` são:

```
  Subject time conc
1       1 0.25 1.50
2       1 0.50 0.94
3       1 0.75 0.78
4       1 1.00 0.48
5       1 1.25 0.37
6       1 2.00 0.19
```

Como nossos dados já se encontram no formato longo, vamos convertê-lo para o formato amplo. Para isso, teríamos que escolher a coluna *Subject* como a variável de identificações (`idvar`); uma ou mais variáveis de medição que diferenciam vários registros do mesmo `idvar` (`timevar`); uma variável que deseja mudar do formato longo para o amplo (`v.names`); e o formato que queremos os dados (`direction`), que nesse caso é `wide` (amplo).

```
> wide <- reshape(Indometh, timevar = "time", idvar = "Subject",
                  direction = "wide")
> wide
       Subject conc.0.25 conc.0.5 conc.0.75 conc.1 conc.1.25 conc.2 conc.3
1            1      1.50     0.94      0.78   0.48      0.37   0.19   0.12
12           2      2.03     1.63      0.71   0.70      0.64   0.36   0.32
23           3      2.72     1.49      1.16   0.80      0.80   0.39   0.22
34           4      1.85     1.39      1.02   0.89      0.59   0.40   0.16
45           5      2.05     1.04      0.81   0.39      0.30   0.23   0.13
56           6      2.31     1.44      1.03   0.84      0.64   0.42   0.24

   conc.4 conc.5 conc.6 conc.8
1    0.11   0.08   0.07   0.05
12   0.20   0.25   0.12   0.08
23   0.12   0.11   0.08   0.08
34   0.11   0.10   0.07   0.07
45   0.11   0.08   0.10   0.06
56   0.17   0.13   0.10   0.09
```

Agora temos apenas uma linha para cada indivíduo, uma coluna dos Subject e outras colunas para cada tempo. Mas se fosse ao contrário, ou seja, se já tivessemos os dados no formato amplo e quissesemos no formato longo, o procedimento seria similar. Como exemplo, converteremos os dados

anteriores no formato amplo para o formato longo.

```
> long <- reshape(wide, varying = names(wide[2:12]), idvar = "Subject",
+                 time = c("0.25", "0.5", "0.75","1", "1.25", "2", "3", "4",
+                          "5", "6", "8"),
+                 timevar = "time", v.names = "conc", new.row.names = 1:66,
+                 direction = "long")
> head(long)
  Subject time conc
1       1 0.25 1.50
2       2 0.25 2.03
3       3 0.25 2.72
4       4 0.25 1.85
5       5 0.25 2.05
6       6 0.25 2.31
```

• Estruturas de condição

As estruturas de condição têm por objetivo executar uma ação se determinada condição for satisfeita. A seguir são listadas três estruturas de repetições bastante úteis.

1. `if`: Executa diversos comandos se a condição inicial for verdadeira.

```
if(condição){
  comandos
}
```

2. `if...else`: Executa diversos comandos se a condição inicial for verdadeira, caso contrário executa outros comandos.

```
if(condição){
  comandos
}else{
  comandos
}
```

3. `ifelse`: Bem parecida com a anterior, porém executa apenas uma linha de comando se a condição inicial for verdadeira, caso contrário executa outro comando.

```
ifelse(condição, comando se verdadeira, comando se falsa)
```

Vejamos um exemplo: suponha dois números `a` e `b` e informe o maior dentre os dois.

```
> # Dois objetos numéricos
> a <- 2
> b <- 5
> # Estrutura de condição
> if(a > b){
+   cat("O maior é: ",a)
+ }else{
+   cat("O maior é: ",b)
+ }
O maior é:  5
```

Importante: A função `cat()` é útil para informar frases e objetos.

• Estruturas de repetição

As estruturas de repetição são úteis para repetir uma série de operações. Podem ser utilizadas, por exemplo, com vetores, matrizes e tabelas e também com estruturas de condição. Para que o leitor entenda a utilidade da estrutura, vamos exemplificar. Inicialmente, criaremos um *data frame* com informações sobre o salário de 10 pessoas.

```
> # data frame de salários de 10 indivíduos de uma empresa
> dados <- data.frame(individuo = c("A", "B", "C", "D", "E", "F", "G", "H",
+                                   "I", "J"),
+                     salario = c(1250, 800, 8500, 900, 2010, 2200, 3600,
+                                 7580, 5100, 9400))
> dados
   individuo salario
1          A    1250
2          B     800
3          C    8500
4          D     900
5          E    2010
6          F    2200
7          G    3600
8          H    7580
9          I    5100
10         J    9400
```

Uma função de estrutura de repetição muito popular é a função `for()`. Esta percorre os elementos (previamente definidos) de um objeto e executa os comandos escolhidos.

```
for(i in a:b){
   comandos
}
```

Para ficar claro o uso da função `for()`, suponha que se deseja saber quantas pessoas da empresa

recebem menos de R$ 3000,00. Vamos criar um objeto que conte o número de pessoas que satisfazem essa condição. Lembre-se de que é necessário criar o objeto antes de começar a estrutura de repetição.

```
> # criando um contador
> contador <- 0
> # estrutura for
> for(i in 1:10){
+   if(dados[i, 2] < 3000){contador <- contador + 1}
+ }
> contador   # número de funcionário que recebem menos de R$ 3000,00

[1] 5
```

Inicialmente, o contador começa recebendo o valor zero. Se a condição de salário for satisfeita (menor do que 3000), o contador recebe `+1`. Isso será feito para todas as `i` linhas do *data frame*. Por fim, será retornado o valor final do contador. Veja a seguir mais um exemplo.

Suponha também que seja necessário criar a soma acumulada dos salários dos indivíduos da empresa e armazená-la numa nova coluna do *data frame*. Então, será criada uma nova coluna no *data frame* e a primeira linha da nova coluna será o próprio salário do indivíduo A; a segunda linha será a soma do salário do indivíduo A e do indivíduo B; a terceira linha será a soma dos salários dos indivíduos A, B e C. Assim, seguindo o raciocínio, a última linha é a soma de todos os salários. Veja como fazer isso a seguir.

```
> # nova coluna para preencher com a soma acumulada
> dados$salario_acum <- 0
> dados

   individuo salario salario_acum
1          A    1250            0
2          B     800            0
3          C    8500            0
4          D     900            0
5          E    2010            0
6          F    2200            0
7          G    3600            0
8          H    7580            0
9          I    5100            0
10         J    9400            0

> # o salário acumulado 1 é o primeiro salário
> dados[1, 3] <- dados[1, 2]
> dados
```

```
   individuo salario salario_acum
1          A    1250         1250
2          B     800            0
3          C    8500            0
4          D     900            0
5          E    2010            0
6          F    2200            0
7          G    3600            0
8          H    7580            0
9          I    5100            0
10         J    9400            0
> # estrutura de repetição para as somas acumuladas
> for (i in 2:10){
+       dados[i, 3] <- dados[i-1, 3] + dados[i, 2]
+ }
> dados
   individuo salario salario_acum
1          A    1250         1250
2          B     800         2050
3          C    8500        10550
4          D     900        11450
5          E    2010        13460
6          F    2200        15660
7          G    3600        19260
8          H    7580        26840
9          I    5100        31940
10         J    9400        41340
```

• **Como criar funções**

O R permite que você crie funções que atendam suas necessidades. Para criar uma função utiliza-se a função `function()`.

```
nomefuncao <- function(argumento1, ..., argumento n){
  comandos
}
```

Vamos ver um exemplo. Suponha que se queira calcular a área de um triângulo dado sua base e altura, como na Figura 1.1. Sabe-se que a área é dada por: $(base \times altura)/2$. Dessa forma, uma função que calcule a área de um triângulo necessita de dois argumentos: um que indique a base e outro que indique a altura do triângulo.

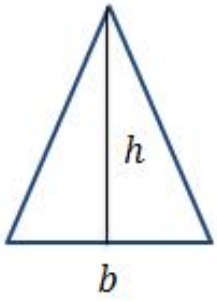


Figura 1.1: Exemplo de triângulo

```
> # Criando uma função que calcula a área de um triângulo
> area <- function(b, h){
+   A <- (b * h)/2
+   return(A)
+ }
> # Área de um triângulo de base 5 e altura 3
> area(5,3)
[1] 7.5
> area(h = 3, b = 5)
[1] 7.5
```

• **Outras funções interessantes**

Segue uma lista de funções que podem ser úteis para você.

Ação	**Comando**
sair do programa	`q()`
arredonda o número para o inteiro mais próximo	`round(x)`
arredonda o número para n casas decimais	`round(x,n)`
arredonda o número para cima	`ceiling(x)`
trunca o número	`trunc(x)`
arredonda o número para n casas decimais	`signif(x,n)`
produtório	`prod(x)`

Tabela 1.2: Algumas funções úteis

Manipulando diretório de trabalho

O local padrão onde são importados e exportados os arquivos e os comandos digitados no console do R é na pasta Documentos. Você pode vê-lo digitando o comando abaixo.

```
> getwd() # visualizar diretório de trabalho
```

Podemos criar um diterório com o comando `dir.create()`, por exemplo `meu_dir`, e depois alterar com o comando `setwd()` para informar ao R que é nesse local que iremos trabalhar. Exemplo:

```
> dir.create("C:/Users/Descktop/meu_dir") # criando o diretório "meu_dir"
> setwd("C:/Users/Descktop/meu_dir") # altera o diretório para "meu_dir"
```

Esse comando altera o diretório para a pasta `meu_dir` que se encontra na área de trabalho dentro da unidade `C`. Sugerimos que mude seu diretório para o local onde estão os arquivos que você precisará utilizar durante sua programação em R. Você também pode alterar o diretório utilizando as teclas de atalho `Ctrl+Shift+H`.

Importante: ao definir caminho do diretório, deve-se usar ou uma barra simples "/" ou duas contra-barras "\\".

Salvando o histórico

Os comandos digitados anteriormente no console podem ser salvos em um novo arquivo de texto caso você queira usá-los posteriormente.

Use as funções `savehistory()` e `loadhistory()` para salvar e carregar o histórico, respectivamente. Esse arquivo é salvo automaticamente no diretório de trabalho do R.

```
> # salvar
> savehistory(file = "nomedoarquivo.txt")
> # carregar
> loadhistory(file = "nomedoarquivo.txt")
```

Caso você queira salvar o histórico em outro local, especifique o caminho do arquivo, por exemplo:

```
> # salvar histórico em outro local
> savehistory(file = "V:/novo_dir/nomedoarquivo.txt")
```

Como criar um script

O script no RStudio funciona como um editor de texto. É bastante prático e facilita a alteração de sua programação. Veja a Figura 1.2 e descubra como criar um *script*.

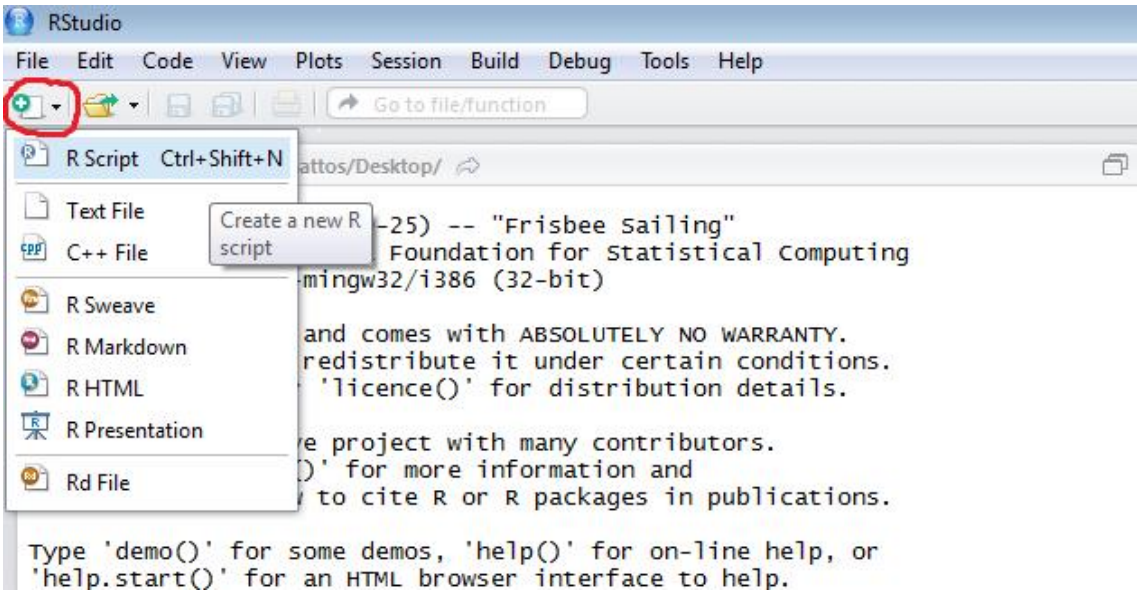


Figura 1.2: Como criar um *script*

Veja um exemplo de script na figura 1.3.

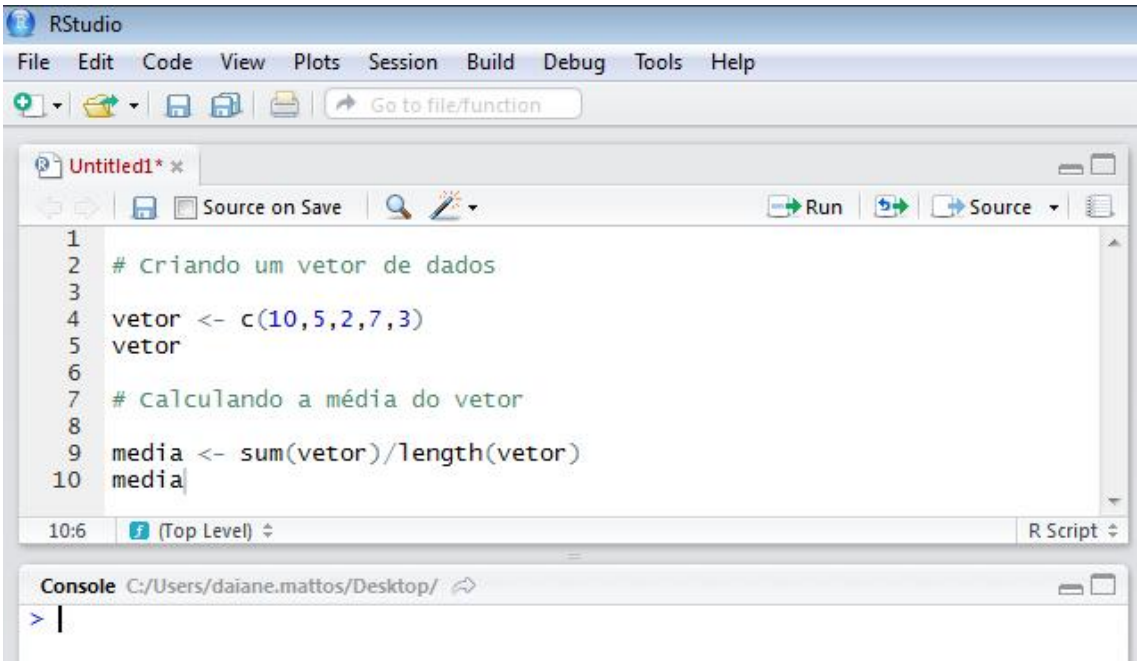


Figura 1.3: Exemplo de *script*

Para enviar as linhas de comando digitadas no *script* para o console, selecione as linhas desejadas e clique em ***Run*** ou apenas use o atalho `Ctrl+Enter`.

Como instalar e carregar pacotes (*packages*)

Todas as funções do R pertencem a algum pacote. As funções que apresentamos até agora pertencem aos pacotes que foram instalados e carregados quando instalamos o R no computador. Para utilizar outras funções e tornar sua experiência mais rica no R é necessário instalar e carregar pacotes. Há duas formas simples de instalar pacotes. A primeira é utilizando a função `install.packages()`:

```
install.packages("nome_do_pacote")
```

A segunda é utilizando a opção *Install Packages* que está na aba *Packages* no canto direito inferior na tela do RStudio (Figura 1.4). Após clicar nesta opção, digite o nome do pacote.

Depois de instalar o pacote, é necessário carregá-lo para que você consiga utilizar as funções que este disponibiliza. Para carregá-lo, você pode selecionar o pacote instalado anteriormente na aba Packages do RStudio (através do "*checklist*") ou, também pode usar as funções `require()` e `library()`.

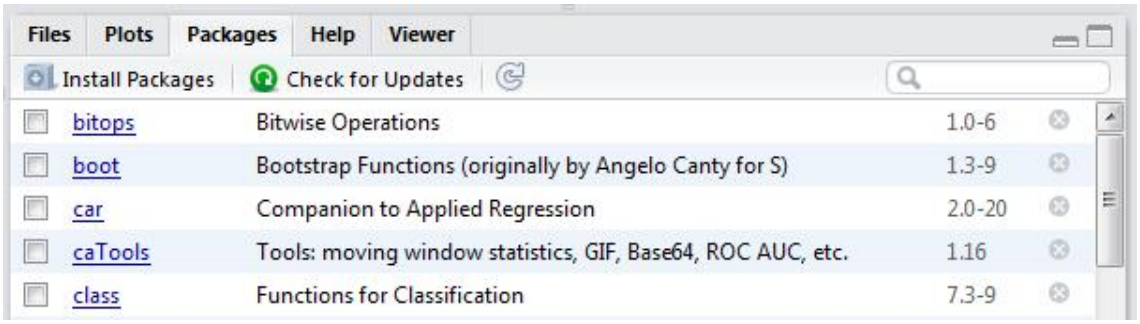


Figura 1.4: Como instalar pacotes

`require(nome_do_pacote)`

`library(nome_do_pacote)` ou `library("nome_do_pacote")`

Como exemplo, vamos instalar um pacote que permite que arquivos de extensão .xlsx sejam importados para o R. O nome do pacote é `xlsx` (Dragulescu (2014)).

```
> # O pacote xlsx permite a importação de arquivos .xlsx
> install.packages("xlsx")  # instalar pacote
> require(xlsx)             # carregar pacote
```

Após a instalação e o carregamento do pacote, as funções disponibilizadas por ele podem ser usadas.

Dica: digite `library()` e `search()` para visualizar, respectivamente, os pacotes que já estão instalados e os pacotes carregados no momento.

Nota: Em alguns ambientes o pacote `xlsx` (Dragulescu (2014)) apresenta alguns problemas, como alternativa, utilize o pacote `readxl` (Wickham (2016)) ou o pacote `openxlsx` (Walker (2015)).

Leitura de dados

Ler dados nos formatos de texto, tais como .txt e .csv dentre outros, é uma função indispensável para qualquer *software* de estatística. A função `read.table()` lê arquivos no formato tabela em um `data.frame`. Cada coluna tem uma classe própria automaticamente determinada pela função `read.table()`. Os principais argumentos são:

- `file`: nome do arquivo (com caminho se não estiver no diretório de trabalho);
- `header`: determina se o arquivo tem cabeçalho ou não (*default:* `TRUE`). No caso de `FALSE`, a função cria nomes para as colunas;
- `sep`: separador da coluna (*default*: tabulação);

- `dec`: separador decimal (*default*: ponto).

Embora a função `read.table()` seja provavelmente a função mais utilizada para ler dados, existem outras versões dessa com pequenas variações nos *defaults* que podem ser úteis. Veja: `read.csv()`, `read.csv2()`, `read.delim()` e `read.delim2()`. Como exemplo, vamos ler um arquivo de extensão .txt, chamado meu_arquivo que se encontra no diretório de trabalho.

```
> dados <- read.table("meu_arquivo.txt", header = T, sep = "\t", dec = ".")
```

Para visualizar o arquivo anterior no R, digite `dados`. A seguir algumas funções interessantes que ajudam a visualizar como é composto seu arquivo de dados. Considere `x` como seu arquivo de dados.

- `head(x)`: mostra as 6 primeiras linhas de um *data frame* ou as 6 primeiras posições de um vetor;
- `tail(x)`: mostra as 6 últimas linhas de um *data frame* ou as 6 últimas posições de um vetor;
- `View(x)`: visualização dos dados no formato de uma tabela (mais elegante);
- `edit(x)`: além de visualizar, é possível editar de uma maneira mais simples seu arquivo de dados.

Importando arquivos de extensão .xlsx

Para importar planilhas do Excel no formato .xlsx é necessário instalar um novo pacote uma vez que `read.table()` e sua família não estão aptos para realizar tal função.

O pacote `xlsx` (Dragulescu (2014)) permite importar e exportar arquivos de extensão .xlsx. Para importar utilize a função `read.xlsx()`. É importante saber que esse pacote não importa arquivos de formato `.xls` tampouco exporta. Os principais argumentos da função `read.xlsx()` são:

- `file`: nome do arquivo (com caminho se não estiver no diretório de trabalho);
- `header`: determina se o arquivo tem cabeçalho ou não (*default:* `TRUE`). No caso de `FALSE`, a função cria nomes para as colunas;
- `sheetIndex`: índice da aba que deve ser lida;
- `sheetName`: nome da aba que dever ser lida (*default:* `NULL`);
- `rowIndex`: índices das linhas que devem ser lidas (*default*: `NULL`);
- `colIndex`: índices das colunas que devem ser lidas (*default*: `NULL`).

Exemplo: vamos importar a primeira planilha de um arquivo de extensão .xlsx.

```
> tabela1 <- read.xlsx("arquivo.xlsx", sheetName = "Plan1")
```

Salvar dados

Escrever dados nos formatos de texto .txt, .csv, entre outros, é outra função indispensável para um *software* de estatística. A função `write.table()` salva arquivos de formatos formatos .txt e .csv no diretório padrão ou especificado. Os principais argumentos dessa função são:

- `x`: matriz ou *data frame* que será salva;
- `file`: nome do arquivo (com caminho se não estiver no diretório de trabalho);
- `append`: se `TRUE`, a saída será anexada a um arquivo já existente. Se `FALSE`, se houver algum arquivo com o mesmo nome, este será substituído pelo novo.
- `na`: caracter que indicará valores faltantes nos dados.

Embora a função `write.table()` seja provavelmente a função mais utilizada para salvar dados, existem outras versões dessa com pequenas variações nos *defaults* que podem ser úteis.. Veja: `write.csv()` e `write.csv2()`.

Exemplo: Vamos criar um objeto *data frame* e salvá-lo como um arquivo de formato .txt.

```
> # criando um data frame de idades
> idades <- data.frame(Nome = c("Tabi", "Gabi", "Andressa", "Vanessa",
+                               "Natália", "Natasha"),
+ Idades = c(36, 25, 33, 48, 21, 24))
> idades
      Nome Idades
1     Tabi     36
2     Gabi     25
3 Andressa     33
4  Vanessa     48
5  Natália     21
6  Natasha     24
> # salvando o objeto idades em formato .txt
> write.table(idades, "idades.txt",  quote = F, sep = "\t", row.names = F,
+             col.names = T)
```

Descobrindo Informações sobre o Objeto

Neste livro utilizaremos como base de dados a Pesquisa de Orçamentos Familiares do IBGE, a POF (IBGE, 2015a). O propósito da POF é traçar um perfil dos hábitos das famílias brasileiras. Para isso, são coletados dados gerais sobre os domicílios e sobre as famílias e pessoas contidas neles. São informações referentes a gastos, consumos e rendimentos. Por exemplo, condições do entorno e acesso

Código da UF	UF	Capital
29	BA	Salvador
31	MG	Belo Horizonte
33	RJ	Rio de Janeiro
35	SP	Sao Paulo
43	RS	Porto Alegre
53	DF	Brasília

Tabela 1.3: Capitais contidas no nosso arquivo da POF

à coleta de lixo em relação aos domicílios; idade, religião e posição de ocupação em relação às pessoas; e despesas com saúde e viagens em relação à família.

A pesquisa abrange as zonas urbanas e rurais de todo o território brasileiro e tem duração de 12 meses, recolhendo informações durante todas as épocas do ano para um estudo mais eficaz. Através dela é possível conhecer os bens consumidos e serviços utilizados pelas famílias, o que pode ser útil para criação de políticas públicas que auxiliam no combate à pobreza e na melhoria da saúde.

Nosso arquivo não contém todos os dados da pesquisa. Somente estão incluídas nele 6 unidades de federação (UF) que são representadas, apenas, por suas respectivas capitais (Tabela 1.3).

Vamos ler a tabela da POF no R, que está em formato .csv, e guardá-la num objeto chamado dados.

```
> # Ler a base de dados da POF
> dados <- read.csv2("POF_capitais.csv")
```

Algumas funções são utilizadas para descobrir mais informações acerca do objeto de dados. Exemplo: tipo (texto, numérico, lógico...), tamanho (número de linhas ou colunas), entre outras coisas. Apresentaremos algumas funções a seguir.

• **Função** `str()`

A função `str()` mostra um resumo estrutural de um objeto: classe, número de linhas e colunas, nome das colunas e o tipo (inteiro, numérico, etc). Vamos ver um resumo estrutural para a base de dados da POF.

```
> # resumo estrutural para a base de dados da POF
> str(dados)
```

```
'data.frame':        3504 obs. of  31 variables:
 $ TIPO.DE.REGISTRO                              : int  1 1 1 1 1 1 1 1 1 1 ...
 $ CÓDIGO.DA.UF                                  : int  33 33 33 33 33 33 33 33 33 33 ...
 $ NÚMERO.SEQUENCIAL                             : int  1 1 1 1 1 1 1 1 1 1 ...
 $ DV.DO.SEQUENCIAL                              : int  9 9 9 9 9 9 9 9 9 9 ...
 $ NÚMERO.DO.DOMICÍLIO                           : int  1 10 11 12 13 3 5 6 7 8 ...
 $ ESTRATO.GEOGRÁFICO                            : int  1 1 1 1 1 1 1 1 1 1 ...
 $ FATOR.DE.EXPANSÃO.2..AJUSTADO.P..ESTIMATIVAS.: num  1887 1887 1887 1887 1887 ...
 $ PERÍODO.REAL.DA.COLETA                        : int  1 10 11 12 14 4 5 6 7 9 ...
 $ QUANTIDADE.DE.MORADORES                       : int  4 5 3 4 3 2 1 3 3 4 ...
 $ QUANTIDADE.DE.UC                              : int  1 1 1 1 1 1 1 1 1 1 ...
 $ QUANTIDADE.DE.FAMÍLIAS                        : int  1 1 1 1 1 1 1 1 1 1 ...
 $ TIPO.DE.DOMICILIO                             : int  2 1 2 2 2 2 1 2 2 2 ...
 $ MATERIAL.QUE.PREDOMINA.NAS.PAREDES.EXTERNAS   : int  1 1 1 1 1 1 1 1 1 1 ...
 $ MATERIAL.QUE.PREDOMINA.NA.COBERTURA           : int  2 2 2 2 2 2 1 2 2 2 ...
 $ MATERIAL.QUE.PREDOMINA.NO.PISO                : int  2 2 2 6 4 2 4 2 2 3 ...
 $ QUANTIDADE.DE.CÔMODOS                         : int  8 7 8 6 8 8 5 8 8 8 ...
 $ CÔMODOS.SERVINDO.DE.DORMITÓRIO                : int  3 2 3 3 3 2 1 3 2 3 ...
 $ EXISTÊNCIA.DE.ÁGUA.CANALIZADA                 : int  1 1 1 1 1 1 1 1 1 1 ...
 $ PROVENIÊNCIA.DA.ÁGUA                          : int  1 1 1 1 1 1 1 1 1 1 ...
 $ QUANTIDADE.DE.BANHEIROS                       : int  2 2 2 1 2 2 1 2 2 2 ...
 $ ESCOADOURO.SANITÁRIO                          : int  1 1 1 1 1 1 1 1 1 1 ...
 $ CONDIÇÃO.DE.OCUPAÇÃO                          : int  1 6 1 1 6 1 1 2 1 1 ...
 $ TEMPO.DE.ALUGUEL                              : int  0 2 0 0 2 0 0 0 0 0 ...
 $ TIPO.DE.CONTRATO.DE.ALUGUEL                   : int  0 3 0 0 2 0 0 0 0 0 ...
 $ EXISTÊNCIA.DE.PAVIMENTAÇÃO.NA.RUA             : int  1 1 1 1 1 1 1 1 1 1 ...
 $ IMPUTAÇÃO...QUANTIDADE.DE.CÔMODOS             : int  0 0 0 0 0 0 0 0 0 0 ...
 $ IMPUTAÇÃO...QUANTIDADE.DE.BANHEIROS           : int  0 0 0 0 0 0 0 0 0 0 ...
 $ IMPUTAÇÃO...ESCOADOURO.SANITÁRIO              : int  0 0 0 0 0 0 0 0 0 0 ...
 $ RENDA.MONETÁRIA.MENSAL.DO.DOMICÍLIO           : num  2496 2112 8400 4046 1142 ...
 $ RENDA.NÃO.MONETÁRIA.MENSAL.DO.DOMICÍLIO       : num  848 0 587 416 0 ...
 $ RENDA.TOTAL.MENSAL.DO.DOMICÍLIO               : num  3344 2112 8987 4462 1142 ...
```

A saída anterior mostra que o objeto dados é um `data.frame` com 3504 linhas (observações) e 31 colunas (variáveis). É possível ver o nome e o tipo de dado que cada coluna recebe. Para mostrar na tela apenas uma coluna do objeto dados, você pode utilizar os comandos abaixo.

```
> # As duas formas abaixo mostram a segunda coluna do objeto dados
> dados$CÓDIGO.DA.UF
> dados[, 2]
```

• **Funções** `dim()` **e** `names()`

Outras funções úteis são as funções `dim()` e `names()`. A função `dim()` mostra a dimensão do objeto. O objeto de dados é um *data frame*, então esta função retornará o número de linhas e colunas. A função `names()` mostra o nome de cada coluna do objeto. É bem útil para localizar o número de alguma coluna desejada.

```
> # dimensão: número de linhas e colunas
> dim(dados)
```

```
[1] 3504   31
```

O arquivo dados possui 3504 linhas e 31 colunas.

```
> #nome de cada coluna
> names(dados)
```

```
 [1] "TIPO.DE.REGISTRO"
 [2] "CÓDIGO.DA.UF"
 [3] "NÚMERO.SEQUENCIAL"
 [4] "DV.DO.SEQUENCIAL"
 [5] "NÚMERO.DO.DOMICÍLIO"
 [6] "ESTRATO.GEOGRÁFICO"
 [7] "FATOR.DE.EXPANSÃO.2..AJUSTADO.P..ESTIMATIVAS."
 [8] "PERÍODO.REAL.DA.COLETA"
 [9] "QUANTIDADE.DE.MORADORES"
[10] "QUANTIDADE.DE.UC"
[11] "QUANTIDADE.DE.FAMÍLIAS"
[12] "TIPO.DE.DOMICILIO"
[13] "MATERIAL.QUE.PREDOMINA.NAS.PAREDES.EXTERNAS"
[14] "MATERIAL.QUE.PREDOMINA.NA.COBERTURA"
[15] "MATERIAL.QUE.PREDOMINA.NO.PISO"
[16] "QUANTIDADE.DE.CÔMODOS"
[17] "CÔMODOS.SERVINDO.DE.DORMITÓRIO"
[18] "EXISTÊNCIA.DE.ÁGUA.CANALIZADA"
[19] "PROVENIÊNCIA.DA.ÁGUA"
[20] "QUANTIDADE.DE.BANHEIROS"
[21] "ESCOADOURO.SANITÁRIO"
[22] "CONDIÇÃO.DE.OCUPAÇÃO"
[23] "TEMPO.DE.ALUGUEL"
[24] "TIPO.DE.CONTRATO.DE.ALUGUEL"
[25] "EXISTÊNCIA.DE.PAVIMENTAÇÃO.NA.RUA"
[26] "IMPUTAÇÃO...QUANTIDADE.DE.CÔMODOS"
[27] "IMPUTAÇÃO...QUANTIDADE.DE.BANHEIROS"
[28] "IMPUTAÇÃO...ESCOADOURO.SANITÁRIO"
[29] "RENDA.MONETÁRIA.MENSAL.DO.DOMICÍLIO"
[30] "RENDA.NÃO.MONETÁRIA.MENSAL.DO.DOMICÍLIO"
[31] "RENDA.TOTAL.MENSAL.DO.DOMICÍLIO"
```

Maneiras fáceis de aprender mais no R

Como sempre nos deparamos com algo novo no R, é fundamental saber como aprender a usar novas funções. Para isso iremos apresentar algumas formas de enfrentar o desconhecido.

• **Função** `apropos()`

Quando queremos procurar um objeto e não lembramos o nome completo dele, podemos usar a função `apropos()`. Ela funciona como um filtro que possibilita encontrar os objetos que tenham o pedaço do nome que você lembra. Por exemplo :

```
> apropos("air")
 [1] "airmiles"             "AirPassengers"        "airquality"
 [4] "as.pairlist"          "cairo_pdf"            "cairo_ps"
 [7] "HairEyeColor"         "is.pairlist"          "pairlist"
[10] "pairs"                "pairs.default"        "pairwise.prop.test"
[13] "pairwise.t.test"      "pairwise.table"       "pairwise.wilcox.test"
```

Mostra uma lista de 15 nomes de objetos que contém o texto "air" no nome.

• **Função** `help()` **ou** `?`

Quando queremos saber de mais detalhes do objeto, você pode pedir ajuda ao R com o `help(nome_do_objeto)` ou `?nome_do_objeto`. Por exemplo:

```
> help(AirPassengers)   # ou
> ?AirPassengers
```

Uma janela abrirá no help com a descrição dos dados da série temporal *AirPassengers*.

• **Função** `example()`

Essa função é muito útil para mostrar alguns exemplos de uso de outras funções.

```
> example(plot)
```

Dá alguns exemplos de gráficos usando a função `plot()`.

Suponha que queremos criar um gráfico de barras, mas nunca vimos alguma função que faça isso. Primeiro vamos buscar nomes de objetos que tenha "*bar*" (barra em inglês) no nome. Para isso utilizamos a função `apropos()`.

```
> apropos("bar")
 [1] "barplot"           "barplot.default"   "bartlett.test"
 [4] "getTxtProgressBar" "getWinProgressBar" "setStatusBar"
 [7] "setTxtProgressBar" "setWinProgressBar" "txtProgressBar"
[10] "winProgressBar"
```

O R retornou 10 nomes de objetos, dentre ele apareceu o `barplot` que é uma candidata a função para criar gráfico de barra. Agora vamos pedir ajuda para o R sobre `barplot()` para ter mais detalhes da função.

```
> help(barplot)      # ou
> ?barplot
```

E ainda podemos pedir alguns exemplos de `barplot()` com a função `example()`.

```
> example(barplot)
```

Estatística Descritiva e Gráficos

Daiane Marcolino de Mattos

Pedro Costa Ferreira

Estatística Descritiva

Estatística descritiva é a área da estatística que descreve e resume informações sobre os dados. O R disponibiliza uma variedade de funções que permitem fazer isso. Porém, antes de começar a programar, é necessário entender alguns conceitos fundamentais.

População e amostra

Os dados que trabalhamos em estatística descritiva são provenientes de uma amostra ou população. População é um conjunto de todas as unidades que tem uma característica em comum na qual estamos interessados. Por exemplo, todos os funcionários da Fundação Getúlio Vargas no estado do RJ. Como, geralmente, fazer uma pesquisa que envolva o conjunto inteiro da população requer bastante tempo, dinheiro e, em alguns casos, o experimento pode ser destrutivo, utiliza-se uma amostra, ou seja, uma parte da população. Para selecionar uma amostra que represente de fato a população, há diversas metodologias possíveis.

Variáveis

Além da quantidade de unidades na amostra, existem as variáveis. Estas são as características nas quais estamos interessados em descobrir e podem mudar de acordo com cada unidade amostral. Por exemplo, a idade, o sexo, o grau de escolaridade e o salário de cada funcionário. Variáveis podem ser classificadas em qualitativas e quantitativas.

As variáveis qualitativas, também conhecidas como categóricas, são aquelas que assumem atributos ou qualidades. São divididas em ordinais e nominais. As ordinais são aquelas que podem ser ordenadas de alguma forma, por exemplo, o grau de escolaridade de um funcionário (fundamental, médio e superior). As nominais são aquelas em que não faz sentido alguma ordenação, por exemplo, o sexo do funcionário (masculino ou feminino).

As variáveis quantitativas são aquelas que podem ser medidas, já que apresentam valores numéricos. São classificadas em discretas e contínuas. As discretas podem assumir apenas um número finito ou infinito contável de valores e, assim, somente fazem sentido valores inteiros. Geralmente são resultados de contagens. Por exemplo: número de filhos do funcionário. As variáveis quantitativas contínuas são aquelas que assumem valores numéricos (toda a reta real) e, em geral, são resultantes de mensurações. Por exemplo: a idade, a altura, o peso e o salário do funcionário.

Após a coleta dos dados, utilizamos ferramentas da estatística descritiva para analisar e resumir informações sobre eles.

Tabela de Frequências

Quando se tem uma grande quantidade de dados, uma boa forma de resumi-los é criando uma tabela de frequências. A tabela de frequências mostra os tipos de valores que uma variável assume e a quantidade de vezes que os mesmos ocorrem. Para criá-la utilize a função `table()`.

Exemplo: Criaremos uma tabela de frequência sobre o material que predomina na cobertura (telhado) dos domicílios (coluna 14 do arquivo da POF (IBGE, 2015a)).

```
> # Ler a base de dados da POF
> dados <- read.csv2("POF_capitais.csv")

> # Criando uma tabela de frequências
> table(dados[ ,14])

   1    2    3    4    5    7
1678 1789   14    3    2   18
```

A primeira linha indica os números observados na pesquisa. Cada número representa um tipo de material diferente:

1. Telha qualquer;
2. Laje de concreto;
3. Madeira para construção;
4. Chapa metálica;
5. Madeira aproveitada;
6. Palha;
7. Outro material.

A segunda linha representa a frequência com que os números ocorrem, ou seja, quantas vezes cada tipo se repete. Pelo resultado acima, vemos que a maioria dos domicílios possui cobertura de telha qualquer (1) ou laje de concreto (2). O número 6 (cobertura de palha) foi omitido já que não apresenta frequência.

Medidas de Posição

Além da criação de tabelas, é possível resumir dados a partir de simples valores numéricos. Medidas de posição são assim chamadas, pois retornam um ponto (ou valor) no qual estão concentrados os outros valores observados.

• Média

A média é a medida de posição mais conhecida para resumir dados.

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} \tag{2.1}$$

Para calculá-la precisamos da soma da variável e do número de unidades que a variável possui. Para calcular a soma de uma variável quantitativa podemos usar a função `sum()` e para saber o número de elementos que a variável possui podemos usar a função `length()`.

Exemplo: Calcularemos a média de moradores por domicílio (coluna 9).

```
> # somatório da coluna 9
> soma <- sum(dados[, 9])
> # número de unidades na coluna 9
> n <- length(dados[, 9])
> # cálculo da média de moradores
> media <- soma/n
> media
[1] 3.134989
```

Podemos calcular a média, também, utilizando a função `mean()` do R.

```
> # calcula a média da coluna 9
> mean(dados[, 9])
[1] 3.134989
```

• Mediana

A mediana é a medida que divide os dados ordenados exatamente ao meio, ou seja, o valor que separa os 50% menores dos 50% maiores valores.

$$md(X) = \begin{cases} x_{(\frac{n+1}{2})} & \text{,se } n \text{ é ímpar} \\ \frac{x_{(\frac{n}{2})} + x_{(\frac{n}{2}+1)}}{2} & \text{,se } n \text{ é par} \end{cases}$$

Exemplo: Encontraremos a mediana para a quantidade de moradores no domicílio (coluna 9). Antes de usar a fórmula acima, precisamos ordenar os dados. Para isso utilizamos a função `sort()`.

```
> # ordenando os dados
> ordem <- sort(dados[, 9])
> n <- length(ordem)
> # calculando a mediana
> if(n %% 2 == 0){
+   med <- (ordem[n/2] + ordem[n/2 + 1])/2
+ }else{
+   med <- ordem[(n+1)/2] }
> med
[1] 3
```

Podemos calcular a mediana, também, utilizando a função `median()` do R:

```
> # calcula a mediana da quantidade de moradores
> median(dados[, 9])
[1] 3
```

Importante: Para calcular a mediana com a função `median()` não é necessário ordenar os dados, o R faz isso automaticamente.

• Máximo e Mínimo

O R possui duas funções que retornam o maior e o menor valor observado no conjunto de dados: `min()` e `max()`.

Exemplo: Encontraremos o menor e o maior valor observado para a quantidade de moradores no domicílio.

```
> # retorna o menor valor
> min(dados[, 9])
[1] 1
> # retorna o maior valor
> max(dados[, 9])
[1] 15
```

Caso quiséssemos identificar quais domicílios possuem uma ou quinze pessoas, poderíamos usar a função `which()`. Essa função retornará quais as linhas que satisfazem determinada condição.

Exemplo: Retornar quais domicílios na base de dados possui a quantidade de moradores igual a 1.

```
> # Função which
> which(dados[, 9] == 1)
```

• Quantis

Quantil é uma medida de posição que corresponde a uma proporção. O quantil de a% ($0 < a < 100$) representa o valor que separa os a% menores valores dos (100-a)% maiores valores no conjunto de dados ordenados. Para entender melhor, veja os exemplos a seguir.

A mediana é o quantil de 50%; O quantil de 20% representa o valor que separa os 20% menores dos 80% maiores no conjunto de dados ordenados. O quantil de 35% representa o valor que separa os 35% dos 65% maiores no conjunto de dados ordenados. E assim sucessivamente.

Para calcular os quantis, assim como a mediana, é necessário que os dados estejam ordenados, porém, o R já faz isso automaticamente usando a função `quantile()`.

```
quantile(x, probs = vetor de quantis)
```

Exemplo: Encontraremos os quantis de 10%, 25%, 50%, 75%, 90% para a variável Renda Total Mensal do Domicílio (coluna 31).

```
> # Encontrando os quantis
> quantile( dados[, 31], probs = c(0.10, 0.25, 0.50, 0.75, 0.90))
      10%       25%       50%       75%       90%
 756.9295 1217.9387 2218.8527 4481.7586 9186.2312
```

Podemos interpretar esses números da seguinte forma: 10% dos domicílios possuem renda inferior a R$ 756,93. 75% dos domicílios possuem renda inferior a $4481,76.

• Moda

Podemos descobrir qual é o número mais comum de moradores em um domicílio. Uma maneira de fazê-lo é criar uma tabela de frequências (subseção 2.1.3) e descobrir o número que mais se repete. Em estatística descritiva, chamamos isso de moda. Para criar uma tabela de frequências use a função `table()`.

Exemplo: Encontraremos a quantidade mais comum de moradores em um domicílio.

```
> # cria uma tabela de frequências
> table(dados[, 9])
  1   2   3   4   5   6   7   8   9  10  11  15
488 839 873 746 316 145  52  23  11   6   4   1
```

A primeira linha, na saída acima, representa o número de moradores em um domicílio observados na pesquisa. A segunda linha corresponde à frequência dele. Ou seja, há apenas um domicílio com 15 moradores enquanto há quatro domicílios com 11. A maior frequência observada é 873, que corresponde ao número de três moradores. Portanto, três é a moda de moradores na pesquisa.

• Função `summary`

A função `summary()` retorna várias medidas de posições de uma só vez: a média, a mediana, o primeiro e o terceiro quartis e o menor e o maior valor da variável em estudo. Caso a variável seja categórica, essa função retornará apenas a frequência de cada categoria.

Exemplo: Obteremos, então, um resumo de informações sobre a Quantidade de Moradores (coluna 9) e para Renda Total Mensal do Domicílio (coluna 31).

```
> # estatísticas para a quantidade de moradores
> summary(dados[, 9])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   3.135   4.000  15.000
```

Concluímos que o menor número de moradores em um domicílio, nessa pesquisa, é um e o maior é quinze e que, em média, há 3,135 moradores em um domicílio.

```
> # estatísticas para a renda
> summary(dados[,31])
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
    50.78   1218.00   2219.00   4082.00   4482.00 117200.00
```

Observe a má distribuição de renda do Brasil. Há domicílios que recebem R$ 50,78 e outros que recebem R$ 117.200,00. A média da renda é de R$ 4.082,00. A média é uma medida que é influenciada por *outliers* (valores atípicos). Assim, dependendo do comportamento dos dados (muito dispersos ou não), esta pode não ser uma boa medida para representar os dados.

Medidas de Dispersão

Medidas de dispersão são úteis para verificar se os dados são homogêneos ou heterogêneos. A seguir serão apresentadas as funções mais utilizadas para quantificar a variabilidade dos dados.

• Variância Amostral

$$s^2 = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n-1} = \frac{\sum_{i=1}^{n} y_i^2 - n\bar{y}^2}{n-1}$$

Como exemplo, calcularemos a variância amostral para os dados da coluna 9 (quantidade de moradores). Usaremos a função `var()`.

```
> var(dados[, 9])
[1] 2.437668
```

• Desvio-Padrão

A variância por ser uma medida quadrática, diferente da ordem de grandeza dos dados observados, pode, às vezes, dificultar a interpretação. Como alternativa, utilizamos a raiz da variância, o desvio padrão, para compreender melhor.

Quanto maior o desvio padrão mais dispersos estão os dados.

$$s = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n-1}}$$

Podemos calculá-lo apenas tirando raiz da variância ou usando a função `sd()`.

```
> # raiz da variância
> sqrt(var(dados[, 9]))
[1] 1.561303
> # desvio-padrão
> sd(dados[, 9])
[1] 1.561303
```

Percebe-se que o desvio padrão não é grande em relação à média. Isso significa que não há uma dispersão tão forte entre o número de pessoas que moram em cada domicílio. Isso pode ser visualizado de forma melhor com a elaboração de gráficos (seção 2.2).

• Amplitude Total e Amplitude Interquartil

Amplitude total é a diferença entre o maior (máximo) e o menor (mínimo) valor observado. Porém, se por acaso no conjunto de dados houver valores atípicos/*outliers* (valores extremamente pequenos ou grandes), essa medida pode não representar bem o conjunto de dados. Portanto, não é uma boa medida de dispersão.

Amplitude interquartilíca é a diferença entre o terceiro e o primeiro quartis. É preferível em relação à amplitude total por não ser afetada por valores extremos. O R utiliza a função `IQR()` para essa finalidade.

Calcularemos as amplitudes total e interquartílica para a coluna 31 do arquivo de dados (renda total mensal do domicílio).

```
> # calculando a amplitude total
> ampt <- max(dados[, 31]) - min(dados[, 31])
> ampt
[1] 117168.4
> # calculando a amplitude interquartil
> IQR(dados[, 31])
[1] 3263.82
```

• Coeficiente de Variação

O coeficiente de variação é uma medida de dispersão útil para comparar dois ou mais conjuntos de dados quando estes estão em unidades de medidas diferentes, caso contrário poderiam ser comparados pelo desvio padrão. É uma medida relativa resultante da divisão do desvio padrão pela média.

$$CV = \frac{s}{\bar{x}} \times 100$$

Quanto menor for o coeficiente de variação menor será a dispersão dos dados em torno da média, ou seja, os dados são homogêneos. Um CV maior do que 30% sugere que os dados são heterogêneos, ou seja, estão dispersos em torno da média. Há quem diga isso para CV maior do que 25% ou 50%, tornando mais difícil haver um valor padrão de comparação. Porém, ao comparar diversos conjuntos de dados, aquele que possuir o menor CV é o mais homogêneo. Abaixo o CV da Renda Monetária Mensal do Domicílio (coluna 29).

```
> # calculando o coef. de variação
> sd(dados[, 29])/mean(dados[, 29])*100
[1] 147.3442
```

Como se observa, o CV é igual a 147,34%, indicando que os dados são muito heterogêneos, ou seja, dispersos em relação à média. Assim, a média não é uma medida que representa tão bem esse conjunto de dados, principalmente por ela ser afetada pelas rendas muito altas que representam uma minoria de observações.

Covariância e Correlação

Covariância e correlação medem a dependência linear de duas variáveis quantitativas. Porém, a covariância não é uma medida padronizada. Sendo assim, a correlação mais fácil de interpretar.

A correlação varia entre -1 e 1. Quanto mais perto de -1 ou 1 significa que as variáveis possuem uma forte associação linear. Quanto mais próximo de zero significa que as variáveis possuem fraca associação linear. Sinal positivo indica que quando uma variável cresce (ou diminui), a outra variável também cresce (ou também diminui). Já o sinal negativo indica que quando uma variável cresce, a outra diminui e vice-versa. A seguir estão as fórmulas de como calcular a covariância e a correlação.

$$cov(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1} = \frac{\sum_{i=1}^{n}(x_i y_i - n\bar{x}\bar{y})}{n-1}$$

$$\rho(x,y) = \frac{cov(x,y)}{s_x s_y}$$

Como exemplo, calcularemos essas medidas para quantificar a associação linear entre a Quantidade de Banheiros (coluna 20) e a Renda Total Mensal do Domicílio (coluna 31). A covariância e a

correlação podem ser calculadas diretamente pelas funções `cov()` e `cor()`, respectivamente.

```
> cov(dados[, 20], dados[, 31])
[1] 3221.584
> #correlação
> cor(dados[, 20], dados[, 31])
[1] 0.5807561
```

A correlação entre as variáveis resultou em 0,5807 indicando que há uma associação linear positiva entre elas, ou seja, quanto maior o número de banheiros, maior é a renda do domicílio. Lembrando que nem sempre há relação de causa e efeito, portanto ter um grande número de banheiros não significa maior renda.

Medidas calculadas por grupos

Muitas vezes estamos interessados em calcular a média, ou outras medidas, de uma variável de acordo com os grupos de outra variável. A função `tapply()` é simples e útil para essa finalidade.

```
tapply(x, INDEX, FUN)
```

onde `x` é a variável principal, `INDEX` é o grupo (a outra variável) e `FUN` é a medida a ser calculada: média, variância, etc.

Exemplo: Imagine que queremos descobrir a renda média de domicílio por Unidade de Federação.

```
> # Calculando a renda média do domicílio por UF
> tapply(dados[,31], dados[,2], mean)
      29       31       33       35       43       53
2659.425 4799.979 4485.673 4188.369 3862.561 4081.720
```

Assim, temos a renda média de acordo com a UF. Verifica-se que a maior renda média é a de Minas Gerais enquanto a menor é a da Bahia.

Exemplo: Podemos descobrir também a dispersão da renda por UF.

```
> # Calculando a dispersão da renda do domicílio por UF
> tapply(dados[, 31], dados[, 2], sd)
      29       31       33       35       43       53
4125.793 8319.749 6352.097 4710.368 5296.013 5546.274
```

Minas Gerais também apresentou a maior dispersão de renda, porém, o coeficiente de variação seria mais eficaz para comparar a dispersão entre as UF's.

Para facilitar a visualização é possível combinar esses dois resultados.

```
> media <- tapply(dados[, 31], dados[, 2], mean)
> desvio <- tapply(dados[, 31], dados[, 2], sd)
> # visualizar em data frame
```

```
> UF <- data.frame(media,desvio)
> UF
      media   desvio
29 2659.425 4125.793
31 4799.979 8319.749
33 4485.673 6352.097
35 4188.369 4710.368
43 3862.561 5296.013
53 4081.720 5546.274
```

Calcularemos o coeficiente de variação para esses dados também.

```
> # coeficiente de variação
> UF$CV <- UF$desvio/UF$media*100
> UF
      media   desvio       CV
29 2659.425 4125.793 155.1385
31 4799.979 8319.749 173.3289
33 4485.673 6352.097 141.6086
35 4188.369 4710.368 112.4631
43 3862.561 5296.013 137.1114
53 4081.720 5546.274 135.8808
```

Em todas as Unidades de Federação, o coeficiente de variação é extenso, ou seja, há uma dispersão grande na renda nos estados.

Caso você queira descobrir alguma medida para mais de uma variável de uma só vez, utilize a função `agregate()`.

```
aggregate(variáveis, list(grupo), medida)
```

Exemplo: Descobrir, de acordo com a UF, as médias da renda e do número de moradores ao mesmo tempo.

```
> # variáveis: renda e número de moradores
> variaveis <- data.frame(dados[,9],dados[,31])
> # média das variáveis por UF
> medias <- aggregate(variaveis, list(dados[,2]), mean)
> # editando o nome das colunas
> colnames(medias) <- c("UF", "Moradores", "Renda")
> medias
  UF Moradores    Renda
1 29  3.316699 2659.425
2 31  2.994056 4799.979
3 33  3.015050 4485.673
4 35  3.196629 4188.369
5 43  2.753731 3862.561
6 53  3.312842 4081.720
```

Assim obtivemos a média das variáveis de renda e de quantidade de moradores ao mesmo tempo.

Criando Gráficos com o R

Muitas vezes um gráfico bem feito torna mais fácil de entender o comportamento de um conjunto de dados. Nesta seção serão vistos os principais gráficos utilizados para descrever dados. São eles: histograma, boxplot, gráfico de dispersão ou de pontos, gráfico de pizza e gráfico de barras. Na Figura (2.1) podem ser vistos alguns exemplos dos gráficos anteriores.

No R, existem alguns pacotes que permitem a criação de gráficos bem elaborados com diversas opções de edição. Aqui, utilizaremos o pacote `graphics` (R Core Team, 2015a) que foi instalado junto com o próprio R, ou seja, não é necessário nenhum outro pacote para executar a criação dos gráficos nessa seção.

Cada gráfico que faremos possui sua própria função no R. Porém, há alguns argumentos que são comuns em cada uma das funções. São eles:

- `main`: título do gráfico;
- `xlab`: texto do eixo x;
- `ylab`: texto do eixo y;
- `col`: cor do gráfico.

Com esses quatro argumentos você cria um gráfico com o mínimo de informações necessárias para o entendimento do mesmo.

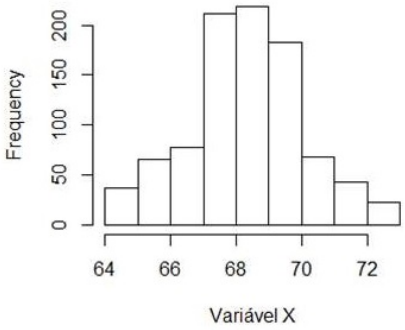
A seguir, veremos detalhadamente as funções que criam os tipos de gráficos apresentados anteriormente.

Histograma

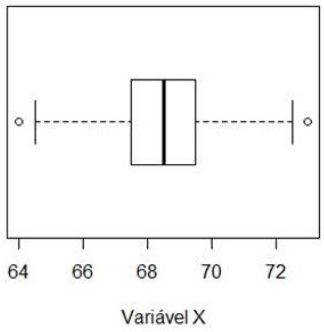
O histograma é a representação gráfica de uma distribuição de frequências. É útil para visualizar como os dados se comportam, os valores mais e menos frequentes. O histograma pode mostrar a proporção dos dados ao invés da frequência absoluta em cada classe.

No R, utilizamos a função `hist()` para esboçar um histograma. Os argumentos mais utilizados nesta função são:

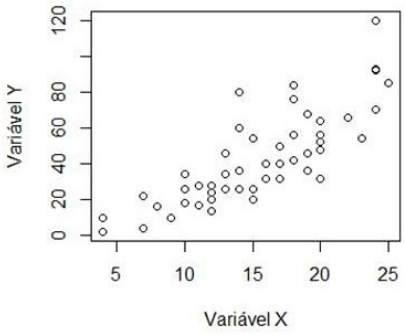
- `x`: vetor de dados (necessário);
- `main`: título do gráfico;
- `xlab`: texto do eixo x;



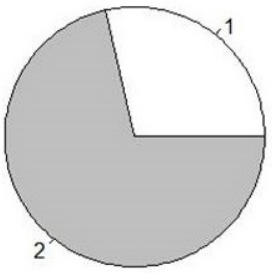
(a) Exemplo de Histograma



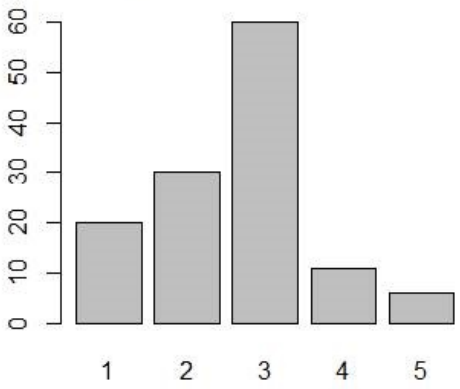
(b) Exemplo de Boxplot



(c) Exemplo de Gráfico de Dispersão



(d) Exemplo de Gráfico de Pizza



(e) Exemplo de Gráfico de Dispersão

Figura 2.1: Exemplos de gráficos no R

- `ylab`: texto do eixo y;
- `xlim`: limite do eixo x;
- `ylim`: limite do eixo y;
- `col`: cor do preenchimento do histograma;
- `border`: cor da linha/borda do histograma.

Exemplo: Utilizaremos uma base de dados chamada `galton`, para fazer o histograma. Esta base contém 928 medidas de altura de pais e de seus respectivos filhos. A altura dos pais foi calculada como uma média entre as alturas da mãe e do pai. Há, na base, pais que possuem mais de um filho, portanto existem alturas de pais repetidas. A unidade de medida é dada em polegadas, portanto, para um melhor entendimento, converteremos para centímetros.

Para ler a base é necessário instalar o pacote UsingR (Verzani, 2015).

```
> # lendo a base
> install.packages("UsingR")
> require(UsingR)
> head(galton)

  child parent
1  61.7   70.5
2  61.7   68.5
3  61.7   65.5
4  61.7   64.5
5  61.7   64.0
6  62.2   67.5

> # Convertendo de polegadas para centímetros (1 polegada é aproximadamente 2,54 cm)
> galton <- 2.54 * galton
> head(galton)

    child parent
1 156.718 179.07
2 156.718 173.99
3 156.718 166.37
4 156.718 163.83
5 156.718 162.56
6 157.988 171.45
```

Esboçaremos um histograma para a altura dos filhos.

```
> # esboçando o histograma
> hist(galton$child, main = "", xlab = "Altura (cm)",ylab="Frequência",ylim=c(0,200))
```

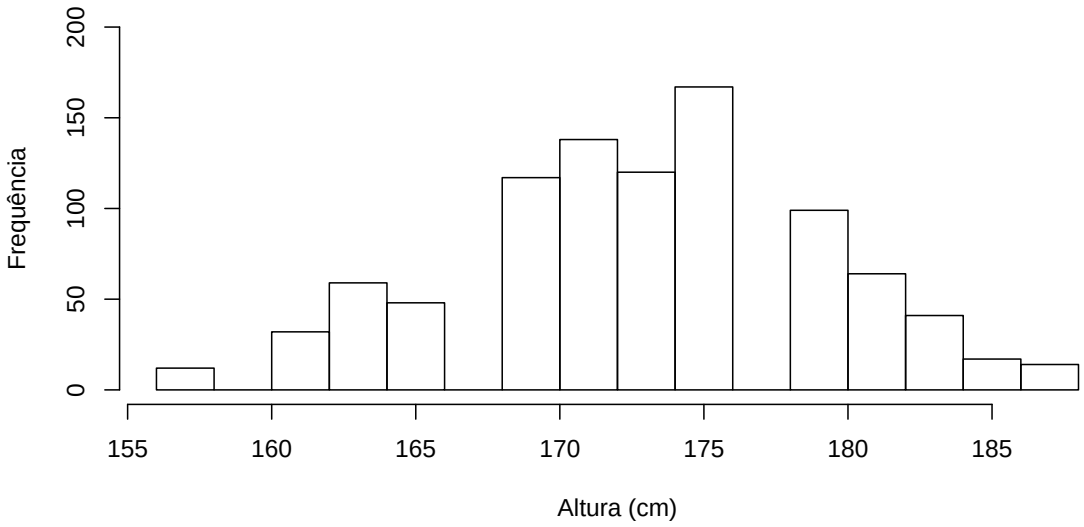


Figura 2.2: Histograma das alturas dos filhos (base `galton`)

Podemos ver através do gráfico que as maiores frequências estão por volta de 170 cm e 175 cm de altura. Caso queira acrescentar alguma cor ao gráfico, utilize os argumentos `col` e `border`. Nas referências desse documento é possível encontrar uma lista de cores que podem ser usadas.

```
> # esboçando o histograma
> hist(galton$child, main = "", xlab = "Alturas (cm)", ylab="Frequência",ylim=c(0,200),
 col = "lightgray", border = "steelblue")
```

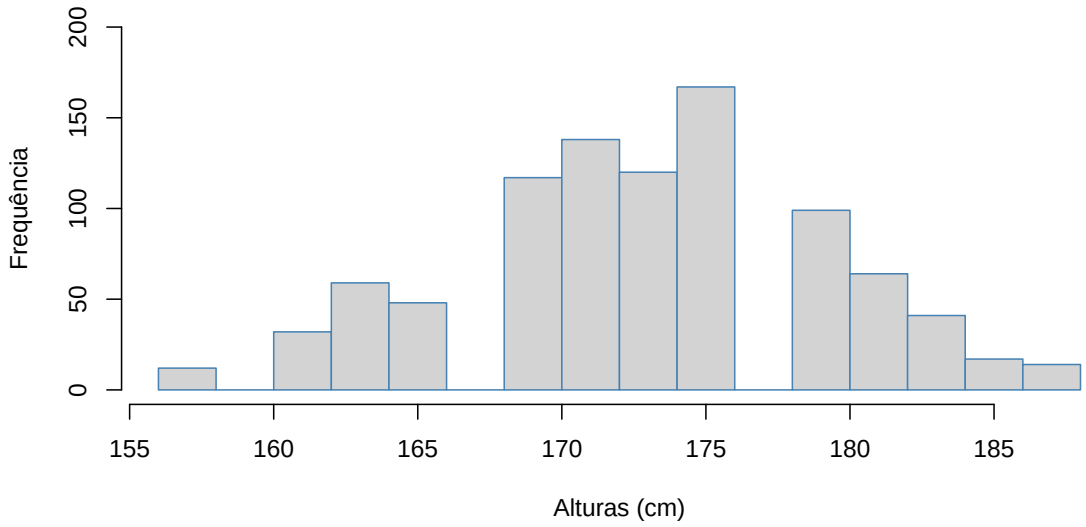


Figura 2.3: Histograma das alturas dos filhos com cor (base `galton`)

Agora, vamos a outro exemplo. Esboçaremos um histograma para a Renda Total Mensal do Domicílio (coluna 31) da POF.

```
> # criando o gráfico
> hist(dados[, 31], main = "",xlab = "Renda Total Mensal do Domicílio",
ylab="Frequência", col = "lightsteelblue3")
```

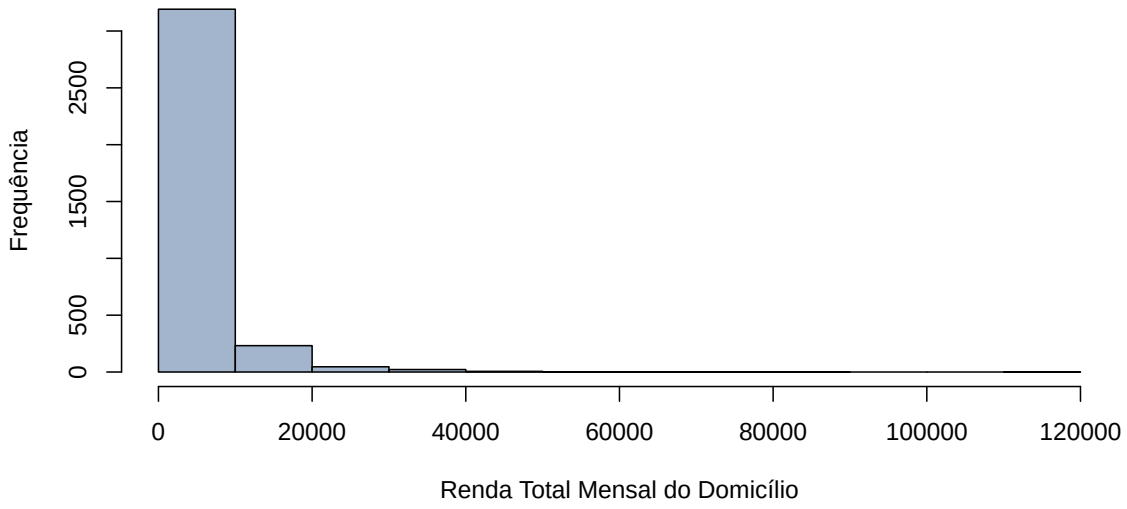


Figura 2.4: Histograma da Renda dos Domicílios (POF)

Pelo histograma concluímos que um pouco mais do que 3000 domicílios possuem renda entre 0 e 10 mil reais, lembrando que temos 3504 domicílios na amostra, e que uma minoria (menos de 500 domicílios) possui renda superior a 10 mil reais.

Para ver uma melhor distribuição daqueles que possuem renda inferior a 10 mil reais, podemos filtrar os dados e fazer outro histograma só para eles. Para isso, usamos a função `subset()`.

```
subset(x, condição de x)
```

A condição, nesse caso, será os `x` menor do que dez mil reais:

```
> # filtrando os dados
> menorq10 <- subset(dados[, 31], dados[, 31] < 10000)
```

Após guardar os dados na condição que escolhemos, podemos fazer o histograma.

```
> # criando o gráfico
> hist(menorq10, main = "", xlab = "Renda Total Mensal do Domicílio em R$",
ylab="Frequência", col = "lightskyblue")
```

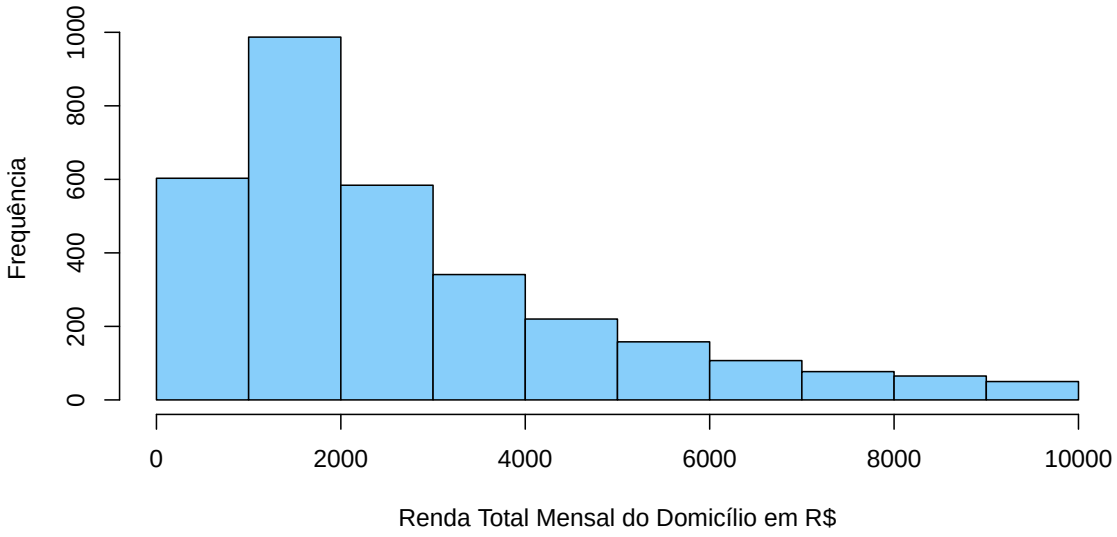


Figura 2.5: Histograma da Renda dos Domicílios para domícilios com renda inferior a 10 mil reais (POF)

Agora, está mais fácil visualizar o comportamento da maioria dos domicílios. Percebe-se que há uma grande concentração entre zero e três mil reais, que representam aproximadamente 2200 domicílios. A partir de 2 mil reais, o número de domicílios começa a decrescer.

É assim que vemos que a distribuição dos dados é assimétrica, pois há uma concentração maior em um lado e a cauda do gráfico se encontra em outro. Podemos calcular o coeficiente de assimetria

para essa distribuição, para quantificar o quão assimétrica é a distribuição dos dados.

Classificação do coeficiente de assimetria:

	0,15		**1**	
Simétrica	|	**Assimétrica moderada**	|	**Assimétrica Forte**

É necessária a instalação do pacote `moments` (Komsta & Novomestky, 2015) para calcular o coeficiente.

```
> # instalando e carregando o pacote moments
> install.packages("moments")
> require(moments)
> # calcular assimetria com a fórmula
> skewness(menorq10)
```

```
[1] 1.355681
```

O coeficiente de assimetria igual 1,355 diz que a distribuição é assimétrica forte. Essa distribuição é assimétrica à direita, já que a cauda está a direita. Gráficos de renda geralmente são classificados assim. Também é possível ver assimetrias através de boxplots.

Boxplot

O boxplot é um gráfico construído com base no resumo de cinco números: limite inferior (LI), 1º quartil (Q1), mediana (Q2), 3º quartil (Q3) e limite superior (LS). O gráfico tem um formato de caixa cuja sua largura é representada pelos 1º e 3º quartis. Portanto, 50% das observações estão concentradas dentro da caixa. Os limites inferior e superior são representados por linhas fora da caixa.

O boxplot é muito útil para descobrir se há *outliers* no conjunto de dados, ou seja, valores afastados da maioria das observações. Esses valores aparecem, no gráfico, fora dos limites inferior e superior.

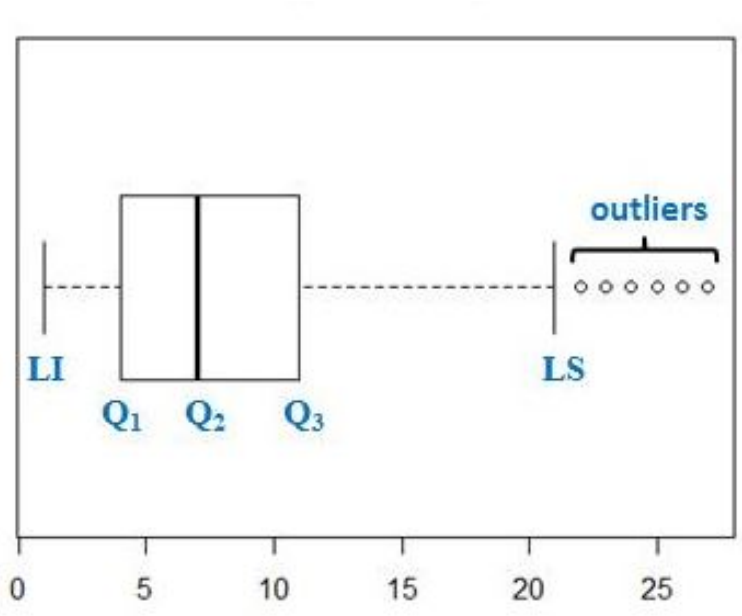


Figura 2.6: Exemplo de boxplot

Para esboçar um boxplot, utilizamos a função `boxplot()`. Os argumentos mais utilizados nesta função são:

- `x`: vetor de dados (necessário);
- `main`: título do gráfico;
- `xlab`: texto do eixo x;
- `ylab`: texto do eixo y;
- `col`: cor do preenchimento da caixa;
- `border`: cor da linha/borda da caixa;
- `horizontal`: se `TRUE`, a caixa aparece no formato horizontal, se `FALSE` (*default*), aparece no formato vertical.

Exemplo: Ainda com a base `galton`, façamos um boxplot para a altura dos pais (coluna 2).

```
> boxplot(galton$parent, main = "", ylab = "Altura (cm)", col = "seagreen3")
```

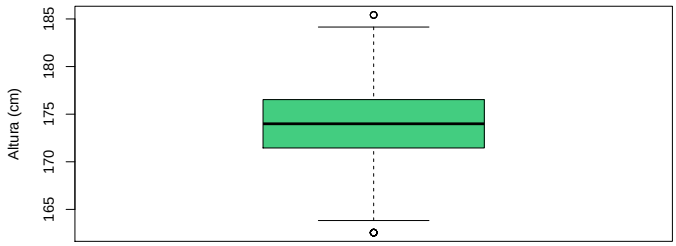


Figura 2.7: Boxplot para altura dos pais (base `galton`)

O gráfico indicou dois valores *outliers* e os dados se distribuem de forma simétrica.

Criaremos um boxplot para a renda dos domicílios que possuem renda inferior a dez mil reais (objeto criado anteriormente).

```
> #criando o boxplot
> boxplot(menorq10, horizontal = T, col = "gold",
          xlab = "Renda Total Mensal do Domicílio",
          main = "")
```

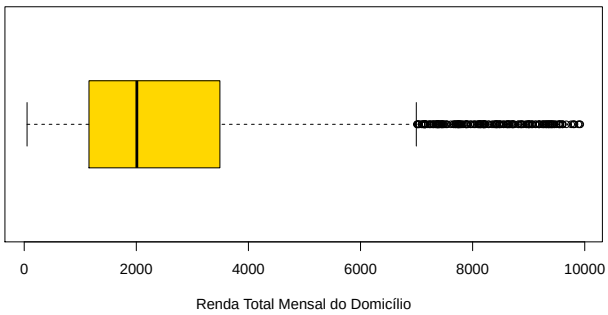


Figura 2.8: Boxplot para Renda Total do Domícilio (POF)

Repare que existem muitos dados acima do limite superior. Esses são considerados *outliers* ou valores atípicos, valores distantes da maioria dos dados. Através do boxplot também é possível ver a assimetria da distribuição dos dados, assim como foi visto no histograma.

Gráfico de Pontos ou Gráfico de Dispersão

Gráfico de pontos é útil quando se tem dados pareados (x,y) e se quer descobrir se há alguma relação entre eles. No R, utilizamos a função `plot()` para esboçar esse tipo de gráfico. Os principais

argumentos desta função estão a seguir.

- `x`: vetor de dados (necessário);
- `y`: vetor de dados;
- `main`: título do gráfico;
- `xlab`: texto do eixo x;
- `ylab`: texto do eixo y;
- `col`: cor dos pontos;
- `pch`: formato dos pontos;
- `xlim`: limites do eixo x;
- `ylim`: limites do eixo y;
- `type`: tipo de linha que liga os pontos (*default*: sem linha);
- `lty`: formato da linha que liga os pontos;
- `lwd`: espessura dos pontos.

Exemplo: Utilizaremos outra base de dados do pacote **UsingR**: `kid.weights`. Essa base representa uma amostra sobre 250 crianças. Contém a idade, o peso, a altura e o sexo da criança. Façamos um gráfico de pontos para verificar se há relação entre o peso (coluna 2) e a altura (coluna 3) das crianças.

```
> # base a ser utilizada
> head(kid.weights)
```

```
  age weight height gender
1  58     38     38      M
2 103     87     43      M
3  87     50     48      M
4 138     98     61      M
5  82     47     47      F
6  52     30     24      F
```

```
> # criando o gráfico
> plot(kid.weights$weight, kid.weights$height, main = "",xlab = "Peso", ylab = "Altura")
```

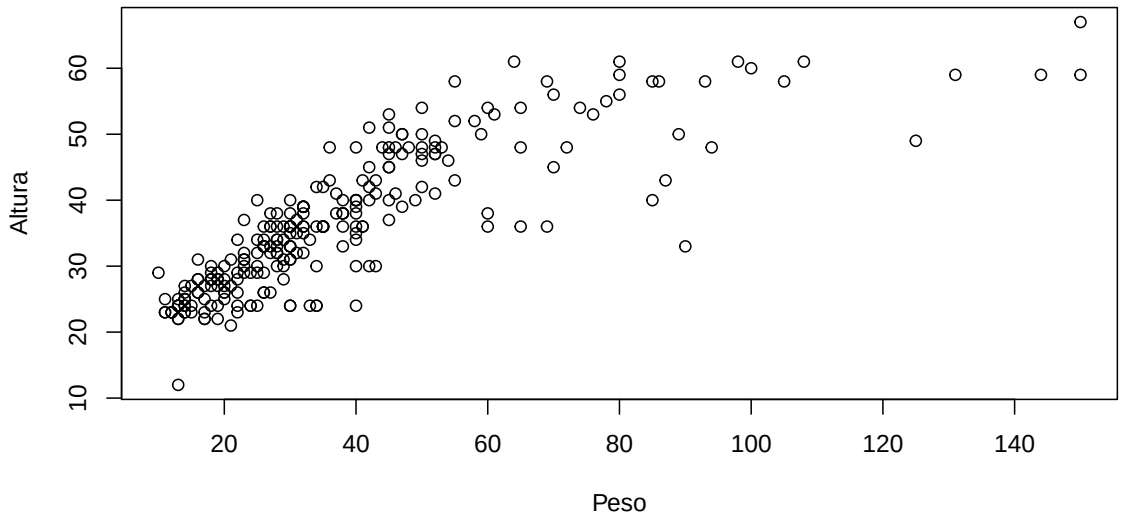


Figura 2.9: Gráfico de dispersão entre peso e altura das crianças (base `kid.wights`)

Podemos observar que há uma relação positiva entre o peso e a altura das crianças (quanto maior uma, maior a outra).

Utilizaremos os dados da POF para esboçar um gráfico de pontos que mostra se há alguma relação entre a quantidade de banheiros (coluna 20) e cômodos (coluna 16) em um domicílio.

```
> # criando o gráfico
> plot(dados[, 16], dados[, 20], main = "", pch = 19,
      col = "dodgerblue3",  xlab = "Quantidade de Cômodos no Domicílio",
      ylab = "Quantidade de Banheiros no Domicílio")
```

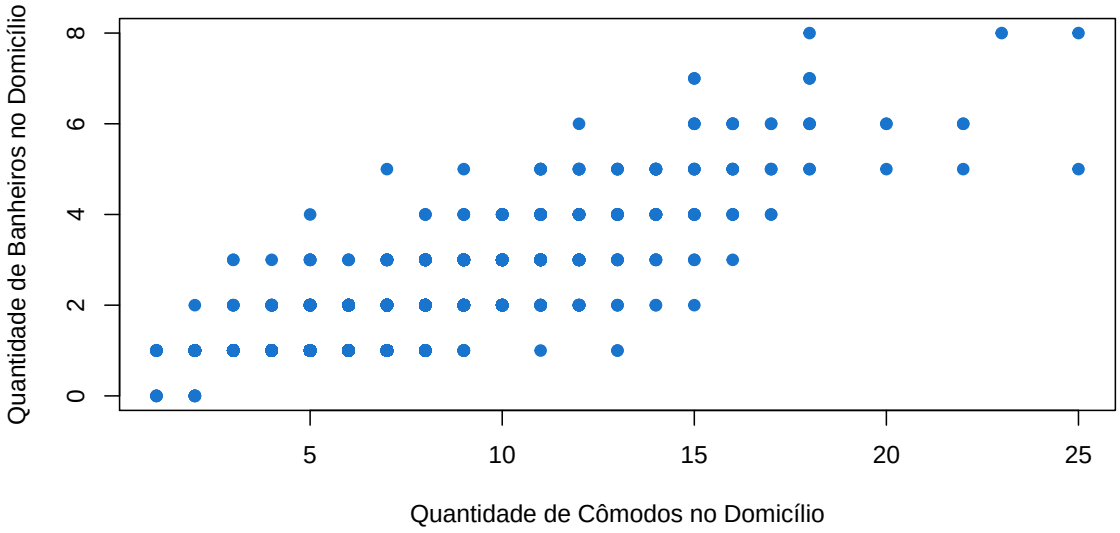


Figura 2.10: Gráfico de dispersão entre Quantidade de cômodos e Banheiros no Domicílio (POF)

Parece haver alguma relação positiva entre a quantidade de cômodos e de banheiros no domicílio: à medida que o número de cômodos cresce, o número de banheiros aumenta. Testes estatísticos podem confirmar essa relação.

Gráfico de Setores ou de Pizza

O gráfico de setores, mais conhecido como gráfico de pizza, é bastante usado para representar categorias de uma variável de acordo com suas proporções. Não é indicado quando se tem muitas categorias, nesse caso, é preferível utilizar um gráfico de barras.

No R, utiliza-se a função `pie()` para criar um gráfico de pizza. Os principais argumentos da função são:

- `x`: vetor com as frequências ou proporções de cada fatia;
- `main`: título do gráfico;
- `labels`: vetor de texto para cada fatia;
- `col`: vetor de cores para cada fatia.

Um gráfico com diversas categorias necessita de uma legenda para o entendimento do leitor. Para criar uma legenda no R, utiliza-se a função `legend()`. Os principais argumentos dessa função são

mostrados a seguir.

- `legend`: vetor de texto para cada fatia;
- `x`: posição da legenda: `"bottomright"`, `"bottom"`, `"bottomleft"`, `"left"`, `"topleft"`, `"top"`, `"topright"`, `"right"` e `"center"`;
- `fill`: vetor de cores de cada fatia;
- `cex`: tamanho da legenda.

Exemplo: Ainda utilizando a base `kid.weights` criaremos um gráfico de pizza que mostre a proporção de meninos e meninas na amostra (coluna 4).

```
> # tabela de frequência das categorias
> prop <- table(kid.weights[, 4])
> prop
  F   M
129 121
> # esboçando o gráfico
> pie(prop, main = "",  labels = c("51.6%", "48.4%"),
     col = c("palevioletred2", "dodgerblue3"))
> # adicionando a legenda
> legend(x = "topright", bty = "n", cex = 0.8,
        legend = c("Feminino", "Masculino"),
        fill = c("palevioletred2", "dodgerblue3"))
```

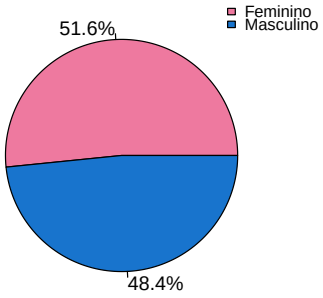


Figura 2.11: Gráfico de pizza para composição por sexo (base `kid.weights`)

Vemos (Figura 2.11) que a amostra é composta de um pouco mais meninas do que meninos.

Exemplo: Agora, utilizando a POF, esboçaremos um gráfico de pizza para que representa as proporções dos tipos de domicílios da amostra: casa, apartamento ou cômodo (cortiço).

```
> # tabela de frequência das categorias
> setores <- table(dados[, 12])
> # cálculo da porcentagem de cada categoria
> valores <- signif(setores/sum(setores)*100, 3)
> # construção do gráfico
> pie(setores, labels = paste(valores, "%", sep=""),
col = c("steelblue1","olivedrab3","orange"),main = "")
> # criando a legenda
> texto <- c("Casa", "Apartamento", "Cômodo")
> legend(x = "topright", legend = texto,
        fill = c("steelblue1","olivedrab3","orange"), cex =  0.65)
```

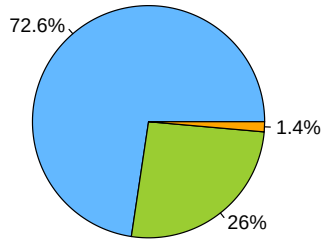


Figura 2.12: Gráfico de pizza para tipo de domicílio (POF)

Podemos ver (Figura 2.12) que a maioria dos domicílios são do tipo casa e apenas 1,4% dos domicílios são do tipo cômodo ou cortiço.

Gráfico de Barras

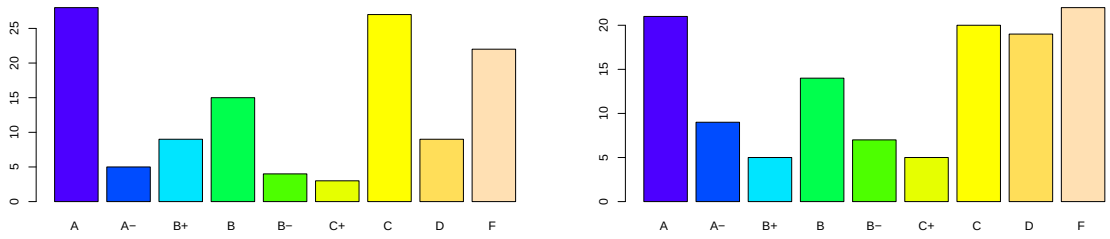
Um gráfico de barras mostra as frequências de diversas categorias de dados. É bastante útil para observar as diferenças entre as categorias.

No R, utiliza-se a função `barplot()` para esboçar um gráfico de barras. Os principais argumentos estão apresentados a seguir.

- `x`: vetor de dados com a frequência de cada categoria;
- `names`: vetor de texto para o nome de cada categoria / barra;
- `main`: título do gráfico;
- `col`: um vetor de cores para cada categoria.

Exemplo: utilizaremos outra base de dados do pacote UsingR: `grades`. Essa base representa uma amostra sobre notas de 122 alunos na classe atual e anterior na escala A-F. Façamos um gráfico de barras para as notas atuais (coluna 2) para verificar a frequência de cada nota nessa amostra.

```
> # dados a serem utilizados
> grades
> tabela1 <- table(grades[, 1])
> tabela2 <- table(grades[, 2])
> # criando o gráfico
> barplot(tabela1, main = "",
+         col = topo.colors(9))
> barplot(tabela2, main = "",
+         col = topo.colors(9))
```



(a) Frequências de notas anteriores

(b) Frequências de notas atuais

Figura 2.13: Frequências de notas (base `grade`)

Vejamos outro exemplo. Esboçaremos um gráfico de barras para os dados da POF para a renda média por Unidade de Federação. Precisamos calcular a renda média por UF antes de esboçar o gráfico.

```
> # calculando a renda média por UF
> renda <- tapply(dados[,31], dados[,2], mean)
> #criando o gráfico
> barplot(renda, names = c("BA", "MG", "RJ", "SP", "RS", "DF"),
          ylim = c(0, 5000), main = "",
          col = gray.colors(6))
```

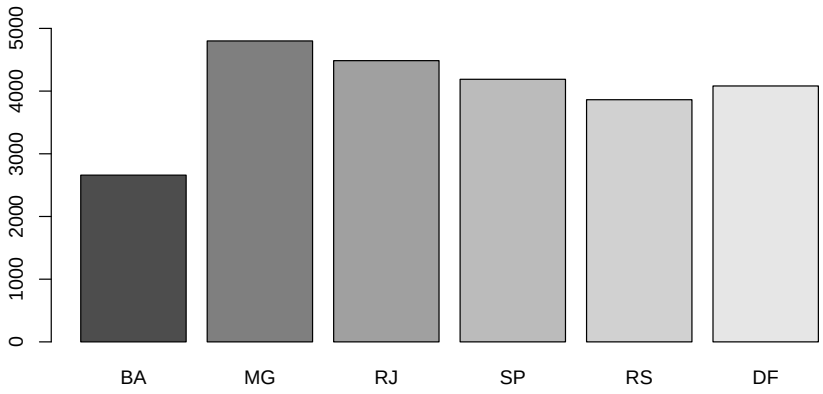


Figura 2.14: Gráfico de barras para Renda Média por UF (POF)

Por esse gráfico (Figura 2.14) podemos comparar a renda média em cada estado contido na amostra. Na Bahia, por exemplo, a renda média domiciliar não ultrapassa R$3.000,00. Enquanto em Minas Gerais, a renda média se aproxima dos 5 mil reais.

Pode ser feito também um gráfico de barras múltiplas. É ótimo para comparar diversas variáveis diferentes. Veja o exemplo a seguir. Façamos um gráfico de barras que mostre os tipos de domicílios de acordo com as unidades de federação. Nesse caso, foram inseridos os seguintes argumentos na função `barplot`:

- `beside`: use `TRUE` para colocar as barras lado a lado ou `FALSE` para colocar as barras uma em cima da outra;
- `legend.text`: para adicionar a legenda ao gráfico.

```
> # tabela de dados que mescla UF com os tipos de domicílios
> tabela <- table(dados[, 2], dados[, 12])
> # criando o gráfico
> barplot(tabela, names = c("Casa", "Apartamento", "Cômodo"),
+         main = "", beside = T,
+         col = terrain.colors(6),
+ legend.text = c("BA ", "MG", "RJ", "SP", "RS", "DF"))
```

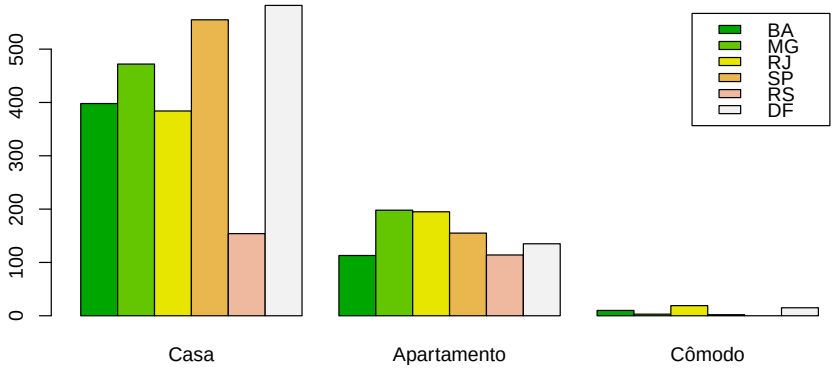


Figura 2.15: Gráfico de barras para Tipos de domicílio por UF (POF)

Na Figura (2.15), comparamos três categorias de tipo de domicílio de acordo com cada UF. Observando Rio de Janeiro (cor amarela) e São Paulo (cor laranja), percebe-se que SP é superior em número de casas em relação ao Rio. Mas em relação a apartamentos, SP é inferior.

Outros gráficos

Há ainda, outros gráficos que permitem descrever os dados. Abaixo será mostrado um breve resumo sobre eles.

• **Ramo–e–folhas**

O diagrama de ramo-e-folhas é uma boa maneira de organizar os dados a fim de obter uma apresentação que facilite a visualização de informações. Recomenda-se que seja utilizado para quantidades pequenas de dados, pois em grandes quantidades é possível a perda de informações.

No R, utiliza-se a função `stem()` para fazê-lo. Faremos o diagrama para a variável Renda Total Mensal do Domicílio (coluna 31).

```
> stem(dados[, 31])
```

```
The decimal point is 4 digit(s) to the right of the |

 0 | 00000000000000000000000000000000000000000000000000000000000000000000+3096
 1 | 00000000000000000000000000000000000000000111111111111111111111111111+165
 2 | 00000111111122222333333444444555555555556888899
 3 | 0122333333445556667888 8
 4 | 345567
 5 | 3
 6 | 29
 7 | 6
 8 | 8
 9 |
10 |
11 | 7
```

À esquerda de | está o ramo e à direita estão as folhas. O R nos informa que o ponto decimal está quatro dígitos à direita de |. Isso significa que os algarismos à direita de | representam a casa dos milésimos em um número. Porém, como os valores da renda são altos e a quantidade de dados é grande, perdem-se informações. Assim, o R trunca os números. Por exemplo, a maior observação que é 117219.2, é representada por 11|7 que é 117000, já que a folha 7 representa a casa dos milésimos.

• Distribuição Empírica

Um gráfico de distribuição empírica é útil para determinar a proporção de valores que estão abaixo de um número particular. Para construir esse gráfico, é necessário ordenar os dados. Utilizaremos a função `plot()` já vista anteriormente. Como exemplo, esboçaremos este gráfico para a variável Renda Total Mensal do Domicílio (coluna 31).

```
> # número de observações e vetor de proporções
> n <- length(dados[, 31])
> y <- (1:n)/n
> # ordenando a renda
> renda <- sort(dados[, 31])
> # esboçando o gráfico
> plot(renda, y, type = "S", xlab ="Renda Total Mensal do Domicílio",
+      ylab = "Probabilidade",
+      main = "")
```

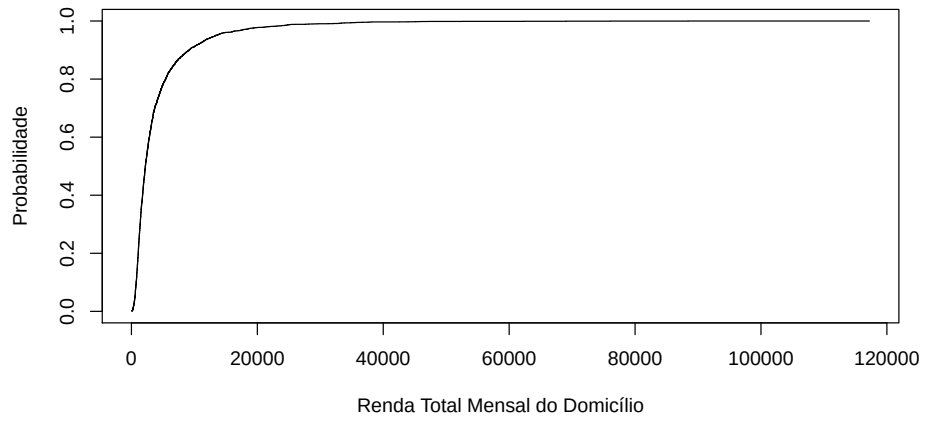


Figura 2.16: Distribuição Empírica da Renda dos Domicílios (POF)

Se fosse traçada uma reta para a renda R$10.000,00, através desse gráfico (Figura 2.16) veríamos que aproximadamente 90% dos dados estariam abaixo dele. Essa é a ideia de um gráfico de distribuição empírica.

Adicionaremos essas linhas ao gráfico para uma melhor visualização.

Adicionando elementos ao gráfico

Podemos adicionar pontos, linhas, curvas, textos e etc. ao gráfico para facilitar a visualização de algumas informações. Veremos a seguir como adicionar esses elementos.

• Adicionando linhas

No R, para adicionar linhas a um gráfico pronto, utiliza-se a função `abline()`. Os principais argumentos dessa função são:

- `v`: valores para linhas verticais;
- `h`: valores para linhas horizontais;
- `col`: vetor de cores para as linhas;
- `lty`: tipo de linha;
- `lwd`: espessura da linha.

Como exemplo, adicionaremos linhas horizontais e verticais ao gráfico anterior (Figura 2.16) que facilitem a visualização da proporção de domicílios com renda inferior a cinco e dez mil reais.

```
> # adicionando 2 linhas verticais e tracejadas
> abline(v = c(5000, 10000), col = c("tomato", "blue"), lty = 2)
> # adicionando 2 linhas horizontais e pontilhadas
> abline(h = c(0.8, 0.9), col = "darkgray", lty = 3)
```

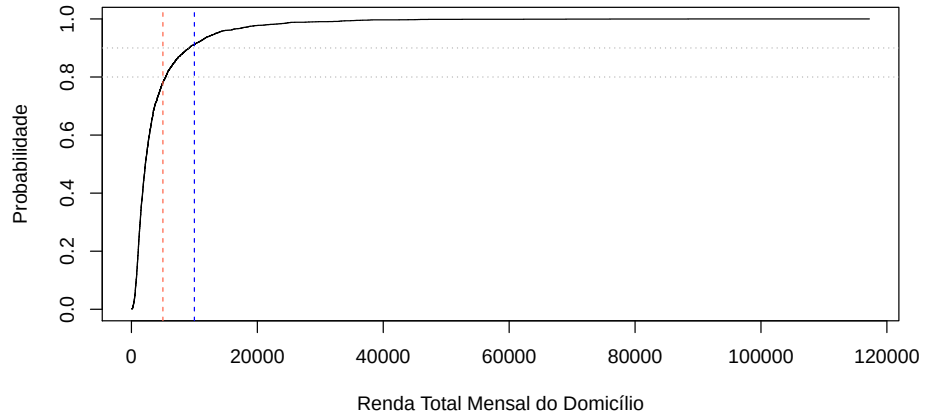


Figura 2.17: Distribuição Empírica da Renda dos Domicílios com linhas (POF)

As linhas tracejadas representam, respectivamente, renda igual a cinco e dez mil reais. Repare que a proporção de domicílios com renda inferior a cinco mil reais é 80%.

• Adicionando pontos

No R, para adicionar pontos a um gráfico pronto, utiliza-se a função `points()`. Os principais argumentos dessa função são:

- `x,y`: coordenadas (x,y) para posição do ponto;
- `col`: cor do ponto;
- `pch`: tipo do ponto.

Exemplo: Adicionaremos os pontos 5 e 10 mil ao gráfico de distribuição empírica.

```
> # adicionando 2 pontos ao gráfico
> points(5000, 0.8, col = "tomato", pch = 19)
> points(10000, 0.9, col = "blue", pch = 19)
```

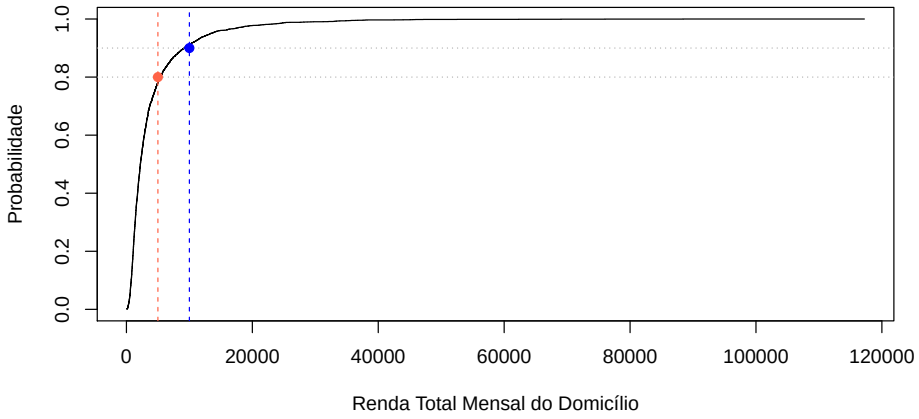


Figura 2.18: Distribuição Empírica da Renda dos Domicílios com pontos (POF)

• **Adicionando textos**

No R, para adicionar textos a um gráfico pronto, utiliza-se a função `text()`. Os principais argumentos dessa função são:

- `x,y`: coordenadas (x,y) para posição do texto;
- `labels`: texto a ser escrito;
- `pos`: (*default*: `NULL` (centro)). Opções: 1 (abaixo), 2 (à esquerda), 3 (acima) e 4 (à direita);
- `cex`: tamanho do texto;
- `col`: cor do texto.

Exemplo: Adicionaremos ao gráfico anterior, ao lado direito de cada ponto, os textos 5 mil e 10 mil.

```
> # adicionando 2 textos ao gráfico
> text(5000, 0.8, labels = "5 mil", col = "tomato", pos = 2)
> text(10000, 0.9, labels = "10 mil", col = "blue", pos = 4)
```

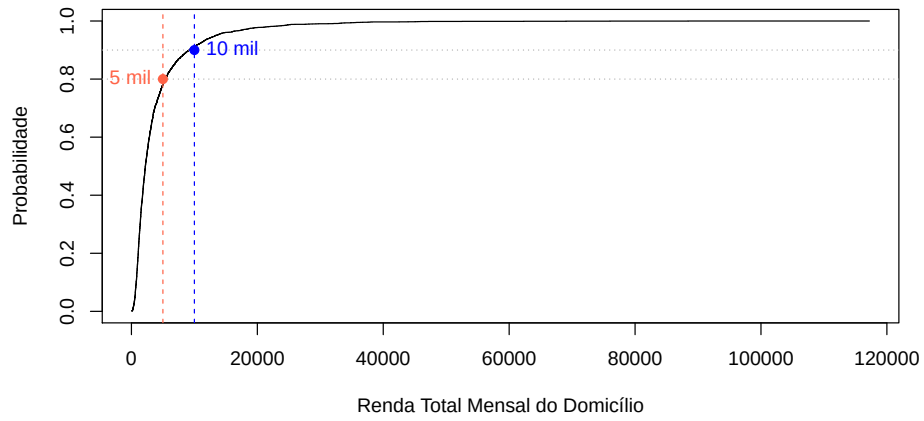


Figura 2.19: Distribuição Empírica da Renda dos Domicílios com texto (POF)

• **Adicionando títulos**

Há uma função no R que permite adicionar títulos a um gráfico de outra maneira. Esta função permite alterar o tipo e a cor da fonte tanto dos eixos quanto do título principal. Assim, você pode criar um gráfico sem títulos e adicioná-los depois. Utilizamos a função `title()`. Os principais argumentos da função `title()` são:

- `main`: título principal;
- `font.main`: tipo de fonte do título principal;
- `col.main`: cor da fonte do título principal;
- `cex.main`: tamanho do título principal;
- `xlab`: texto do eixo x;
- `ylab`: texto do eixo y;
- `font.lab`: tipo de fonte do texto dos eixos x e y;
- `col.lab`: cor da fonte do texto dos eixos x e y;
- `cex.lab`: tamanho do texto dos eixos x e y.

Exemplo: Suponha o gráfico apresentado a seguir na Figura 2.20 sem título e sem texto nos eixos.

```
> # Gráfico sem título
> plot(1:10, 1:10, xlab = "", ylab = "", pch = 19)
```

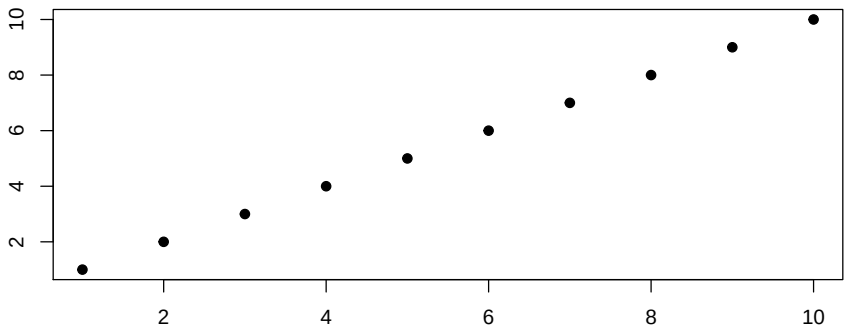


Figura 2.20: Gráfico sem título

```
> # Adicionando título ao gráfico
> title(main = "Título do gráfico", font.main = 8, col.main = "blue",
       xlab = "eixo x", ylab = "eixo y", font.lab = 10, col.lab = "red")
```

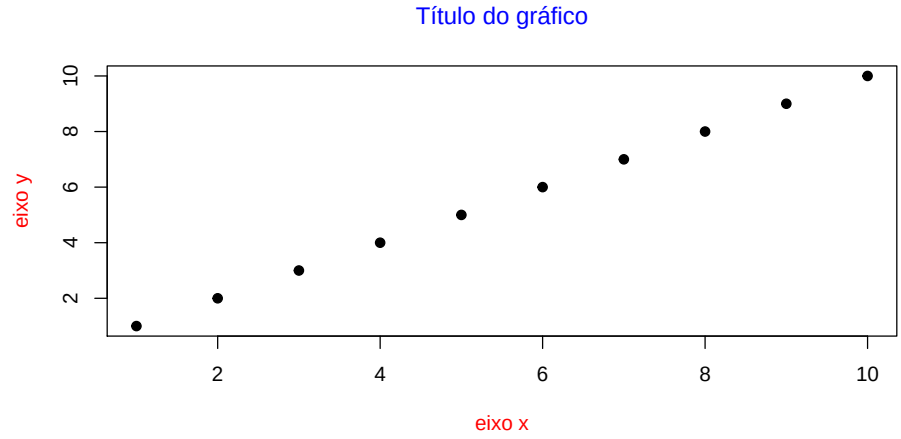


Figura 2.21: Gráfico com título

• **Esboçando retas**

A função `lines()` permite adicionar retas a um gráfico pronto. Parecida com a função `points()` vista anteriormente, a função `lines()` traça retas entre os pontos informados. Os principais argumentos da função `lines()` são:

- `x,y`: coordenadas (x,y) para os pontos;
- `type`: tipo de linha que liga os pontos (*default*: `"l"`).

Exemplo: Utilizando um gráfico já esboçado, adicionaremos uma reta nele.

```
> # Gráfico
> plot(0:10, 0:10, pch = 19, main = "Gráfico")
> # Adicionando 6 pontos que serão ligados por uma linha
> lines(c(0, 2, 4, 6, 8, 10), c(1, 4, 2, 8, 9, 4), type = "o")
```

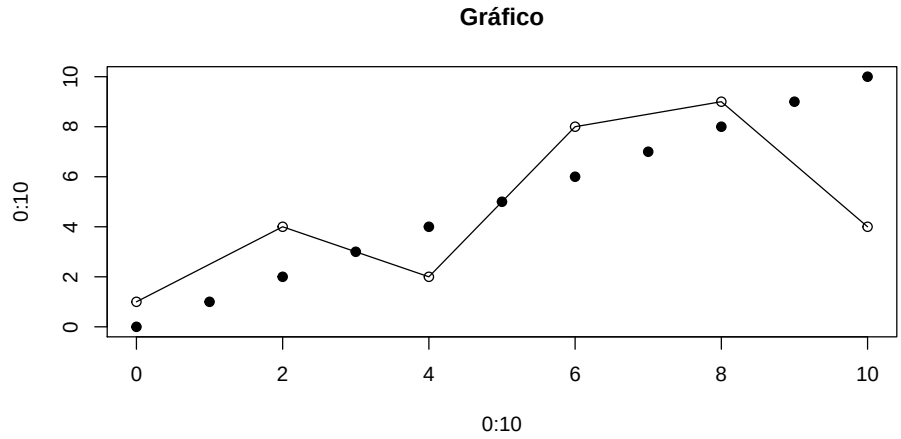


Figura 2.22: Gráfico com linhas adicionadas

• **Esboçando curvas ou expressões**

O R permite o esboço de expressões através da função `curve()`. Os principais argumentos da função `curve()` são:

- **expr**: expressão a ser desenhada;
- **from**: valor inicial;
- **to**: valor final;
- **add**: se `TRUE` esboça a expressão em um gráfico já existente;

Exemplo: Utilizando um gráfico já esboçado, adicionaremos nele uma expressão de uma função quadrática.

```
> # Gráfico
> plot(0:10, 0:10, pch = 19, main = "Gráfico")
> # Adicionando a expressão x^2 ao gráfico
> curve(x^2, from = 0 , to = 10 , add = T, col = "blue")
```

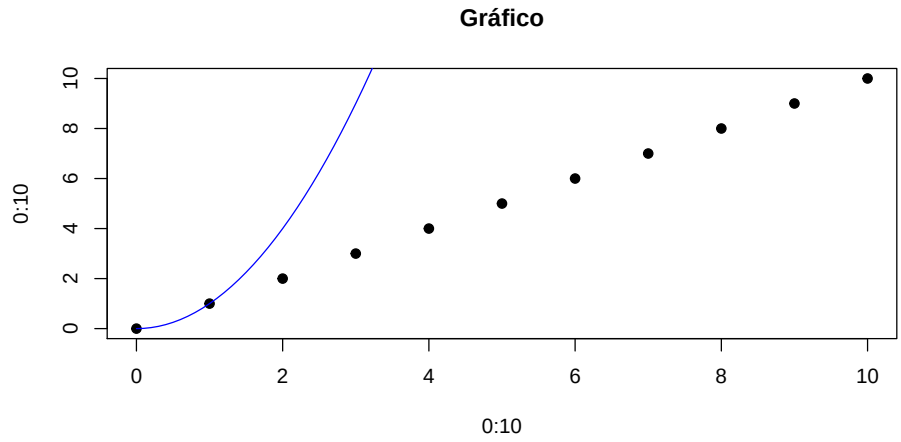


Figura 2.23: Gráfico com curvas adicionadas

Múltiplos Gráficos

No R, é possível esboçar diversos gráficos em uma janela ao mesmo tempo. Para isso, use a função `par()`. Essa função apresenta diversos argumentos, mas vamos nos limitar a apenas um: `mfrow`.

```
par(mfrow = c(número de linhas, número de colunas)
```

Exemplo: Esboçaremos quatro gráficos (já feitos nas seções anteriores) em uma só janela.

```
> # Exibe 4 gráficos na tela: 2 linhas e 2 colunas
> par(mfrow=c(2,2))
> # Gráfico 1 - Histograma
> hist(galton$child, main = "Distribuição das alturas dos filhos",
      xlab = "Alturas (cm)", col = "lightgray", border = "steelblue")
> # Gráfico 2 - Boxplot
> boxplot(galton$parent, main = "Boxplot para a altura dos pais",
         ylab = "Altura (cm)", col = "seagreen3")
> # Gráfico 3 - Pizza
> prop <- table(kid.weights[, 4])
> pie(prop, main = "Composição por sexo",  labels = c("51.6%", "48.4%"),
     col = c("palevioletred2", "dodgerblue3"))
> # Gráfico 4 - Barras
> renda <- tapply(dados[,31], dados[,2], mean)
> barplot(renda, names = c("BA", "MG", "RJ", "SP", "RS", "DF"),
         ylim = c(0, 5000), main = "Renda Média por UF",
         col = gray.colors(6))
```

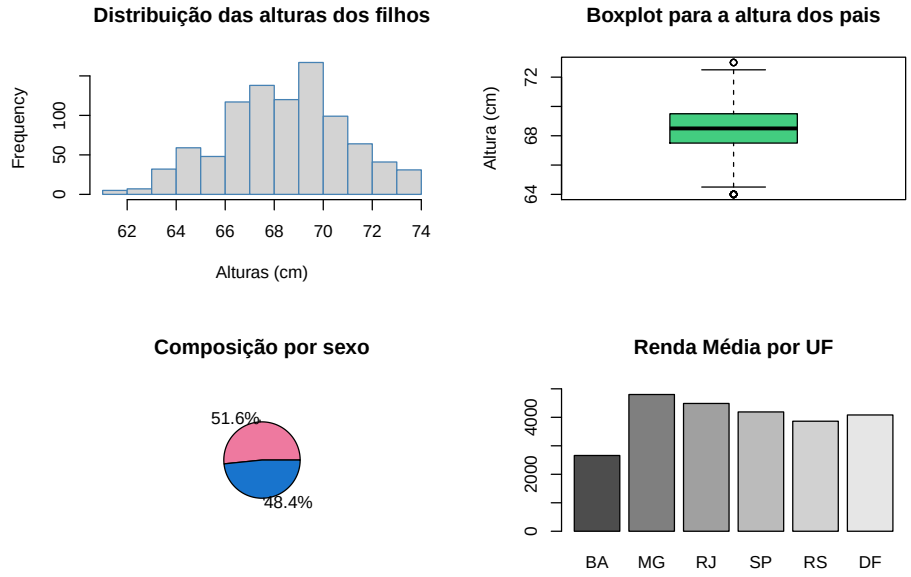


Figura 2.24: Mútiplos gráficos

O preenchimento dos gráficos na tela é por linha. Observe que quanto maior o número de gráficos, pior pode ficar a visualização dos mesmos.

Exercícios: Introdução ao R e Estatística Descritiva

Durante os meses de maio e junho de 2013, o Instituto de Pesquisa Econômica Aplicada(IPEA (2015)) realizou a pesquisa intitulada de "Tolerância social à violência contra a mulher". A pesquisa foi feita por amostragem, ou seja, foi selecionada uma parte da população do Brasil. A amostra é composta por 3810 indivíduos de ambos os sexos. E abrange os municípios metropolitanos e não-metropolitanos das cinco regiões (Norte, Nordeste, Centro- Oeste, Sudeste e Sul). Na pesquisa, frases foram lidas para os entrevistados, que em seguida deveriam dizer se concordavam total ou parcialmente, ou se discordavam total ou parcialmente ou se nem concordavam nem discordavam (neutralidade).

Os exercícios a seguir dependem da base de dados dessa pesquisa (`base_ipea.csv`). Você pode fazer o download em https://github.com/pedrocostaferreira/TSinR. É importante o leitor saber que a base de dados possui observações faltantes (`NA`). Assim, em algumas funções será necessária a inclusão do argumento `na.rm = TRUE`.

1. Defina o seu diretório de trabalho para o local onde se encontra a base de dados.
2. Importe a base de dados `base_ipea.csv` para o R .
3. A base é composta por quantas linhas e colunas?
4. Calcule a frequência de pessoas em cada região do Brasil.
5. Qual é a região mais frequente (moda)?
6. Qual é a idade da pessoa mais nova nessa amostra? E da mais velha?
7. Calcule a média, a mediana e a moda para a variável idade. A partir disso, o que você pode dizer sobre a distribuição dessa variável (assimétrica positiva, assimétrica negativa ou simétrica)?
8. Classifique as idades de acordo com as faixas etárias a seguir. (Crie uma nova coluna no *data frame* para essa classificação). A amostra é composta de mais Jovens, Adultos ou Idosos?

Idade (x)	Faixa Etária
$x \leq 29$	Jovens
$30 \leq x \leq 59$	Adultos
$x \geq 60$	Idosos

9. Calcule a média, a mediana, o primeiro quartil, o terceiro quartil e os valores máximo e mínimo para a variável "renda total de todos os moradores, parentes e agregados no último mês". Comente os resultados.

10. Interprete o primeiro e o terceiro quartis encontrados no item anterior.
11. Crie uma função que calcule o coeficiente de variação.
12. Calcule o coeficiente de variação para a variável idade e renda. Compare os dois coeficientes de variação.
13. Calcule o desvio-padrão para a renda de acordo com cada região do Brasil. Qual é a região que possui um comportamento mais homogêneo em relação à renda?
14. Crie um histograma para a variável "renda total de todos os moradores, parentes e agregados no último mês". Defina o título do gráfico como "Histograma para a renda total do domicílio" e o texto do eixo x como "Renda total do domicílio". Baseado no gráfico, você conclui que a distribuição da variável é assimétrica positiva, assimétrica negativa ou simétrica?
15. Crie um boxplot para a variável "número de moradores no domicílio, parentes e agregados". (Não se esqueça do título do gráfico). Baseado no gráfico, qual é o número mediano de moradores no domicílio? Há *outliers* (valores extremos)? Se sim, a partir de qual valor um domicílio é considerado com número de moradores extremos?
16. Crie um gráfico de dispersão (gráfico de pontos) para as variáveis "renda total de todos os moradores, parentes e agregados no último mês" e "renda total do chefe da família no último mês". (Não esqueça de colocar o título principal e o texto nos eixos). É possível notar alguma relação entre essas variáveis?
17. Crie uma tabela de frequências para a variável "sexo". Faça o mesmo para a variável "religião".
18. Utilizando a tabela de frequências para a variável "sexo", crie um gráfico de pizza para essa variável. (Não se esqueça do título do gráfico). Visualizando o gráfico, esta amostra é composta por mais homens ou mulheres?
19. Utilizando a tabela de frequências para a variável "religião", crie um gráfico de barras para essa variável. (Não se esqueça do título do gráfico). Visualizando o gráfico, qual é a religião mais frequente na amostra?
20. Utilizando a função `par()`, esboce os dois últimos gráficos na mesma janela.

Parte II

Análise de Séries Temporais:

Modelos Univariados

NAIVE, Médias Móveis e Modelo de Amortecimento Exponencial

Pedro Costa Ferreira

Victor Eduardo

Introdução

Este capítulo tem como objetivo apresentar algumas formas de suavizações e previsões utilizando o software R. Em cada seção abordaremos um modelo e explicaremos o que ele faz e quando deve ser utilizado. Veremos os modelos NAIVE (Ingênuo), Média Móvel e também alguns modelos de Amortecimento Exponencial.

Dentro do ambiente R existem séries de estudos disponveís nos diversos pacotes existentes do software. Hyndman (2015), autor do famoso pacote **forecast**, criou um banco de dados com vários exemplos didáticos disponíveis no pacote **fpp**. Caso o leitor queira consultar as bases de dados deste pacote, basta utilizar o comando `data(package= "fpp")`. O autor também possui o site "https://datamarket.com/"com milhares de séries disponíveis gratuitamente.

Para construção desse capítulo foi utilizado como base o livro de Morettin & Toloi (2006), Hyndman et al. (2008) e os pacotes **forecast** (Hyndman et al. (2012)), **TTR** (Ulrich et al. (2013)) e **ggplot2** (Wickham & Chang (2015)).

Modelo NAIVE

O modelo NAIVE, ou modelo ingênuo, é o modelo mais simples de previsões para uma série temporal. A previsão dele é igual ao valor da última observação ou, quando há sazonalidade, à previsão de um mês futuro é igual ao valor da última observação daquele mês.

Este é o modelo de referência para previsões pelo fato de não ter praticamente "nenhum custo", ser rápido e fácil de preparar, embora não tenha muita precisão. Em outras palavras, partimos normalmente do modelo NAIVE e só trocamos para outro se for mais vantajoso (custo-benefício). As equações de previsão do modelo Naive são apresentadas abaixo.

$$E[Y_{t+1} \mid Y_t] = Y_t (\text{sem sazonalidade}) \tag{3.1a}$$

$$E[Y_{t+12} \mid Y_t] = Y_t (\text{com sazonalidade}) \tag{3.1b}$$

Como exemplo de aplicação no R, utilizaremos as funções `naive()` e `snaive()` do pacote **forecast**. Estimaremos os valores futuros das séries temporais `gold` e `wineind`, que pertencem ao pacote **fpp**.

A série temporal `gold` refere-se aos preços diários do ouro em dólares americanos no período

de 1^o de janeiro de 1985 a 31 de março de 1989 e a série temporal wineind refere-se às vendas totais australianas de vinho por fabricantes de vinho em garrafas de 1 litro ou menos no período de jan/1980 a ago/1994.

Os principais argumentos das funções `naive()` e `snaive()` são:

- x: série temporal que queremos prever (argumento obrigatório) ;
- h: número de períodos de previsão (default = 10);
- level: nível de confiança para o intervalo de previsão (default = c(80,95)).

Este modelo é um caso particular de média móvel, onde o tamanho da "janela" é igual a um.

Para a avaliar o desempenho de um modelo é necessário recorrer a algumas medidas de acurácia. Em muitos casos recorre-se a medidas como MAPE e RMSE. A sigla MAPE significa *Mean Absolute Percentage Erro* e expressa a acurácia em porcentagem. Já a medida RMSE significa *Root Mean Square Erro*. Quanto menor o valor encontrado para o MAPE e RMSE, melhor é o desepenho do modelo estudado. No presente exemplo, estas medidas foram, respectivamente, 0,7737 e 6,0712.

Em relação à previsão, o valor previsto para os próximos dias foi 382,3 (o último valor observado).

```
> install.packages("forecast")
> library("forecast")
> summary(naive(gold,h=12))


Forecast method: Naive method

Model Information:
Series: x
ARIMA(0,1,0)

sigma^2 estimated as 36.86:  log likelihood=-3468.23
AIC=6938.46   AICc=6938.46   BIC=6943.47

Error measures:
                     ME     RMSE      MAE        MPE      MAPE
Training set 0.06703449 6.071223 3.080662 0.01051411 0.7736669
                  MASE      ACF1
Training set 1.001702 -0.306562

Forecasts:
     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
1109          382.3 374.5194 390.0806 370.4006 394.1994
1110          382.3 371.2966 393.3034 365.4717 399.1283
1111          382.3 368.8236 395.7764 361.6897 402.9103
1112          382.3 366.7388 397.8612 358.5012 406.0988
1113          382.3 364.9021 399.6979 355.6922 408.9078
1114          382.3 363.2415 401.3585 353.1526 411.4474
```

```
1115          382.3 361.7145 402.8855 350.8172 413.7828
1116          382.3 360.2932 404.3068 348.6435 415.9565
1117          382.3 358.9582 405.6418 346.6019 417.9981
1118          382.3 357.6956 406.9044 344.6709 419.9291
1119          382.3 356.4947 408.1053 342.8342 421.7658
1120          382.3 355.3473 409.2527 341.0793 423.5207
```

A figura (3.1) mostra que à previsão para qualquer tempo futuro é igual ao último valor observado e o intervalo de confiança vai aumentando com o tempo. Esse modelo desperdiça muita informação, ignorando todas as observações anteriores à última. Por outro lado, ele se adapta mais rápido à mudança de comportamento da série.

```
> plot(naive(gold,h=12),include=200,main="",xlab="Tempo",ylab="Dados")
```

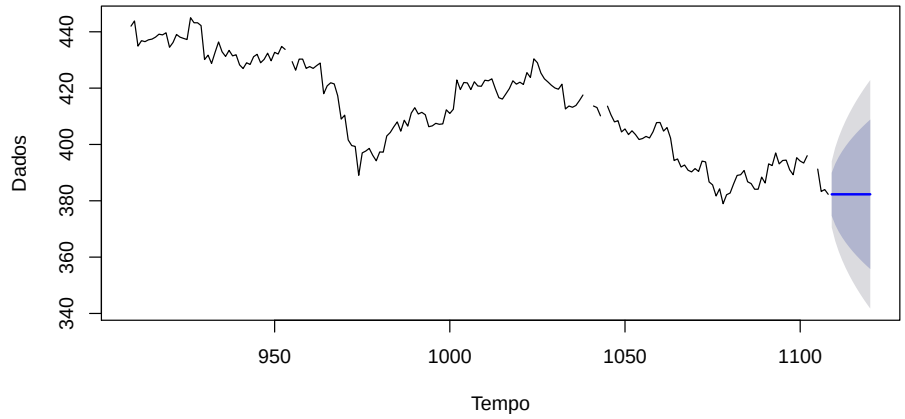


Figura 3.1: Previsão 12 passos à frente utilizando o modelo NAIVE

Com relação à série temporal de venda de vinho (`wineind`), o modelo Naive Sazonal apresentou melhores resultados quando comparado com o modelo Naive, isto é, as medidas de acurácia MAPE e RMSE foram menores para o modelo Naive Sazonal.

```
> summary(naive(wineind,h=12))

Forecast method: Naive method

Model Information:
Series: x
ARIMA(0,1,0)

sigma^2 estimated as 45850198:  log likelihood=-1791.89
AIC=3585.78   AICc=3585.81   BIC=3588.95

Error measures:
```

```
                    ME     RMSE      MAE       MPE     MAPE     MASE       ACF1
Training set 46.97143 6771.277 4833.726 -3.913764 21.32904 2.456938 -0.3002302

Forecasts:
         Point Forecast      Lo 80    Hi 80       Lo 95    Hi 95
Sep 1994          23356 14678.2588 32033.74  10084.5401 36627.46
Oct 1994          23356 11083.8207 35628.18   4587.3214 42124.68
Nov 1994          23356  8325.7113 38386.29    369.1571 46342.84
Dec 1994          23356  6000.5176 40711.48  -3186.9199 49898.92
Jan 1995          23356  3951.9807 42760.02  -6319.8866 53031.89
Feb 1995          23356  2099.9619 44612.04  -9152.3050 55864.30
Mar 1995          23356   396.8548 46315.15 -11756.9825 58468.98
Apr 1995          23356 -1188.3587 47900.36 -14181.3573 60893.36
May 1995          23356 -2677.2237 49389.22 -16458.3798 63170.38
Jun 1995          23356 -4085.4272 50797.43 -18612.0413 65324.04
Jul 1995          23356 -5424.8117 52136.81 -20660.4530 67372.45
Aug 1995          23356 -6704.5774 53416.58 -22617.6858 69329.69

> summary(snaive(wineind,h=12))

Forecast method: Seasonal naive method

Model Information:
Series: x
ARIMA(0,0,0)(0,1,0)[12]

sigma^2 estimated as 7259042:  log likelihood=-1528.12
AIC=3058.24   AICc=3058.27   BIC=3061.34

Error measures:
                    ME     RMSE      MAE       MPE     MAPE MASE       ACF1
Training set 355.0122 2694.261 1967.378 0.8684261 7.887751    1 0.08275386

Forecasts:
         Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
Sep 1994          22724 19271.17 26176.83 17443.346 28004.65
Oct 1994          28496 25043.17 31948.83 23215.346 33776.65
Nov 1994          32857 29404.17 36309.83 27576.346 38137.65
Dec 1994          37198 33745.17 40650.83 31917.346 42478.65
Jan 1995          13652 10199.17 17104.83  8371.346 18932.65
Feb 1995          22784 19331.17 26236.83 17503.346 28064.65
Mar 1995          23565 20112.17 27017.83 18284.346 28845.65
Apr 1995          26323 22870.17 29775.83 21042.346 31603.65
May 1995          23779 20326.17 27231.83 18498.346 29059.65
Jun 1995          27549 24096.17 31001.83 22268.346 32829.65
Jul 1995          29660 26207.17 33112.83 24379.346 34940.65
Aug 1995          23356 19903.17 26808.83 18075.346 28636.65
```

A figura (3.2) mostra que a previsão para qualquer tempo futuro é igual ao seu último ano observado, por exemplo, para qualquer mês de janeiro previsto, o resultado será igual ao último valor de janeiro observado, e o seu intervalo de confiança aumentará ao longo dos anos. Esse modelo só

será afetado por observações recorrentes de valores anteriores daquele período de tempo, e não por valores de períodos próximos. Mesmo aproveitando a informação de um ano, o modelo ainda perde informações importantes.

```
> plot(snaive(wineind),main="",xlab="Tempo",ylab="Dados")
```

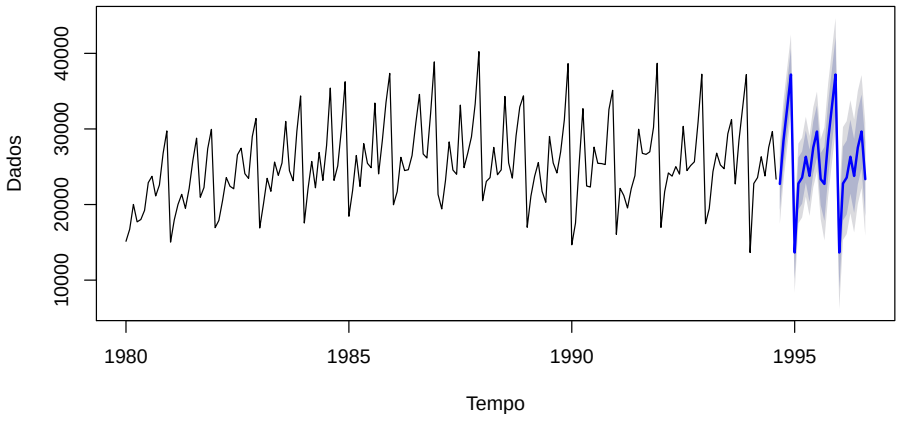


Figura 3.2: Previsão utilizando o modelo NAIVE

Média Móvel

Como o próprio nome já diz, média móvel é uma média que se movimenta, ou seja, todas as médias móveis têm como padrão um intervalo de período fixo ("janela") onde para cada nova informação que entrar, retira-se a mais antiga e recalcula-se a média. Esse tipo de modelo possui o objetivo de suavizar a série temporal, obtendo uma medida de tendência.

Média Móvel Centrada

Existem duas formas de calculá-la ao considerar o tamanho da "janela":

i. Quando n é impar, faz-se a média de n observações consecutivas, colocando o resultado exatamente na posição central:

$$Z_t = (Y_{t-m} + Y_{t-(m-1)} + ... + Y_{t+(m-1)} + Y_{t+m})/n \tag{3.2}$$

Onde $m = (n-1)/2$.

ii. Quando n é par, faz-se uma soma ponderada das $n+1$ observações consecutivas, sendo que a primeira e a última observação têm peso $1/(2n)$, as demais observações têm peso $1/(n)$. O resultado também é colocado exatamente na posição central.

$$Z_t = (Y_{t-m}/2 + (Y_{t-(m-1)} + ... + Y_{t+(m-1)}) + Y_{t+m}/2)/n \tag{3.3}$$

Onde $m = n/2$.

Para calcular a média móvel centrada no R, usaremos a função `ma()` do pacote **forecast**. Suavizaremos a série temporal `wineind`.

Os principais argumentos da função `ma()` são:

- x: série temporal que queremos suavizar (argumento obrigatório) ;
- order: tamanho da "janela" (argumento obrigatório).

Lembre-se que o pacote **forecast** precisa estar instalado e carregado para usarmos a função. Vejamos a série temporal `wineind` suavizada utilizando média móvel centrada.

```
> mm_centrada_6 <- ma(wineind,order=6)
> mm_centrada_12 <- ma(wineind,order=12)
> summary(mm_centrada_6)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   18450   24010   25740   25530   26980   30730       6


> summary(mm_centrada_12)

  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   21140   25010   25860   25620   26620   27810      12
```

Na figura (3.3) é possível notar que quanto maior é a "janela", mais suavizada ficará a série, e quanto menor é a "janela", mais próximo da série original fica o ajuste. Observe também que a amplitude é maior quando a "janela" é menor.

```
> plot(wineind,xlab="Tempo",ylab="Dados")
> lines(mm_centrada_6,col="red",lty=5,lwd =2)
> lines(mm_centrada_12,col="blue",lty=1,lwd =3)
> legend('topleft', legend=c("wineind", "mm_centrada_6","mm_centrada_12"),
         bty = "n",col=c("black","red", "blue"), lty=c(1,5,1), cex=0.8,
         lwd =c(1,2,3))
```

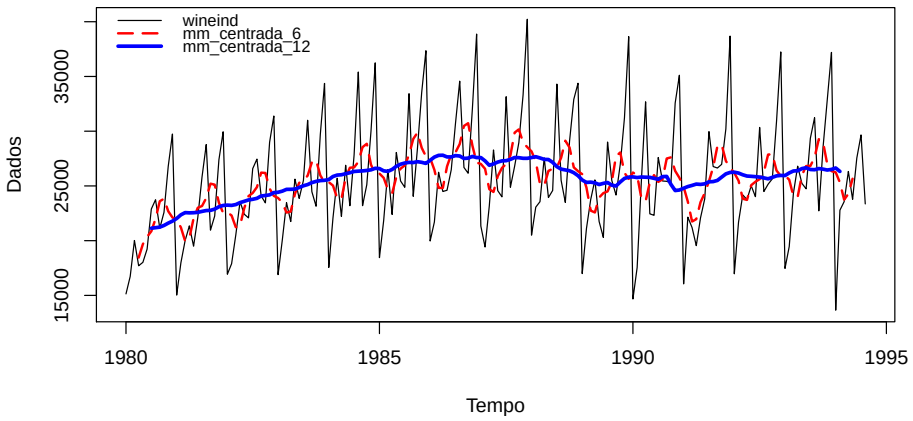


Figura 3.3: Ajuste da Série wineind utilizando média móvel

Para obter uma série suavizada sem influênica da sazonalidade, foi utilizada uma "janela" de tamanho 12. A figura (3.4) mostra uma tendência de crescimento entre os anos 1980 e 1987.

```
> plot(wineind,xlab="Tempo",ylab="Dados")
> mm_centrada_12 <- ma(wineind,order=12)
> lines(mm_centrada_12,col="red")
```

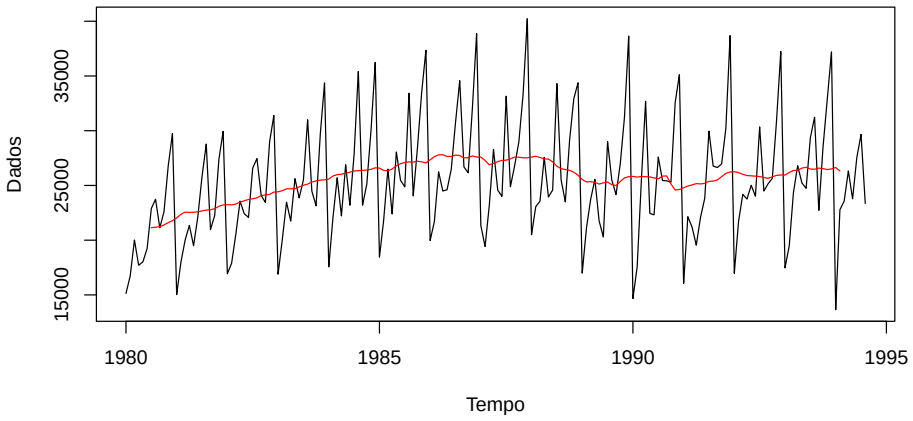


Figura 3.4: Ajuste da Série wineind utilizando média móvel

Média Móvel Simples

Para calcular a média móvel simples, só é preciso fazer a média aritmética das r observações mais recentes, ou seja:

$$M_t = \frac{Z_t + Z_{t-1} + \cdots + Z_{t-r+1}}{r} \tag{3.4}$$

ou

$$M_t = M_{t-1} + \frac{Z_t - Z_{t-r}}{r} \tag{3.5}$$

A útima média móvel é a previsão para todos os valores futuros, isto é:

$$\hat{Z}_t(h) = M_t \quad \forall h > 0 \tag{3.6}$$

ou

$$\hat{Z}_t(h) = \hat{Z}_{t-1}(h+1) + \frac{Z_t - Z_{t-r}}{r} \quad \forall h > 0 \tag{3.7}$$

Para calcular a média móvel simples no R, usaremos a função a `SMA()` do pacote `TTR`. Suavizaremos e preveremos a série temporal `wineind`, mas antes devemos instalar e carregar os pacotes que iremos utilizar:

```
> install.packages("TTR")
> install.packages("forecast")
> install.packages("ggplot2")
```

```
> library("TTR")
> library("forecast")
```

Para não ter influência da sazonalidade da série, utilizaremos uma "janela" de tamanho 12.

```
> # z = Série temporal
> # r = Tamanho da "janela"
> # l = Número de passos a frente
>
> z <- wineind
> l <- 12
> r <- 12
```

A função $IC_MMS()$ retorna a previsão do modelo com o intervalo de confiança.

```
> IC_MMS  <- function(z,r,l){
+   smadf <- SMA(z,r)
+   IC_I <-  rep(smadf[length(z)],l) - 1.96*sd(z)/sqrt(r)
+   IC_S <-  rep(smadf[length(z)],l) + 1.96*sd(z)/sqrt(r)
+   Previsao <-  rep(smadf[length(z)],l)
+   cbind(IC_I,Previsao,IC_S)
+ }
```

```
> IC_MMS(z,r,l)

         IC_I Previsao    IC_S
 [1,] 22973.4 25995.25 29017.1
 [2,] 22973.4 25995.25 29017.1
 [3,] 22973.4 25995.25 29017.1
 [4,] 22973.4 25995.25 29017.1
 [5,] 22973.4 25995.25 29017.1
 [6,] 22973.4 25995.25 29017.1
 [7,] 22973.4 25995.25 29017.1
 [8,] 22973.4 25995.25 29017.1
 [9,] 22973.4 25995.25 29017.1
[10,] 22973.4 25995.25 29017.1
[11,] 22973.4 25995.25 29017.1
[12,] 22973.4 25995.25 29017.1
```

Criaremos o gráfico da série temporal com sua suavização, previsão e intervalo de confiança de 95%.

```
> library(ggplot2)
> a <- IC_MMS(z,r,l)
> b <- c(z,a[,2])
> smadf <- SMA(z,r)
> grafico <- ggplot(data=data.frame(b))+ geom_line(aes(c(1:length(b)),b))+
+   geom_smooth(data=data.frame(a),
+               aes(ymin = IC_I, ymax = IC_S,x = c((length(b)-l+1):length(b)),
+                   y = Previsao), stat="identity")+
+   geom_line(data=data.frame(a),aes(c((length(b)-l+1):length(b)),IC_S))+
+   geom_line(data=data.frame(a),aes(c((length(b)-l+1):length(b)),IC_I))+
+   geom_line(data = data.frame(smadf),aes(c(1:(length(b)-l)),smadf),
+             na.rm = T,col="red")+
+   labs(title =" Previsão", x = "Tempo", y = "Dados")
```

```
> grafico
```

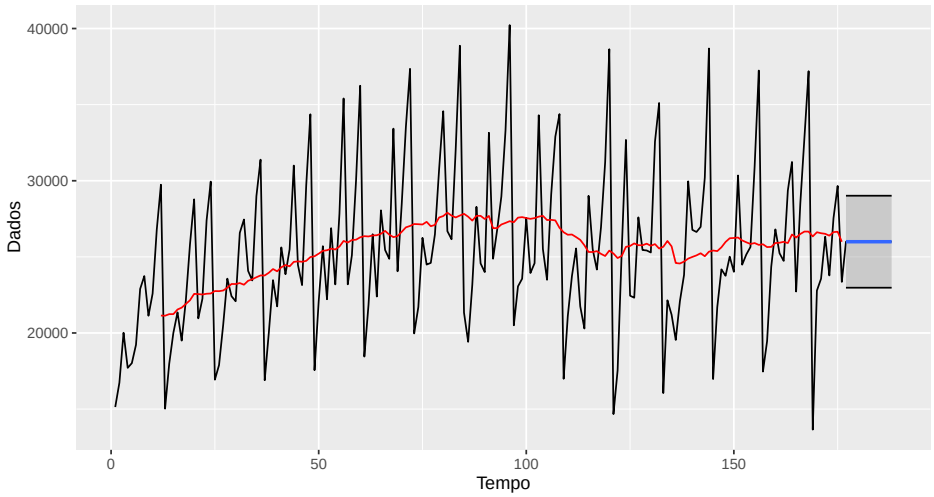


Figura 3.5: Previsão para a série wineind utilizando janela de tamanho 12

A figura (3.5) mostra a série suavizada que começa no 12º mes. A previsão para os meses futuros é 25995.25 (última média móvel de um período de 12 meses).

Modelos de suavização exponencial

Os estudos de métodos de Amortecimento Exponencial tiveram início na década de 1950 e desde então são utilizados com frequência quando o desejo é realizar ajustes e previsões em séries que apresentam tendência e sazonalidade.

Na prática é comum encontrar séries que apresentam tendência ou sazonalidade ou uma combinação destas duas componentes. É possível definir a componente de tendência (T) como um termo que segue uma direção no decorrer do tempo. Já a componente de sazonalidade (S) pode ser definida como padrões que repetem com uma certa periodicidade.

Os modelos estudados seguem como proposta estruturas aditivas e multiplicativas. Estas estruturas nada mais são do que combinações destas componentes. Por exemplo, o modelo aditivo é definido da seguinte maneira:

$$Z_t = T + S + E$$

Já a estrutura do modelo multiplicativo é dado do seguinte modo:

$$Z_t = T \times S \times E$$

É importante ressaltar que ambos os modelos apresentam uma componente de erro (E), comumente descrito na literatura por ε_t, que representa a parte não capturada do modelo.

Suavização Exponencial Simples (SES)

O modelo mais simples, conhecido como Suavização Exponencial Simples (SES) é adequado para séries temporais livres das componentes de tendência e sazonalidade. Muito embora seja difícil observar séries sem estas componentes, é possível utilizar a modelagem ao identificar e remover tais efeitos produzindo séries estacionárias.

Dada uma série estacionária, isto é, sem a presença de tendência e sazonalidade, o modelo SES é descrito da seguinte forma:

$$\bar{Z}_t = \alpha Z_t + \alpha(1-\alpha)Z_{t-1} + \alpha(1-\alpha)^2 Z_{t-2} + ..., \quad \text{onde:} \quad \bar{Z}_0 = Z_1, t = 1, ..., N \tag{3.8}$$

Onde α representa a constante de suavização e compreende valores entre 0 e 1. Quanto menor for a constante de suavização, mais estáveis serão as previsões, isto é, serão atribuídos maiores pesos para as observações passadas do que as mais recentes.

A principal vantagem do modelo (3.8) é dada pela sua simplicidade de implementação e por não necessitar de uma grande quantidade de informação do histórico.

Previsão

A literatura nos diz que a previsão de valores futuros para o modelo SES consiste no último valor exponencialmente suavizado denominado por:

$$\hat{Z}_t(h) = \bar{Z}_t \tag{3.9a}$$

$$\hat{Z}_t(h) = \alpha Z_t + \alpha(1-\alpha)\hat{Z}_{t-1}(h+1) \tag{3.9b}$$

Onde h representa o número de passos à frente a ser previsto.

Note que a equação (3.9) utiliza apenas a constante de suavização exponencial, informação mais recente e sua previsão feita no passo anterior.

Um conceito importante dentro de previsões de séries temporais é estabelecer a estimativa intervalar para os valores futuros em um determinado horizonte de tempo. Segundo (Morettin & Toloi (2006)), é possível representar a média e variância para $\hat{Z}_t(h)$ quando $t \to \infty$ como:

$$E(\hat{Z}_t(h)) = \mu \tag{3.10a}$$

$$Var(\hat{Z}_t(h)) = \frac{\alpha}{2-\alpha}\sigma_\varepsilon^2 \tag{3.10b}$$

Supondo que $\varepsilon_t \sim N(0, \sigma_\varepsilon^2)$, o intervalo de confiança para Z_{t+h} é dado por

$$\left[\hat{Z}_t(h) - z_(\gamma)\sigma_\varepsilon^2\sqrt{\frac{\alpha}{2-\alpha}}; \hat{Z}_t(h) + z_(\gamma)\sigma_\varepsilon^2\sqrt{\frac{\alpha}{2-\alpha}}\right] \tag{3.11}$$

Onde $z_(\gamma)$ é gerado através da distribuição $N(0,1)$.

Exemplo de Aplicação

No R, podemos utilizar a função `HoltWinters()` do pacote forecast para estimar o modelo SES. Lembre-se de instalar e carregar o pacote forecast para utilizar a função. Devido à dificuldade de encontrar na prática séries estacionárias, isto é, séries livres de tendência e sazonalidade, criaremos a seguinte série:

```
> set.seed(1234)
> serie <- ts(runif(100,10,15),start = c(1915,1),frequency = 1)
```

A função `HoltWinters()` indica que o valor determinado para a constante α é dado por 0,1016823, isto significa que informações recentes explicam pouco os movimentos da série.

```
> ajuste<-HoltWinters(serie,beta=FALSE,gamma=FALSE)
Holt-Winters exponential smoothing without trend and without seasonal component.

Call:
HoltWinters(x = serie, beta = FALSE, gamma = FALSE)

Smoothing parameters:
 alpha: 0.1016823
 beta : FALSE
 gamma: FALSE
```

```
Coefficients:
      [,1]
a 11.92001
```

Portanto o modelo (3.9) é representado da seguinte maneira:

$$\hat{Z}_t(h) = 0.10Z_t + 0.90\hat{Z}_{t-1}(h+1) \tag{3.12}$$

Parar gerar o gráfico da série em conjunto com a suavização encontrada, basta utilizar a função `plot`. A figura (3.6) ilustra a situação.

```
> plot(ajuste)
```

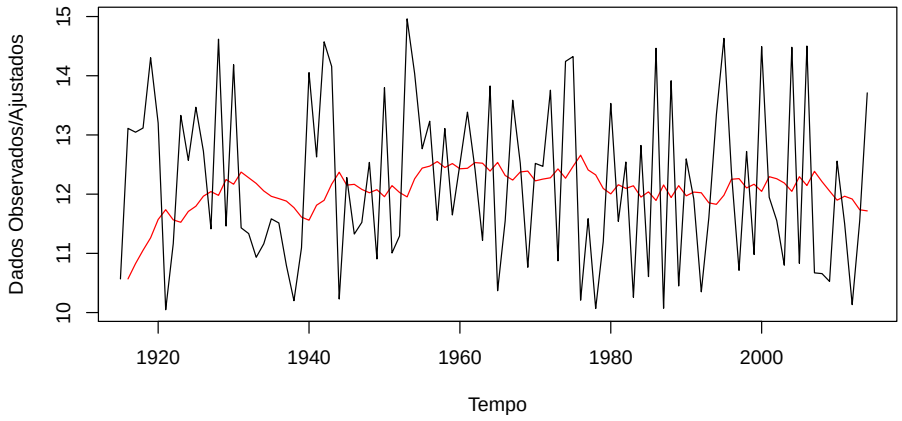


Figura 3.6: Série ajustada através do Modelo SES

As previsões para a série em questão são obtidas através da função `forecast.HoltWinters()`.

```
> forecast.HoltWinters(ajuste,h = 10,level=95)
     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2015       11.92001 9.988344 13.85168 8.965780 14.87425
2016       11.92001 9.978384 13.86164 8.950546 14.88948
2017       11.92001 9.968474 13.87155 8.935391 14.90464
2018       11.92001 9.958615 13.88141 8.920312 14.91972
2019       11.92001 9.948804 13.89122 8.905309 14.93472
2020       11.92001 9.939043 13.90099 8.890380 14.94965
2021       11.92001 9.929329 13.91070 8.875524 14.96450
2022       11.92001 9.919663 13.92037 8.860740 14.97929
2023       11.92001 9.910043 13.92999 8.846028 14.99400
2024       11.92001 9.900468 13.93956 8.831385 15.00864

plot.forecast(forecast.HoltWinters(ajuste,h = 10,level=95),main="",
xlab="Tempo",ylab="Dados")
```

Note que um dos argumentos da função representa o número de valores futuros a serem gerados. Para o presente exemplo este número foi fixado em 10. Como a série é anual, isto significa estamos realizando previsões para os próximos 10 anos. Além disso, outro *output* da função refere-se ao intervalo de confiança com nível 95%. Caso o leitor queira gerar outro nível de confiança, basta alterar o argumento da função para o valor desejado. Por fim, a figura (3.7) representa o gráfico da função em conjunto com sua previsão:

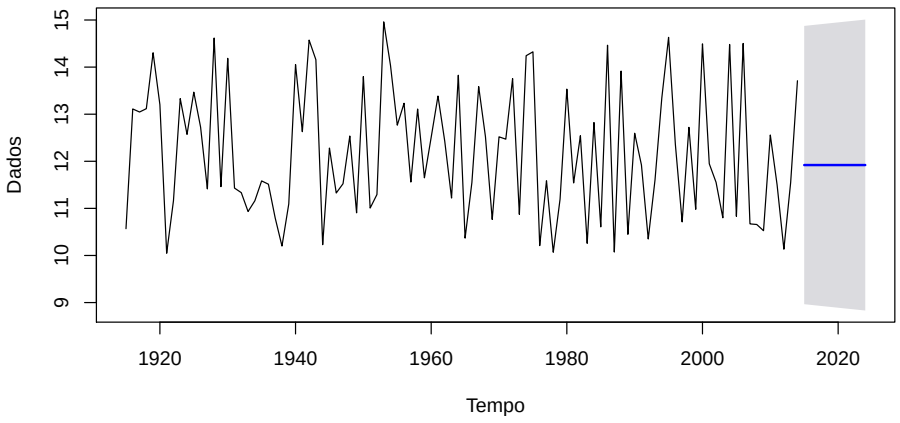


Figura 3.7: Previsão 10 passos à frente utilizando o Modelo SES

Suavização Exponencial de Holt (SEH)

O método visto anteriormente não é apropriado para séries que apresentam tendência. Uma possível aplicação do modelo SES acarretaria em previsões que subestimam ou superestimam continuamente os valores reais (Morettin & Toloi (2006)). Para contornar tal problema, o modelo de suavização exponencial de Holt (SEH), estendido Holt em 1957, permite realizar previsões em séries que apresentam o efeito de tendência.

O modelo SEH é similar ao modelo SES, com a diferença de que teremos uma nova constante de suavização para trabalhar diretamente com a tendência da série. Seus valores serão estimados por:

$$L_t = \alpha Z_t + (1-\alpha)(L_{t-1} + T_{t-1}) \quad 0 \leq \alpha \leq 1; t = 2, ..., N \tag{3.13}$$

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \quad 0 \leq \beta \leq 1; t = 2, ..., N \tag{3.14}$$

Aqui, α e β são classificadas como constantes de suavização, L_t estima o nível da série no tempo t e T_t estima a tendência no tempo t. A previsão para Z_{t+h} é dada por:

$$\hat{Z}_h = L_t + hT_t \quad \forall h > 0 \tag{3.15}$$

Exemplo de Aplicação 1

Para ilustrar a presente seção, utilizaremos a base de dados `ausair`, que representa o total de passageiros anuais de transportadoras aéreas registrada na Austrália. Estes dados são encontrados no pacote **fpp** (Hyndman (2015)).

```
> data(ausair)
> ausair
> ajuste_com_tendencia<-HoltWinters(ausair,gamma=FALSE)
Holt-Winters exponential smoothing with trend and without seasonal component.

Call:
HoltWinters(x = ausair, gamma = FALSE)

Smoothing parameters:
 alpha: 0.9934787
 beta : 0.1082122
 gamma: FALSE

Coefficients:
       [,1]
a 50.048209
b  1.452627
```

Aqui, os valores determinados para as constantes α e β são, respectivamente, 0,9934787 e 0,1082122. Isto significa que observações recentes tem um peso maior para o nível, enquanto a tendência representa o oposto. Portanto, as equações (3.13) e (3.14) são reescritas como:

$$L_t = 0.995Z_t + 0.007(L_{t-1} + T_{t-1}) \quad 0 \leq \alpha \leq 1; t = 2, ..., N \tag{3.16}$$

$$T_t = 0.108(L_t - L_{t-1}) + 0.892T_{t-1} \quad 0 \leq \beta \leq 1; t = 2, ..., N \tag{3.17}$$

Dando sequência a análise, a figura (3.8) representa o ajuste da série utilizando o modelo SEH.

```
plot(ajuste_com_tendencia,main="",xlab="Tempo",ylab="Dados Observados/Ajustados")
```

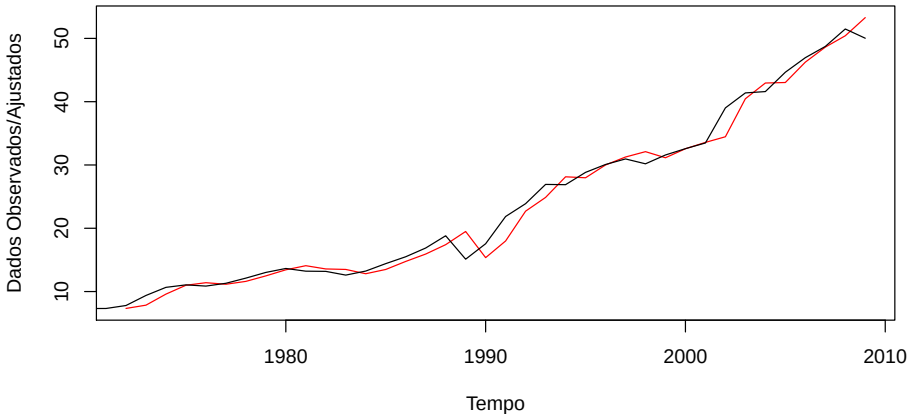


Figura 3.8: Previsão 10 passos a frente utilizando o Modelo SES

Para calcular os valores previstos, utilizaremos a função `holt()`, que trabalha exclusivamente com previsões de modelos SEH. A figura (3.9) representa a previsão para os próximos 10 anos da série de estudo.

```
> prev_serie_tendencia<- holt(ausair, h=10, level=95)
> prev_serie_tendencia
     Point Forecast    Lo 95    Hi 95
2010       51.07289 48.05012 54.09567
2011       52.11855 47.84344 56.39365
2012       53.16420 47.92798 58.40042
2013       54.20985 48.16323 60.25647
2014       55.25550 48.49479 62.01622
2015       56.30116 48.89474 63.70758
2016       57.34681 49.34650 65.34712
2017       58.39246 49.83928 66.94565
2018       59.43812 50.36557 68.51066
2019       60.48377 50.91992 70.04762
> plot(prev_serie_tendencia,main="",xlab="Tempo",ylab="Dados")
```

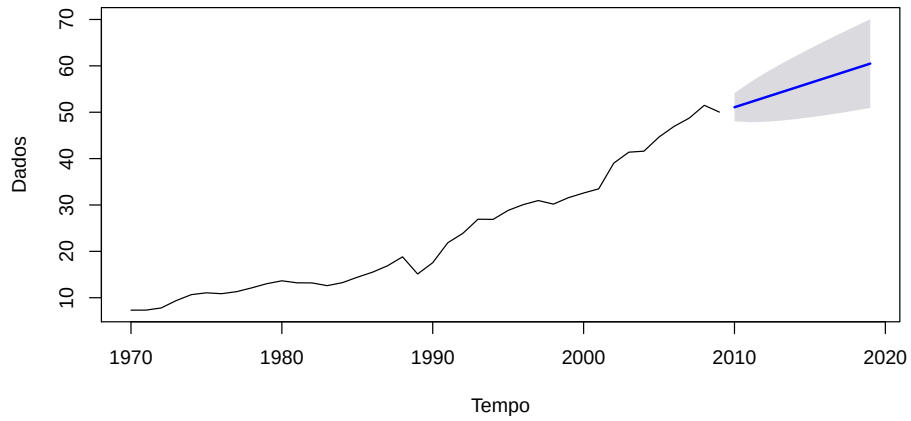


Figura 3.9: Previsão 10 passos a frente utilizando o Modelo SEH

Exemplo de Aplicação 2

Neste exemplo utilizaremos a série `airmiles` que se refere às milhas dos voos dos passageiros pelas companhias aéreas comerciais nos Estados Unidos. Os dados são anuais de 1937 a 1960, logo não têm sazonalidade, mas há tendência.

```
> airmiles

Time Series:
Start = 1937
End = 1960
Frequency = 1
 [1]   412   480   683  1052  1385  1418  1634  2178  3362  5948  6109  5981
[13]  6753  8003 10566 12528 14760 16769 19819 22362 25340 25343 29269 30514
```

Usando a função `HoltWinters()` e definindo o parâmetro γ como FALSE, descobrimos que os melhores valores para as constantes α e β são, respectivamente, 0,8072 e 0,3895. Note que o fato de α estar mais próximo de 1 indica que a estimativa do nível tem mais influência das observações mais recentes, e β estar mais próximo de 0 indica que a estimativa da tendência tem pouca influência dos valores mais recentes.

```
> ajuste_com_tendencia<-HoltWinters(airmiles, gamma=FALSE)
> ajuste_com_tendencia

Holt-Winters exponential smoothing with trend and without seasonal component.

Call:
HoltWinters(x = airmiles, gamma = FALSE)
```

```
Smoothing parameters:
 alpha: 0.8072924
 beta : 0.3895832
 gamma: FALSE

Coefficients:
       [,1]
a 30668.871
b  2100.563
```

O modelo encontrado foi:

$$L_t = 0,81Z_t + 0,19(L_{t-1} + T_{t-1}) \quad t = 3,4,\cdots,N \tag{3.18a}$$

$$T_t = 0,39(L_t - L_{t-1}) + 0,61T_{t-1} \quad t = 3,4,\cdots,N \tag{3.18b}$$

Para construir o gráfico da série original em conjunto com o modelo ajustado via SEH, utiliza-se o seguinte comando:

```
> plot(airmiles,xlab="Tempo",ylab="Dados")
> lines(fitted(ajuste_com_tendencia)[,1],col="red",lty=2,lwd =3)
> legend('topleft', legend=c("airmiles", "ajuste_com_tendencia"),bty = "n",
         col=c("black","red"), lty=c(1,2), cex=0.8,lwd =c(1,3))
```

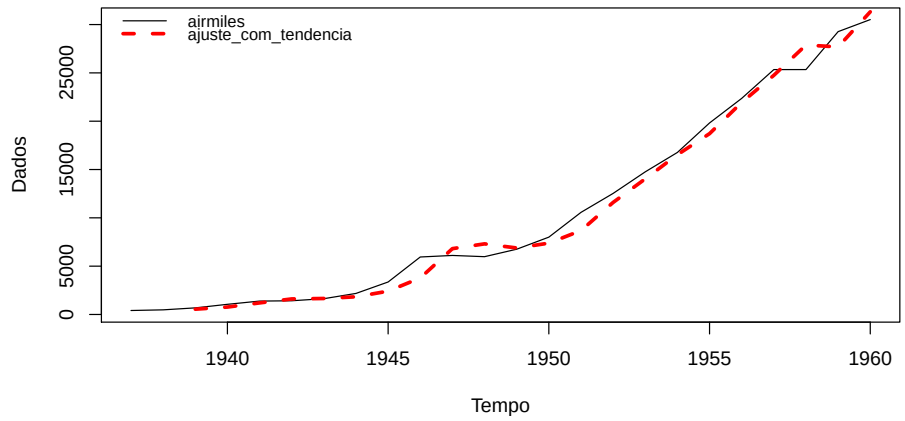


Figura 3.10: Ajuste da série utilizando o modelo SEH

Pela figura (3.10) podemos ver que a suavização acompanha a tendência de crescimento da série original.

Com a função `forecast.HoltWinters()` podemos obter as previsões do modelo SEH com os seus respectivos intervalos de confiança. Plotaremos também a série original com as previsões para os próximos 10 anos. É possível ver a tendência de crescimento nas previsões (figura 3.11).

```
> previsao_com_tendencia<-forecast.HoltWinters(ajuste_com_tendencia)
> plot(previsao_com_tendencia,xlab="Tempo",ylab="Dados",main="")
```

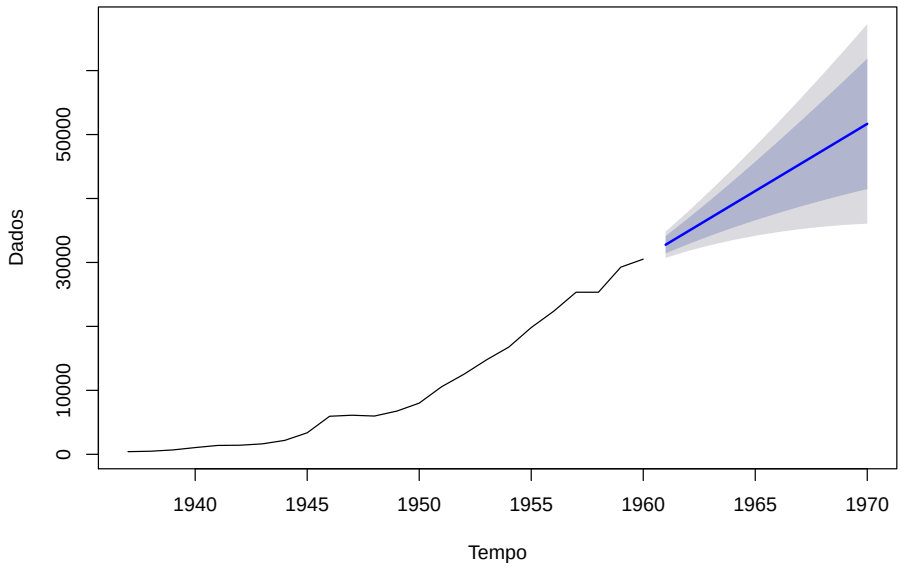


Figura 3.11: Previsão para os próximos 10 anos utilizando o modelo SEH

Suavização Exponencial Sazonal de Holt-Winters

Em séries com comportamento mais complexos, como sazonalidade, o modelo indicado é o de Holt-Winters (HW). Este modelo é uma extensão feita por Winters em 1960 ao modelo proposto por Holt em 1957 com o intuito de caputrar a sazonalidade. O modelo segue a mesma proposta da equação vista na seção anterior e cria uma equação adicional para a sazonalidade. Sua vantagem é semelhante ao método de Holt, no entanto, o modelo HW é adequado para séries de comportamento mais geral, uma vez que na prática a maioria das séries encontradas apresentam a componente de sazonalidade.

Existem duas variações deste método para estudar o comportamento sazonal de uma série temporal: a aditiva e multiplicativa. A seguir, veremos com mais detalhes a construção destes modelos.

O Modelo Aditivo

O modelo aditivo é dado por:

$$L_t = \alpha(Z_t - S_{t-m}) + (1-\alpha)(L_{t-1} + T_{t-1}) \quad 0 \leq \alpha \leq 1; t = m+1, ..., N \tag{3.19}$$

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \quad 0 \leq \beta \leq 1; t = m+1, ..., N \tag{3.20}$$

$$S_t = \gamma(Z_t - L_{t-1} - T_{t-1}) + (1-\gamma)S_{t-m} \quad 0 \leq \gamma \leq 1; t = m+1, ..., N \tag{3.21}$$

$$\hat{Z}_{t+h} = L_t + T_t h + S_{t+h-m} \tag{3.22}$$

Onde α, β e γ encontradas, respectivamente, em (3.19), (3.20) e (3.21) representam os coeficientes de suavização e equações para o nível, tendência e sazonalidade. A equação (3.22) apresenta a formulação para o modelo de previsão h passos a frente.

O Modelo Multiplicativo

Utilizando a mesma proposta do procedimento anterior, o modelo multiplicativo é dado por

$$L_t = \alpha\left(\frac{Z_t}{S_{t-m}}\right) + (1-\alpha)(L_{t-1} + T_{t-1}) \quad 0 \leq \alpha \leq 1; t = m+1, ..., N \tag{3.23}$$

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \quad 0 \leq \beta \leq 1; t = m+1, ..., N \tag{3.24}$$

$$S_t = \gamma\left(\frac{Z_t}{L_{t-1} + T_{t-1}}\right) + (1-\gamma)S_{t-m} \quad 0 \leq \gamma \leq 1; t = m+1, ..., N \tag{3.25}$$

$$\hat{Z}_{t+h} = (L_t + T_t h)S_{t+h-m} \tag{3.26}$$

As equações (3.23), (3.24) e (3.25), representam, respectivamente as estimativas do fator de nível, tendência e sazonal. Já a equação (3.26) representa a formulação para previsão h passos à

frente. Além disso, α, β e γ são constantes de suavização.

Exemplo de Aplicação

Utilizaremos a série temporal sazonal AirPassengers que registra mensalmente o total de passageiros internacionais (em milhares) da linha aérea (Pan Am) no período de janeiro de 1949 a dezembro 1960, nos EUA, que tem sazonalidade.

```
> AirPassengers
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
```

Usando a função `HoltWinters()`, descobrimos que os melhores valores para as constantes α, β e γ são, respectivamente, 0,2479; 0,0345 e 1. A interpretação destes coeficientes é análoga aos métodos de suavização visto anteriormente. Aqui, o fato de α estar mais próximo de 0 do que de 1 significa que a estimativa do nível tem pouco peso nas observações mais recentes, já para o β, que está mais próximo de 0 indica que a estimativa de tendência tem pouca influência dos valores mais recentes e γ ser igual a 1 indica que a estimativa da sazonalidade tem influência direta nas observações mais recentes.

```
> ajuste_com_sazonalidade<-HoltWinters(AirPassengers)
> ajuste_com_sazonalidade

Holt-Winters exponential smoothing with trend and additive seasonal component.

Call:
HoltWinters(x = AirPassengers)

Smoothing parameters:
 alpha: 0.2479595
 beta : 0.03453373
 gamma: 1

Coefficients:
          [,1]
a   477.827781
```

```
b      3.127627
s1   -27.457685
s2   -54.692464
s3   -20.174608
s4    12.919120
s5    18.873607
s6    75.294426
s7   152.888368
s8   134.613464
s9    33.778349
s10 -18.379060
s11 -87.772408
s12 -45.827781
```

A figura (3.12) representa o gráfico da série original em conjunto com o ajuste via HW. Pela figura (3.12) é possível perceber que a suavização acompanha bem a série original.

```
> plot(AirPassengers,xlab="Tempo",ylab="Dados")
> lines(fitted(ajuste_com_sazonalidade)[,1],col="red",lty=2,lwd =3)
> legend('topleft', legend=c("AirPassengers", "ajuste_com_sazonalidade"),
         bty = "n",col=c("black","red"), lty=c(1,2), cex=0.8,lwd =c(1,3))
```

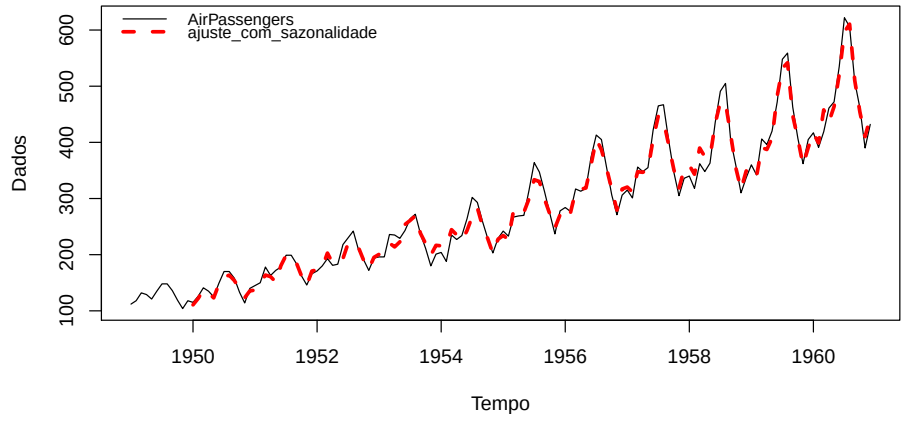


Figura 3.12: Ajuste da série utilizando o modelo HW

Com a função `forecast.HoltWinters()` podemos obter a previsões do modelo HW com os seus respectivos intervalos de confiança. Sendo assim, a figura (3.13) representa a série original com as previsões para os próximos dois anos. Podemos ver pelas previsões a tendência de crescimento somado ao efeito sazonal.

```
> previsao_com_sazonalidade<-forecast.HoltWinters(ajuste_com_sazonalidade)
> plot(previsao_com_sazonalidade,main="",xlab="Tempo",ylab="Dados")
```

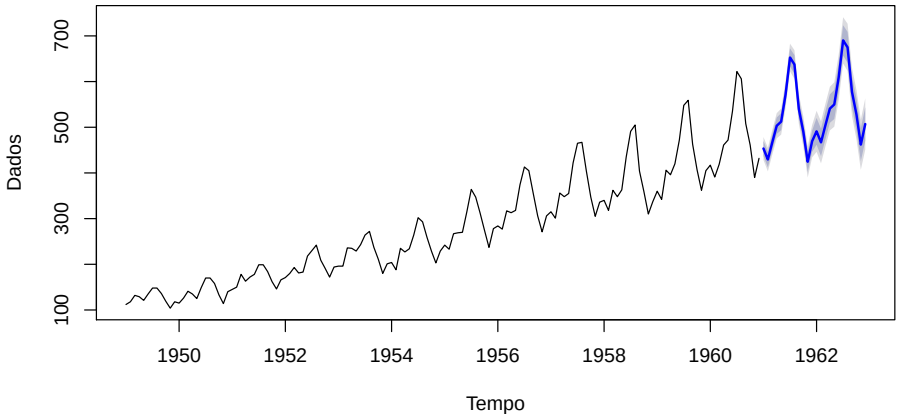


Figura 3.13: Previsão para os próximos 2 anos utilizando o modelo HW

Considerações Finais

Neste capítulo foi apresentado modelos que permitem fazer ajustes e previsões em séries temporais. Inicialmente foi visto o modelo mais simples, conhecido como NAIVE, cuja previsão é igual ao último valor observado. Em seguida o leitor tomou conhecimento de modelos baseados em médias móveis (Centrada e Simples). Por fim, foi apresentado modelos de suavização exponencial de estrutura aditiva e multiplicativa, na qual combinam as componentes de tendência e sazonalidade, além da componente denominada erro, que representa a parte não capturada da série.

Ao final deste capítulo espera-se que o leitor adquira um conhecimento inicial sobre ajustes e previsões de séries temporais utilizando, principalmente, modelos que possuem estruturas que acomodam as componentes de tendência e sazonalidade.

Processos Não-estacionários

Pedro Costa Ferreira

Lucas Farias Lima

Introdução

No contexto deste livro, um processo estocástico é estacionário se sua média e variância são constantes no tempo e sua autocovariância depende apenas da ordem de defasagem[1]. Assim, formalmente, um processo $\{y_t\}$ é estacionário se:

$$E(y_t) = E(y_{t-s}), \tag{4.1a}$$

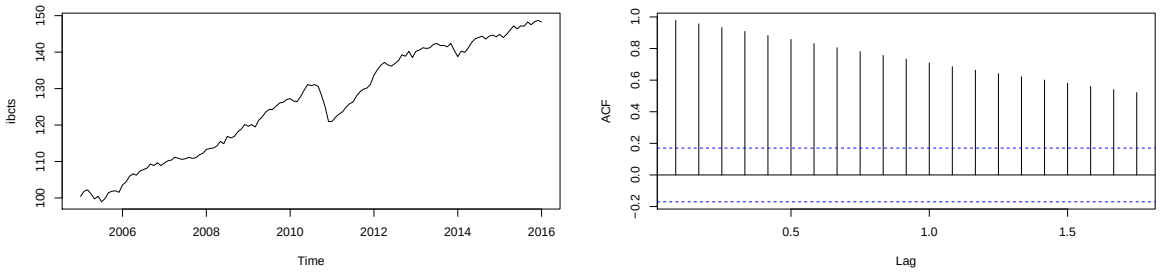
$$E[(y_t)^2] = E[(y_{t-s})^2], \tag{4.1b}$$

$$E[(y_t - \mu)(y_{t-s} - \mu)] = E[(y_{t-j} - \mu)(y_{t-j-s} - \mu)], \tag{4.1c}$$

$\forall\ s, j$, onde $\mu = E(y_t)$.

Na prática, isso significa que o processo não apresenta tendência aparentes, e que tanto a variação quanto o padrão desta variação são constantes ao longo do tempo. Isso sugere que é possível dizer, a partir da visualização de seu gráfico no tempo, se uma série aparenta se comportar de maneira estacionária.

Uma ferramenta que auxilia nessa etapa visual é o gráfico da função de autocorrelação (FAC) da série. Um processo não estacionário apresenta um lento decaimento de sua função de autocorrelação[2]. Essa característica pode ser observada, por exemplo, na série do Índice de Atividade Econômica do Banco Central do Brasil (BC), o IBC-BR, disponibilizado no próprio site do BC (Figura 4.1).



(a) Série do IBC-BR

(b) ACF da série do IBC-BR

Figura 4.1: Índice de Atividade Econômica do Banco Central do Brasil

A análise visual da FAC (Figura 4.1b) sugere que a série é não estacionária, o que nos leva ao

[1] Para maiores detalhes ver Hamilton (1994) e Enders (2008).
[2] A prova é apresentada em Enders (2008, p. 60).

problema de inferirmos a causa e buscar uma maneira de tratar a não-estacionariedade.

Tal problema torna pertinente notar que tão importante quanto definir a existência de não estacionariedade é sermos capazes de determinar sua causa. Nesse sentido, apresentaremos os testes estatísticos mais comuns disponíveis no R, que em geral implementam testes de hipóteses acerca da existência de raízes unitárias, propriedade que, grosso modo, faz com que uma série acumule choques aleatórios indefinidamente.

Não obstante, vale ressaltar que, quase em totalidade, os testes existentes assumem hipóteses e apresentam limitações que tornam a análise de estacionariedade, em alguns casos, uma tarefa menos sistemática e mais investigativa. Uma instância importante dos casos que apresentam complicações são aqueles onde a série apresenta quebras estruturais, o que impede a utilização de testes tradicionais.

Vale lembrar que toda a teoria de séries temporais está amparada sobre a de equações em diferença estocásticas, que é de fato de onde vêm as condições necessárias e suficientes que nos permitem, a partir do resultado de uma estatística, inferir se a série em questão é estacionária ou não[3].

O conteúdo deste texto está baseado principalmente nos livros de Enders (2008), Cowpertwait & Metcalfe (2009) e Zeileis et al. (2001). Ademais, todos os códigos R e dados utilizados para produção deste capítulo, estão disponíveis em:

<https://github.com/pedrocostaferreira/Analise-de-Series-Temporais-em-R>.

Este capítulo está organizado da seguinte maneira: na Seção 4.2 abordamos os motivos mais comuns pelo qual uma série é não-estacionária; na Seção 4.3 apresentamos alternativas para tratar desses problemas; já na Seção 4.4 apresentamos testes formais para identificar raizes unitárias e na Seção 4.5 abordamos o problema das quebras estruturais.

Tipos de não-estacionariedade

Ainda que estacionariedade seja uma propriedade bem definida de uma série temporal, o motivo pelo qual uma série não é estacionária é um problema mais delicado e sua inferência incorreta leva a sérios erros de modelagem e previsão.

Como visto em Enders (2008, p. 156), a presença de uma tendência (determinística ou estocástica) é o motivo mais comum pelo qual uma série temporal é não-estacionária.

[3] Ao leitor interessado sugerimos a leitura do capítulo 1 de Enders (2008).

Todavia, antes de abordar esses dois casos, é interessante apresentar o processo de ruído branco, que é formado por uma sequência de variáveis aleatórias que apresentam média e covariâncias nulas e variância constante. Ou seja, se a sequência $\{\varepsilon_t\}$ é um ruído branco, então:

$$E(\varepsilon_t) = 0, \tag{4.2a}$$

$$E[(\varepsilon_t)^2] = \sigma^2, \tag{4.2b}$$

$$E[(\varepsilon_t - E(\varepsilon_t))(\varepsilon_{t-s} - E(\varepsilon_t))] = E[(\varepsilon_t \varepsilon_{t-s})] = 0. \tag{4.2c}$$

Nesta seção, a parte estocástica dos modelos estacionários será representada por um ruído branco, mas nada impede que utilizemos um outro processo da forma $A(L)\varepsilon_t$, onde $A(L)$ é uma função polinomial do operador de defasagem L (i.e. $Ly_t = y_{t-1}$).

Chamamos tendência-estacionário um processo da seguinte forma:

$$y_t = \alpha + \beta t + \varepsilon_t, \tag{4.3}$$

onde ε_t é um ruído branco.

Um processo desse tipo é não-estacionário devido à presença da tendência determinística gerada pelo termo βt. Como veremos na seção seguinte, para tornar um processo desse tipo estacionário basta estimar o valor de β e subtrair a sequência $\hat{\beta}t$[4] da série original.

Em contrapartida, um passeio aleatório é um processo composto pela soma de choques aleatórios, que pode ser representada pela soma cumulativa de ruídos brancos:

$$y_t = \alpha + \sum_{i=1}^{t} \varepsilon_i. \tag{4.4}$$

Esse tipo de processo é o elemento básico de séries temporais que apresentam tendências estocásticas e pode se apresentar também com um termo de *drift* (i.e. uma contante somada t vezes), que se traduz em uma tendência determinística, mas que na realidade surge, como se vê em Enders (2008, p. 159), da solução de uma equação em diferenças estocásticas composta por um ruído branco e uma

[4] $\hat{\beta}$ é o valor estimado de β.

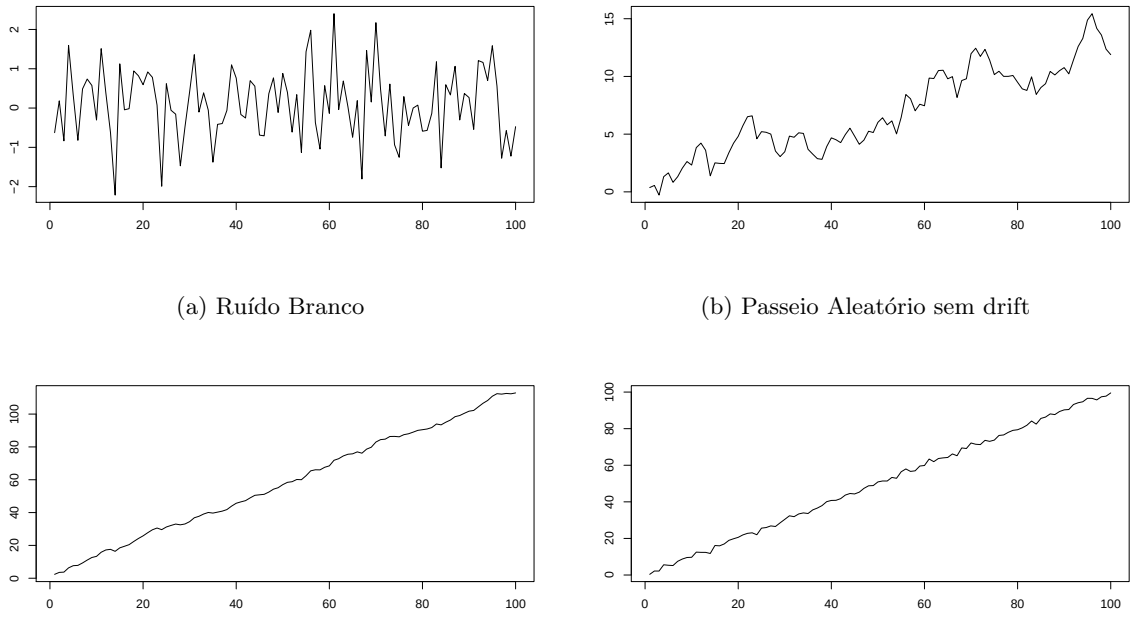
constante:

$$y_t = \alpha + \gamma t + \sum_{i=1}^{t} \varepsilon_i. \tag{4.5}$$

Um passeio aleatório pode ainda apresentar um ruído:

$$y_t = \alpha + \sum_{i=1}^{t} \varepsilon_t + \zeta_t. \tag{4.6}$$

Juntos, os processos representados pelas quatro equações acima norteiam as análises de estacionariedade, pois, de acordo com Cowpertwait & Metcalfe (2009, p. 221), o comportamento essencial de boa parte das séries temporais pode ser mimetizado por suas simples estruturas.



(a) Ruído Branco

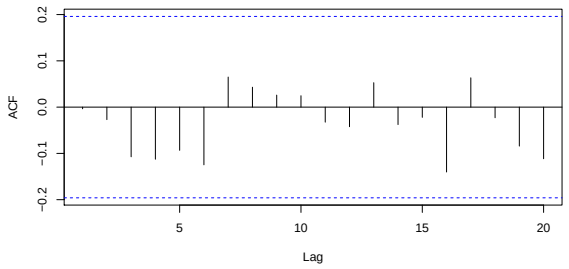
(b) Passeio Aleatório sem drift

(c) Passeio Aleatório com drift

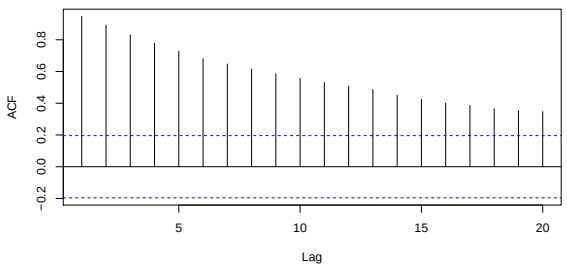
(d) Tendência-estacionário

Figura 4.2: Séries originadas das equações (4.3)-(4.6)

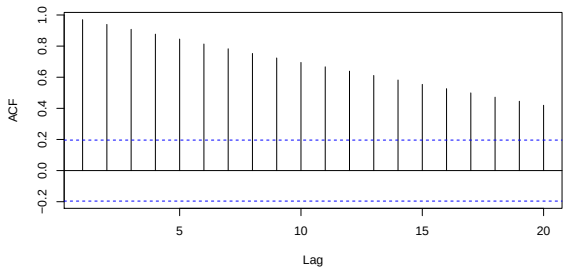
As Figuras 4.2 e 4.3 contêm algumas informações interessantes. De início, dada a estacionariedade do processo de ruído branco (Figura 4.2a), vemos (em contrapartida com a primera Figura deste capítulo) ao que se assemelha a FAC de um processo estacionário (Figura 4.3a). Obviamente estamos olhando para um resultado ideal, mas geralmente buscamos encontrar, para processos estacionários, FAC's com quedas muito rápidas, sem padrões sistemáticos ou alguma autocorrelação significativa.



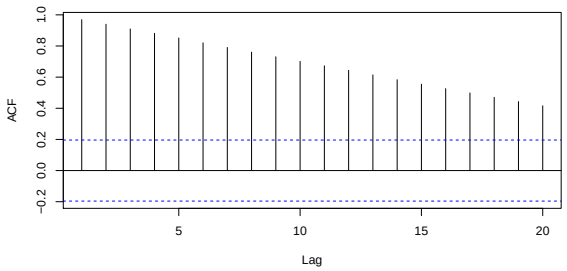
(a) ACF: Ruído Branco



(b) ACF: Passeio Aleatório sem drift



(c) ACF: Passeio Aleatório com drift



(d) ACF: Tendência-estacionário

Figura 4.3: ACF: Séries originadas das equações (4.3)-(4.6)

Em seguida, verificamos que a presença de estacionariedade estocástica ou determinística torna séries temporais com tais características indistinguíveis à luz de sua visualização gráfica e função de autocorrelação. Ainda que o processo de passeio aleatório sem *drift* (Figura 4.2b) claramente se diferencie dos outros dois processos, as FAC's dos três são bem semelhantes (Figuras 4.2b-4.2d) e, memso que estejamos trabalhando com dados simulados, tal complicação é comum em estudos empíricos. Veremos na Seção 4.3 algumas de suas implicações.

Diferenciação e Remoção de Tendência

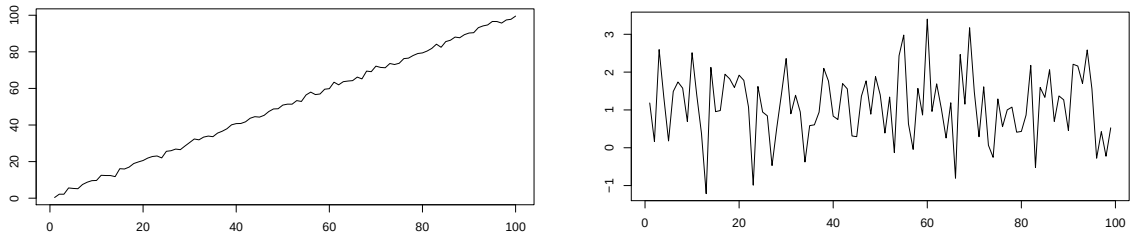
Na seção anterior, verificamos que a presença de tendências determinísticas ou estocásticas geram processos bastante semelhantes. Todavia, a maneira de tratar a não-estacionariedade em cada um dos casos é diferente, e aplicar o método incorreto causa sérios danos à informação contida na série temporal. Em particular, como se vê em Cowpertwait & Metcalfe (2009, p. 221), a diferenciação remove tanto tendências estocásticas quanto determinísticas.

Se diferenciamos um processo tendência-estacionário $y_t = \alpha + \beta t + \varepsilon_t$, obtemos $\Delta y_t = \beta + \varepsilon_t - \varepsilon_{t-1}$.

Note que Δy_t é um processo de média móvel $MA(1)$ somado à uma constante e, portanto, é estacionário.

Todavia, a abordagem sugerida na literatura é a de regredirmos os dados em uma sequência representando o tempo e tomarmos os resíduos como nossa nova série temporal: $y_t = \alpha + (\beta - \hat{\beta})t + \zeta_t$. Tal processo é igualmente estacionário, porém preserva mais da estrutura original removendo apenas a componente determinística.

Na Figura 4.4 vemos a simulação do processo tendência-estacionário antes e depois da remoção da tendência.



(a) Antes da Remoção da Tendência

(b) Depois da Remoção da Tendência

Figura 4.4: Processo Tendência-estacionário

Para uma série que apresenta não-estacionariedade devido à presença de raiz unitária, a remoção de tendência não a torna estacionária. Por exemplo, se tomarmos um passeio aleatório puro, $y_t = \alpha + \sum_{i=1}^{t} \varepsilon_i$, ou com $drift$, $y_t = \alpha + \gamma t + \sum_{i=1}^{t} \varepsilon_i$, e removermos a tendência, obtemos:

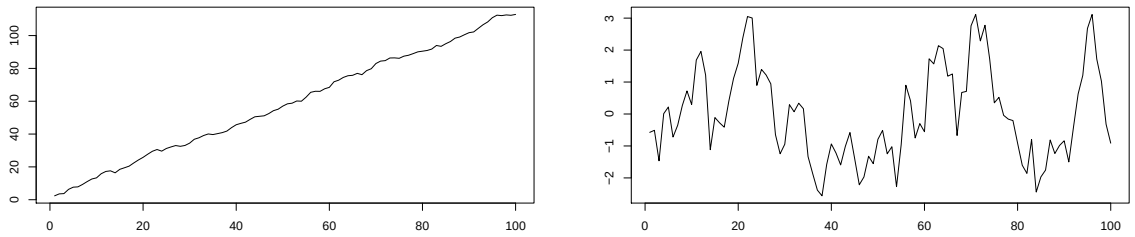
$$\Delta y_t = \sum_{i=1}^{t} \varepsilon_i - \hat{\beta} t \tag{4.7a}$$

$$,\Delta y_t = (\gamma - \hat{\beta})t + \sum_{i=1}^{t} \varepsilon_i, \tag{4.7b}$$

que não são estacionários dado que continuamos com o passeio aleatório.

Na Figura 4.5 está ilustrado um passeio aleatório com *drift* antes (Figura 4.5a) e após remoção de tendência (Figura 4.5b). Visualmente corroboramos o resultado das equações anteriores, dado que o processo resultante não parece ser tão claramente estacionário quanto no caso da presença de tendência determinística.

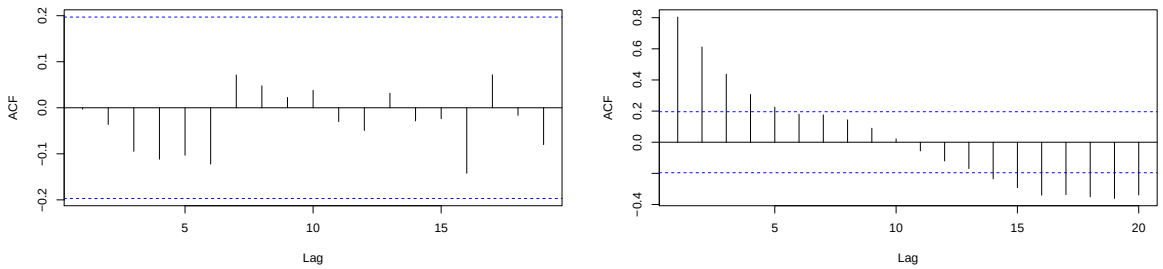
De fato, quando analisamos as FAC's de um passeio aleatório diferenciado e com remoção de tendência (Figura 4.6), vemos que após a remoção da tendência, por mais que a queda seja muito



(a) Antes da Remoção da Tendência (b) Depois da Remoção da Tendência

Figura 4.5: Passeio aleatório com *drift* antes e após remoção de tendência

mais rápida que no caso do processo original, as correlações iniciais (até a 5ª defasagem) continuam significativas, e existe um aparente comportamento sistemático. Por outro lado, a diferenciação produz uma FAC de acordo com o que esperamos de um processo estacionário.



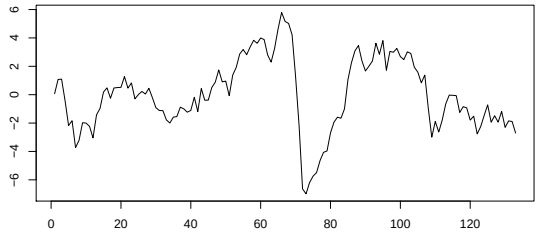
(a) Depois de Diferenciação (b) Depois da Remoção da Tendência

Figura 4.6: ACF: Passeio aleatório com *drift* diferenciado e removido de tendêncida

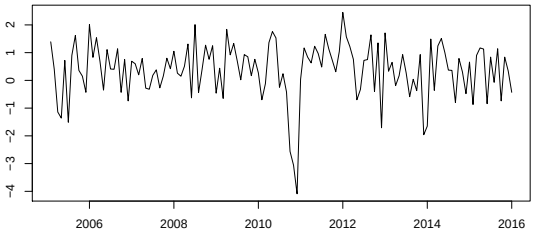
Exatamente por tais ambiguidades, faz-se necessária uma metodologia formal para a análise de estacionariedade. Na literatura, os testes de hipótese baseados na existência de raízes unitárias são a abordagem padrão.

Analisando os resultados dos mesmos procedimentos para a série do IBC-Br da Seção 4.2, temos (Figura 4.7).

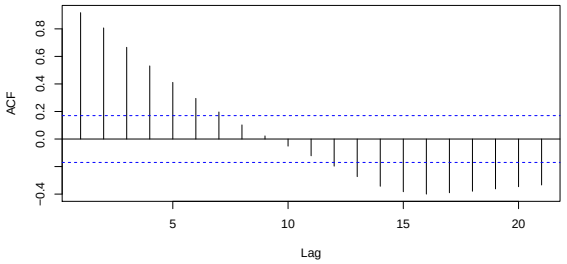
De imediato verificamos que a remoção de tendência não torna a série estacionária. Entretanto, mesmo diferenciada a série apresenta truncagens incômodas, o que se deve provavelmente à forte sazonalidade da mesma, pois ainda que esta esteja dessazonalizda, parte desse comportamento pode continuar presente. A ideia natural seria prosseguirmos a um teste de raiz unitária, mas não há como ignorar a forte queda ocorrida ao redor do ano de 2008 na série.



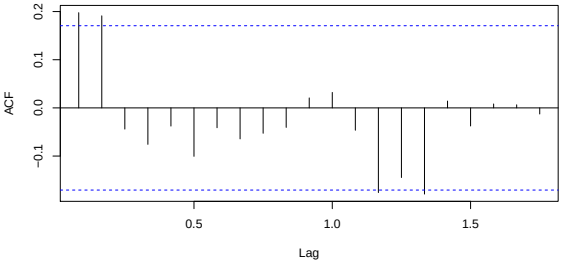
(a) Depois de Remoção de Tendência



(b) Depois de Diferenciação



(c) ACF: Depois de Remoção de Tendência



(d) ACF: Depois de Diferenciação

Figura 4.7: Série do IBC-BR

Dizemos que um processo possui raiz unitária quando os choques aleatórios que ocorrem são carregados indefinidamente. No caso do passeio aleatório, vemos que o termo $\sum_{i=1}^{t} \varepsilon_i$ representa exatamente esse acúmulo de choques aleatórios.

Dessa maneira, descontando tendências determinísticas e quebras estruturais, assumimos que a não-estacionariedade é gerada pela presença de processos que possuem raízes unitárias. Assim, para processos com tendência determinística, a remoção da tendência é suficiente para obtermos um processo estacionário, enquanto que para processos com tendência estocástica necessitamos recorrer à diferenciação.

Testes Formais

O exemplo anterior (Figura 4.7) ilustrou como a remoção da tendência aplicada a um processo que possui raiz unitária não é o suficiente para obtermos uma série estacionária. Todavia, a diferenciação da série é uma transformação forte do processo que se feito sem necessidade, induz a análises e modelagens equivocadas.

A maior motivação na detecção formal de raízes unitárias é a de utilizar os métodos adequados de modelagem, consequentemente determinando também a ordem de integração do processo (informação muito importante, por exemplo, para análises de cointegração). Em particular, o trabalho de Granger & Newbold (1974) mostrou como o uso de séries não estacionárias pode resultar em regressões lineares aparentemente muito bem ajustadas, mas que, para mencionar um dos problemas, possuem resíduos altamente correlacionados, o que viola as hipóteses de testes clássicos (e.g. testes t, F ou R^2).

Como mencionamos, a maior parte dos testes de raiz unitária são testes de hipótese, de maneira que, em muitos dos casos, compararemos estatísticas que obtemos com valores críticos. Tais valores, em sua grande maioria, são obtidos pelo método de Monte Carlo que, grosso modo, se resume na geração de cenários aleatórios em grande número para através destes gerar intervalos de confiança ou estimativas pontuais[5].

Augmented Dickey-Fuller (ADF)

O Dickey-Fuller aumentado é provavelmente o teste de hipótese para raízes unitárias mais utilizado. Grosso modo, se tomarmos um processo da forma $y_t = \alpha + \beta y_{t-1} + \varepsilon_t$, o objetivo do teste é, então, inferir se $\beta = 1$.

O problema é que estimar o modelo linear acima sob a hipótese de não-estacionariedade nos faz incorrer no problema de regressão espúria e, portanto, ficamos impedidos de inferir o valor de β utilizando as estatísticas de testes usuais.

Dessa maneira **?** propõe realizar o teste sob a série diferenciada em três configurações que possibilitam obter as estatísticas de testes adequadas, dado que em suas simulações notaram que a presença de componentes determinísticas alteram os valores críticos obtidos.

Além disso, ainda que a série seja diferenciada, sob a hipótese nula de existência de raiz unitária, é possível que os resíduos das regressões estejam correlacionados. Assim, adiciona-se o termo de média móvel $\sum_{i=1}^{p} \beta_i \Delta y_{t-i}$, o que gera o problema de determinar a quantidade de defasagens a incluir na regressão, o que geralmente se decide, como se vê em Enders (2008, p. 193), a partir de critérios de informação do tipo AIC e BIC/SBC[6].

Portanto, as equações do teste (geralmente referenciadas como ADF versão 1, 2 e 3, respectivamente), são:

[5] Para mais detalhes ver Enders (2008, p. 175).
[6] *Akaike information criterion* e *Bayesian information criterion/Schwarz criterion*, respectivamente.

$$\Delta y_t = \gamma y_{t-1} + \sum_{i=1}^{p} \beta_i \Delta y_{t-i} + \varepsilon_t, \tag{4.8a}$$

$$\Delta y_t = \alpha + \gamma y_{t-1} + \sum_{i=1}^{p} \beta_i \Delta y_{t-i} + \varepsilon_t, \tag{4.8b}$$

$$\Delta y_t = \alpha + \gamma y_{t-1} + \beta t + \sum_{i=1}^{p} \beta_i \Delta y_{t-i} + \varepsilon_t. \tag{4.8c}$$

Assim, escolhida a estrutura e a quantidade p de defasagens adequadas à série de interesse, realiza-se o teste e confrontam-se as estatísticas de testes obtidas com os valores críticos tabelados. Em particular, dado que o teste inclui a presença de componentes determinísticos, os autores fornecem também as estatísticas de testes conjuntas para avaliarmos se, sob a hipótese de raiz unitária, a estrutura que acreditamos que a série segue é adequada, o que é útil para o procedimento descrito no próximo parágrafo.

Uma utilização direta da flexibilidade do teste ADF é a de permitir um estudo quase que exaustivo de um processo quando não conhecemos nada sob sua estrutura teórica. Tal metodologia é apresentada no trabalho elaborado em Dolado et al. (1990).

O procedimento sugere iniciar os testes a partir do modelo irrestrito, contendo todas as componentes determinísticas, parando caso se conclua a não existência de raiz unitária e, a cada etapa em que não se rejeite a hipótese nula, avalia-se a significância do parâmetro do componente determinístico sob a hipótese de raiz unitária, removendo-o se não for significante, passando-se então ao teste mais restrito.

Finalmente, vale lembrar que é de igual importância verificar as estatísticas F geradas pelo teste ADF. A correta especificação do processo a ser testado é crucial para a inferência com base no teste de hipóteses.

Assim como todos os testes apresentados a seguir, o ADF que utilizaremos é parte do pacote **urca** e sua sintaxe é:

```
ur.df(y, type = c("none", "drift", "trend"), lags = 1, selectlags = c("Fixed", "AIC", "BIC"))
```

- `y`: série temporal a ser testada;
- `type`: componente determinística a ser incorporada da regressão do teste;
- `lags`: quantidade máxima de defasagens a se incluir na regresssão;

- `selectlags`: critério de informação para escolha da defasagem ótima, baseada na entrada anterior.

KPSS

Um problema do teste ADF é seu pequeno poder estatístico, que o torna praticamente incapaz de diferenciar uma série com raiz unitária de uma com raiz "quase" unitária, ou seja, cria um viés para a conclusão de existência de raiz unitária. Isso é um problema sério dado que a correção para uma série não estacionária é sua diferenciação, o que implica que se o teste falha, nos leva a diferenciar uma série estacionária, que elimina muita informação a respeito do processo.

Dentre os testes alternativos, o mais conhecido é KPSS apresentado em Kwiatkowski et al. (1992), que inverte a hipótese nula do teste ADF (o que, rigorosamente, o torna um teste de estacionariedade):

$$y_t = d_t + r_t + \varepsilon_t, \tag{4.9}$$

onde:

d_t é uma tendência determinística

r_t é um passeio aleatório

ε_t é um processo de erro estacionário

Além disso, a hipótese nula é que o termo r_t é nulo. Assim como o ADF, o teste necessita da definição da existência de componentes determinísticos, assim como a definição da quantidade de defasagens:

```
ur.kpss(y, type = c("mu", "tau"), lags = c("short", "long", "nil"), use.lag = NULL)
```

- `y`: série temporal a ser testada;
- `type`: se define "mu" para o teste com termo constante e "tau" para o com tendência;
- `lags`: "short" adiciona $\sqrt[4]{4 \times (n/100)}$ defasagens, "long" adiciona $\sqrt[4]{12 \times (n/100)}$ e "nil" nenhuma defasagem (sendo n o tamanho da amostra);
- `use.lag`: define um número arbitrário de defasagens.

Phillips-Perron

O teste apresentado em Phillips & Perron (1988), utiliza a mesma estrutura do teste DF (ADF sem o termo de média móvel), todavia trata do problema de correlação serial corrigindo a estatística de teste. Tais correções, conhecidas na literatura como HAC[7] que, em linhas gerais, normalizam a matriz de variância e covariância com base nos *clusters* (i.e. grupos homogêneos) de concentração dessas medidas, espurgando seus efeitos.

Porém, vale ressaltar que tal correção tem suporte nas hipóteses para séries de grande tamanho, sendo inadequado para pequenas amostras.

```
r.pp(x, type = c("Z-alpha", "Z-tau"), model = c("constant", "trend"), lags = c("short", "long"),
use.lag = NULL)
```

- `x`: série temporal a ser testada;
- `type`: estatísticas de teste a serem computadas, "Z-alpha" sendo para o modelo com constante e "Z-tau" para o com tendência;
- `model`: modelo a ser testado, assim como no teste ADF;
- `lags`: quantidade de defasagens de acordo com o tamanho da amostra, como no teste KPSS;
- `use.lag`: define um número arbitrário de defasagens.

Dickey Fuller-GLS (ERS)

Como observamos na seção sobre o teste ADF, seus dois problemas principais são o baixo poder estatístico e sua sensibilidade à presença de termos determinísticos. Um problema de igual tamanho é a maneira adequada de tratar a presença destes termos nos testes.

O teste DF-GLS (**?**), assim como o ADF, solicita a especificação da quantidade máxima de lags. Possui também duas formas, que adequam o teste para a presença de uma tendência ou drift determinístico. A diferença é que a série passa por uma transformação via MQG (Mínimos Quadrados Generalizados) que, demonstram os autores, aumenta significativamente o poder estatístico do teste.

Os termos de drift e tendência são estimados pelo método MQG e depois removidos da série. Dado que o método MQG ajusta a matriz de variância e covariância dos resíduos para a presença de correlação e heterocedasticidade, a transformação proposta pelos autores faz uma pequena alteração

[7] Do inglês *heteroskedasticity and autocorrelation consistent*, se referindo a matriz de variância e covariância.

na série original, de maneira que a remoção dos componentes determinísticos ocorre de forma local. Assim, após a tranformação, um teste padrão do tipo ADF é realizado na série resultante.

De acordo com os autores, esse procedimento aumenta siginificativamente o poder estatístico do teste ADF.

```
ur.ers(y, type = c("DF-GLS", "P-test"), model = c("constant", "trend"), lag.max = 4)
```

- `y`: série temporal a ser testada;
- `type`: definido "DF-GLS"para realização do teste DF-GLS como descrito (i.e. série removida de tendência e sem intercepto), enquanto que "P-test"corrige as estatísticas para presença de correlação serial na regressão do teste;
- `model`: especifica a componente determinística;
- `lag.max`: define um número máximo de defasagens.

Zivot-Andrews

O teste de Zivot-Andrews, proposto em Zivot & Andrews (1999), busca testar a hipótese nula de raiz unitária na presença de uma quebra estrutural nos parâmetros de nível, inclinação ou ambos. De acordo com os autores, a principal diferença em relação a outros testes é a endogeneização da quebra sob a hipótese nula, o que permite a correta inferência da mudança de parâmetro sob a presença de raiz unitária, o exato motivo pelo qual os outros testes são inadequados.

O teste se baseia no menor valor da estatística t do teste ADF, de maneira que uma quebra existiria onde há menos evidência da hipótese nula de raiz unitária. Os argumentos da função tem o mesmo significado dos testes anteriores.

```
ur.za(y, model = c("intercept", "trend", "both"), lag=NULL)
```

- `y`: série temporal a ser testada;
- `model`: especifica a componente determinística;
- `lag`: define um número de defasagens.

Como veremos na próxima seção, há motivos para acreditarmos que a série do IBC-Br que apresentamos no início do capítulo possua uma quebra estrutural e, sob essa hipótese, dentre os testes de raiz unitária que apresentamos, o Zivot-Andrews é o único adequado.

Assim, analisemos os resultados do teste:

```
> ibcts_za<- ur.za(ibcts)
> summary(ibcts_za)
     ################################
     # Zivot-Andrews Unit Root Test #
     ################################


     Call:
     lm(formula = testmat)

     Residuals:
         Min      1Q  Median      3Q     Max
     -4.2817 -0.5174  0.0977  0.6311  2.2671

     Coefficients:
                Estimate Std. Error t value Pr(>|t|)
     (Intercept) 11.02833    3.62846   3.039  0.00287 **
     y.l1         0.89103    0.03681  24.208  < 2e-16 ***
     trend        0.05088    0.01644   3.094  0.00242 **
     du          -0.87398    0.37080  -2.357  0.01994 *
     ---
     Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

     Residual standard error: 0.9766 on 128 degrees of freedom
       (1 observation deleted due to missingness)
     Multiple R-squared:  0.9958,        Adjusted R-squared:  0.9957
     F-statistic: 1.009e+04 on 3 and 128 DF,  p-value: < 2.2e-16


     Teststatistic: -2.9606
     Critical values: 0.01= -5.34 0.05= -4.8 0.1= -4.58

     Potential break point at position: 69
```

O baixo p-valor sugere rejeição da hipótese nula, de que o processo é um passeio aleatório com comportamento sendo a suposta quebra um ruído de natureza endógena. Ou seja, há evidência de que o processo é estacionário com uma quebra de natureza endógena.

Quebras Estruturais

O problema de quebras estruturais é importante por si própio. Muitas vezes a determinação de uma quebra corrobora a hipótese de que um determinado fato ou acontecimento tenha mudado a estrutura de alguma variável econômica, por exemplo.

No âmbito da análise de estacionariedade, a presença de uma quebra estrutural viola hipóteses na maioria dos testes de raiz unitária. Todavia, antes de pensar em corrigir tal problema, enfrentamos

outro tão difícil quanto: definir se de fato há uma quebra estrutural.

Neste processo, o conhecimento do pesquisador acerca da série em questão e dos contextos relevantes à ela são muito importantes, pois mais interessante do que inferir se a hipótese de quebra se justifica estatisticamente é poder suportá-la sob uma justificativa que trate do processo gerador.

Principais Testes para Quebras Estruturais

Na literatura, um dos primeiros esforços na detecção de quebras estruturais se encontra em Chow (1960), que em linhas gerais propôs a comparação dos resíduos de um modelo onde se calcula duas regressões, separadas pelo momento em que acredita ter ocorrido a quebra (equivalente a um modelo irrestrito) com os resíduos de um modelo de apenas uma regressão para todo o período (modelo restrito). A estatística de teste é, portanto, da forma de uma F:

$$F = \frac{\hat{u}^T\hat{u} - \hat{u}_R^T\hat{u}_R}{\hat{u}_R^T\hat{u}_R/(n-2k)}. \tag{4.10}$$

Assim, na equação (4.10), $\hat{u}$ são os resíduos do modelo irrestrito, $\hat{u}_R$ os do modelo restrito, n é o tamanho da amostra e k o número de parâmetros estimados. A limitação do teste é a necessidade de se conhecer o momento da quebra. Todavia, existem testes baseados nesta mesma estatística que contornam essa limitação. Uma das possibilidades é realizar o teste para vários períodos dentro de uma janela, como sugere e implementa Zeileis et al. (2001) no pacote **strucchange** (assim como o restante dos testes da seção):

```
Fstats(formula, from = 0.15, to = NULL, data = list())
```

- `formula`: estrutura do processo seguido pela série temporal (e.g. ARIMA(1,1,0));
- `from`: intervalo de cálculo da estatística (data inicial). O valor padrão é de 15% da amostra;
- `to`: o intervalo de cálculo da estatística (data final);
- `data`: base de dados a ser utilizada, caso não se queira carregá-la fora da função.

Outra ferramenta são os testes de flutuação empírica (em inglês *empirical fluctuation process* (efp)) que se baseiam no método apresentado originalmente em Brown et al. (1975). A proposta é inferir sobre a estabilidade dos parâmetros a partir do comportamento da soma cumulativa (equação

(4.11)) dos resíduos recursivos normalizados[8] de um modelo que descreva o processo adequadamente.

$$efp(s) = \frac{1}{\hat{\sigma}\sqrt{(n)}} \sum_{t=1}^{|ns|} \hat{\varepsilon}_t. \tag{4.11}$$

De acordo com Zeileis et al. (2001), sob a hipótese nula de estabilidade do processo de soma cumulativa (i.e. ausência de quebra estrutural), o teorema do limite central implica que sua média não deve divergir de zero. Assim, com base no processo de flutuação escolhido, são estabelecidos limites superiores e inferiores para a oscilação do processo de maneira que há evidências de quebra estrutural caso a flutuação empírica extrapole tais limites.

efp(formula, data, type = "Rec-CUSUM", h = 0.15, dynamic = FALSE, rescale = TRUE)

- `formula`: estrutura do processo seguido pela série (e.g. ARMA(1,1,0));
- `data`: um objeto do tipo `data.frame` contendo a série;
- `type`: tipo de processo de flutuação a ser utilizado dentre "Rec-CUSUM", "OLS-CUSUM", "Rec-MOSUM"ou "OLS-MOSUM";
- `h`: janela para as somas sucessivas;
- `dynamic`: permite a inclusão de defasagens na regressão;
- `rescale`: permite a normalização dos resíduos de acordo com a subamostra da regressão (*default* `TRUE`) ou com toda a amostra (`FALSE`).

Voltando à série do IBC-Br, dado o contexto em que a economia mundial foi inserida a partir de 2008, temos motivos pra acreditar que a queda ocorrida naquele ano tenha sido uma quebra estrutural, o que significaria dizer que, após a crise financeira de 2008, o processo gerador do PIB (dado que o IBC-Br é utilizado como sua *proxy*) e, consequentemente, a estrutura da economia brasileira, foram afetados. Todavia, visualmente nada é conclusivo, dado que apesar da queda o processo parece voltar ao seu trajeto original a partir de 2011, de maneira que possa haver ocorrido apenas uma momentânea mudança do parâmetro de média.

Não obstante, na Figura 4.8, realizamos o teste de flutuação empírica para a série diferenciada[9], tomando como parâmetro um modelo AR(1). A Figura mostra que a função não extrapola os limites

[8] Resíduos da estimativa da observação k feita com base na amostra até $k-1$ divididos por $\hat{\sigma}\sqrt{n}$, onde n é o tamanho da amostra.

[9] Como o teste é realizado nos resíduos de uma regressão linear, o ideal é garantir que os dados utilizados são estacionários.

estabelecidos para o processo definido como referência[10]. Assim, dado que a hipótese nula é a de estabilidade do processo de flutuação empírico e o p-valor é alto[11], conclui-se que a série não apresenta quebra estrutural com base nesse teste.

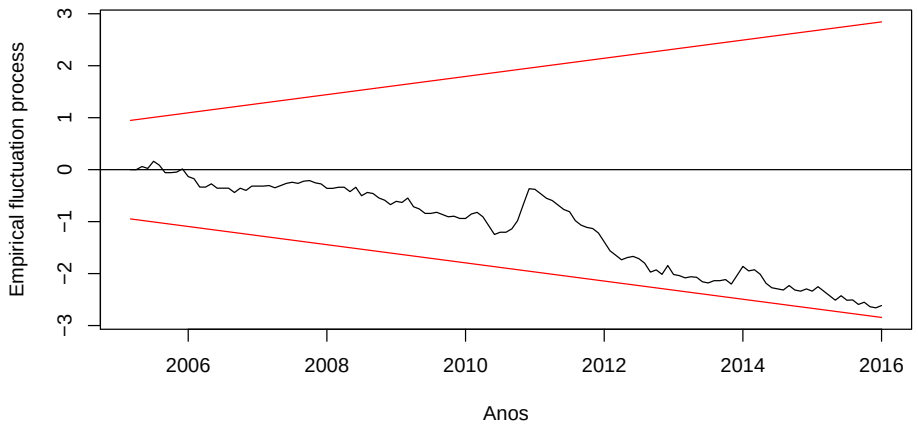


Figura 4.8: Teste de Flutuação Recursivo para a Série do IBC-Br

```
Recursive CUSUM test

data:  ibc_cus
S = 0.69829, p-value = 0.2514
```

Outra abordagem baseada em Bai & Perron (2003), busca datar as quebras estruturais existentes em um processo através de um algoritmo de programação dinâmica que minimiza a soma dos resíduos quadráticos.

A função e seus argumentos são definidos a seguir:

```
breakpoints(formula, h = 0.15, breaks = NULL, data = list(), ...)
```

- `formula`: estrutura do processo seguido pela série (e.g. ARIMA(1,1,0));
- `h`: janela de intervalos de busca (geralmente estipulada entre 10% e 15% do tamanho da amostra);
- `breaks`: número de quebras a ser testado;
- `data`: um objeto do tipo `data.frame` contendo a série temporal.

Para deixar a explicação mais clara, simulemos um passeio aleatório com drift ao qual dobramos

[10] Soma cumulativa dos resíduos recursivos (Rec-CUSUM).
[11] É possível extrair o p-valor do teste a partir da função `sctest()`, utilizando como argumento o próprio teste.

a média temporal a partir da metade da série (Figura 4.9).

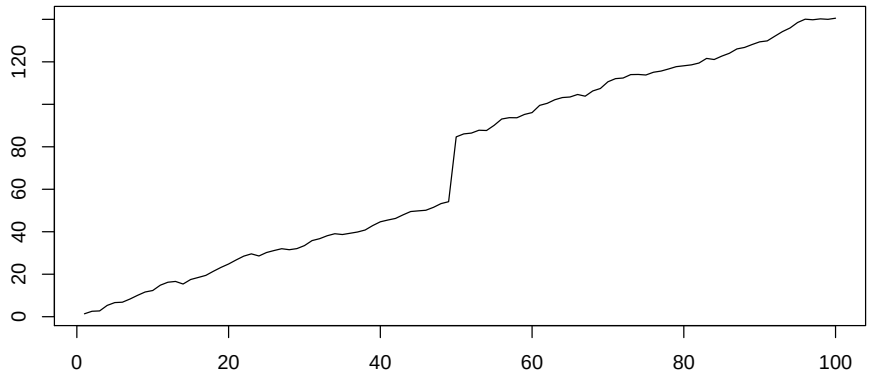
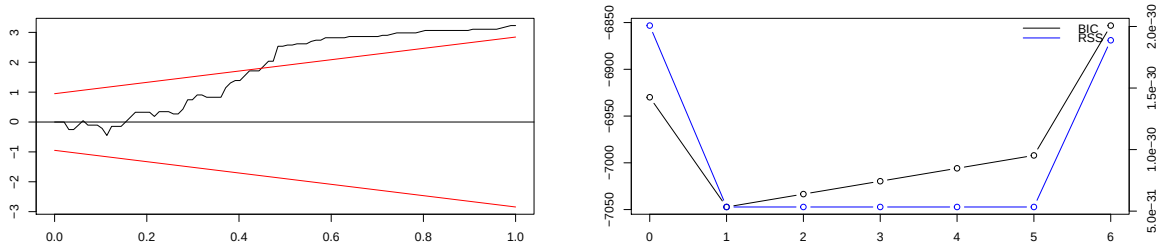


Figura 4.9: Simulação de um Processo com Quebra Estrutural

Agora, plotemos sua função de flutuação CUSUM e o gráfico resultante da função **breakpoints**, ambos gerados tomando como argumento um processo $AR(1)$ de raiz unitária em primera diferença (Figura 4.10).



(a) Teste de Flutuação

(b) Critérios de Informação

Figura 4.10: Teste de Flutuação e Critérios de Informação para Número de Quebras

Como esperávamos, exatamente no ponto onde a média do processo dobra há o rompimento da banda superior para o processo de flutuação escolhido. Todavia, observamos que o processo segue fora do limite superior, provavelmente devido à sua natureza não-estacionária, que tende a fazer com que seus momentos estatísticos cresçam ao longo do tempo, alterando consistentemente os parâmetros do processo.

O gráfico da função `breakpoints()` (Figura 4.10) mostra no eixo das coordenadas a soma dos

resíduos quadrados (RSS, do inglês *residual sum of squares*) e o critério de informação bayesiano (BIC) contra o número de quebras no eixo das ordenadas. Em teoria, ambos os valores apresentam seu mínimo para o número de quebras ótimo contido na série. Os dois valores não vão sempre coincidir, mas, como mostraram Bai & Perron (2003), o critério BIC não é tão confiável para modelos autoregressivos, sendo nesses casos aconselhável se basear na soma dos resíduos quadrados em conjunto com as evidências de quebras de outros testes, uma vez que esse sempre é realizado sob a hipótese de existência de quebras.

Para a série simulada, o critério BIC tem seu mínimo para uma quebra enquanto que para a RSS este está entre 1 e 5[12]. Todavia, dado que a série é não-estacionária e sabemos que inserimos uma quebra, as outras 4 se devem às oscilações abruptas, comuns a passeios aleatórios.

Outras implementações e variações dos testes acima podem ser encontrados no mesmo pacote. Todavia, como alerta Kleiber & Zeileis (2008, p. 173), tal variedade de testes pode se tornar um problema ao invés de uma solução e o conhecimento a priori da natureza do processo e sua história são de grande auxílio na seleção da metodologia adequada.

Decomposição de Hodrick-Prescott (Filtro HP)

Uma ferramenta muito utilizada na inferência de componentes estruturais baseada na decomposição de uma série temporal é o filtro de Hodrick-Prescott, apresentado no artigo Hodrick & Prescott (1997). A proposta da decomposição é a remoção das flutuações cíclicas de uma série temporal, tal suavização resultaria em uma série temporal que representa as flutuações de longo prazo mais evidentemente que as de curto.

Como visto em Enders (2008), nesta metolodogia se assume que a série temporal observada é composta por uma tendência $\{\mu_t\}$ e um elemento estacionário $y_t - \mu_t$. A partir daí, aplicamos a minimização de erros quadráticos para obter os parâmetros do seguinte modelo:

$$\frac{1}{T}\sum_{i=1}^{T}(y_t - \mu_t)^2 + \frac{\lambda}{T}\sum_{i=2}^{T-1}[(\mu_{t+1} - \mu_t) - (\mu_t - \mu_{t-1})]^2. \tag{4.12}$$

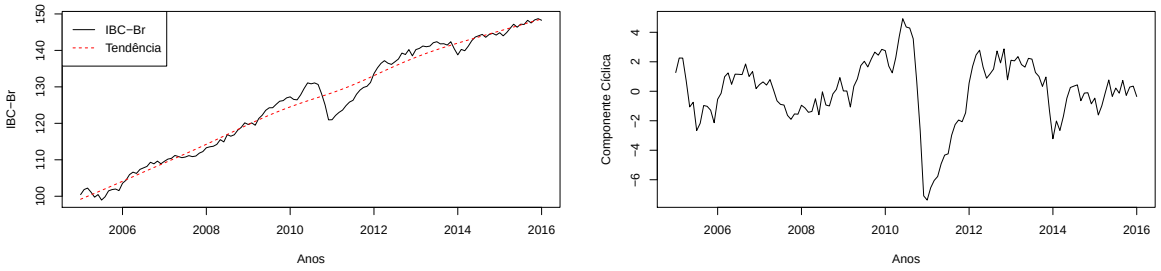
Exemplo

Para fins didáticos, apliquemos a metodologia novamente à série temporal do IBC-Br (Figura 4.1), que é tido como uma proxy para o PIB do país.

[12]Para melhor visualização, omitimos os valores do gráfico para 6 quebras, que aumentam acima dos valores de 0 quebras.

Como a proposta do filtro HP é expurgar oscilações de curto prazo, deixando restar apenas a tendência de longo prazo da série original, pode-se argumentar que aplicado à série do IBC-Br o filtro sugere qual comportamento segue do produto potencial da economia. De fato, olhando a Figura 4.11, observamos que a tendência de longo prazo da série, sugere que, tudo mais constante, o PIB do país tende a crescer.

Além disso, observamos as oscilações ocorridas na série no período apresentado, com maior destaque para a grande queda observada em 2008, período em se sentiram os choques da crise financeira internacional.



(a) IBC-Br e Componente de Tendência (b) Componente Cíclica

Figura 4.11: Decomposição HP do IBC-Br

Todavia, como alerta Enders (2008), é necessário precaução na aplicação do filtro, dado que, por suavizar a tendência da série, o filtro pode apresentar flutuações na parte irregular do processo que de fato não existem. Além disso, French (2001) afirma que o filtro possui resultados mais confiáveis quando aplicado a processos com ordem de integração 2 e os ruídos no processo têm distribuição aproximadamente normal.

Não obstante, realizemos os testes de raiz unitária descritos na seção 4, exibindo apenas as estatísticas de testes (`teststat`) e valores críticos (`cval`), dado que as componentes determinísticas inseridas foram todas significantes:

```
#### Teste ADF ----
> summary(adf_ibc)@teststat
               tau2     phi1
statistic -0.643981 4.862601
> summary(adf_ibc)@cval
      1pct  5pct 10pct
tau2 -3.46 -2.88 -2.57
phi1  6.52  4.63  3.81
#### Teste KPSS ----
> summary(kpss_ibc)@teststat
```

```
[1] 0.1299836
> summary(kpss_ibc)@cval
                10pct  5pct 2.5pct  1pct
critical values 0.119 0.146  0.176 0.216
#### Teste de Phillips-Perron ----
> summary(pp_ibc)@teststat
[1] -0.7728056
> summary(pp_ibc)@cval
                      1pct      5pct     10pct
critical values -3.480626 -2.883322 -2.578251
#### Teste DF-GLS ----
> summary(gls_ibc)@teststat
[1] -3.03545
> summary(gls_ibc)@cval
                 1pct  5pct 10pct
critical values -3.46 -2.93 -2.64
```

Resumindo, os testes ADF e KPSS nos sugerem que a série é não-estacionária, enquanto que o teste de Phillips-Perron e DF-GLS sugerem estacionariedade, como ocorreu com o teste de Zivot-Andrews anteriormente.

Considerações Finais

Neste capítulo, fomos apresentados às definições e propriedades de processos não-estacionários, o que nos ajuda a identificar e categorizá-los. Para séries não-estacionárias devido à presença de uma tendência determinística, vimos que é suficiente estimar o coeficiente dessa componente e removê-la da série original. Porém, caso desconfiamos que o motivo da não estacionariedade é uma tendência estocástica ou presença de quebras estruturais, vimos que o mais adequado é, antes de tudo, o emprego de testes formais para identificação desses problemas, uma vez que as possíveis soluções envolvem transformações profundas na série, como a diferenciação. Finalmente, vimos algumas ferramentas para inferir acerca da existência de quebras estruturais. Além disso, ao longo do texto fizemos um breve estudo da série do Indicador de Atividade Econômica do Banco Central do Brasil (IBC-Br), determinando sua clara não-estacionaridade, avaliando a presença de uma quebra estrutural e estimando seu valor potencial.

Modelos SARIMA

Pedro Costa Ferreira

Daiane Marcolino de Mattos

Introdução

Este capítulo é dedicado à apresentação do modelo SARIMA. Para tal, fez-se uso da série temporal (ST) de vendas de passagens aéreas, mais conhecida como *Air Passengers*. Trata-se de uma série temporal mensal que registra o total de passageiros internacionais (em milhares) da linha aérea Pan Am no período de janeiro de 1949 a dezembro 1960, nos EUA (Box & Jenkins, 1970).

A ST de vendas de passagens aéreas é um exemplo clássico de representação da modelagem de Box & Jenkins e a estrutura que "melhor" representa essa série temporal (ST) é um modelo SARIMA$(0,1,1)(0,1,1)_{12}$. Dado a fama obtida por esse modelo, é equivalente dizer que uma ST segue um SARIMA$(0,1,1)(0,1,1)_{12}$ ou um modelo *Airline*.

A abordagem de Box & Jenkins (1970) permite que valores futuros de uma série sejam previstos tomando por base apenas seus valores presentes e passados. Tais modelos são chamados de modelos autorregressivos integrados de médias móveis ou, simplesmente, ARIMA. Ao considerar relações sazonais, o modelo é nomeado SARIMA. Um modelo SARIMA$(p,d,q)(P,D,Q)_s$ é representado da seguinte forma:

$$\phi(L)\Phi(L)\Delta^d\Delta^D y_t = \theta(L)\Theta(L)\varepsilon_t \tag{5.1}$$

onde:

p é a ordem do polinômio autoregressivo não sazonal $\phi(L)$;

P é a ordem do polinômio autoregressivo sazonal $\Phi(L)$;

q é a ordem do polinômio de médias móveis não sazonal $\theta(L)$;

Q é a ordem do polinômio de médias móveis sazonal $\Theta(L)$;

d é a ordem de diferença não sazonal;

D é a ordem de diferença sazonal;

$\phi(L) = (1 - \phi_1 L - \phi_2 L^2 - ... - \phi_p L^p)$;

$\Phi(L) = (1 - \Phi_1 L^s - \Phi_2 L^{2s} - ... - \Phi_p L^{Ps})$;

$\theta(L) = (1 - \theta_1 L - \theta_2 L^2 - ... - \theta_p L^q)$;

$\Theta(L) = (1 - \Theta_1 L - \Theta_2 L^{2s} - ... - \Theta_p L^{Qs})$;

$\Delta = 1 - L$;

L é o operador de defasagem tal que $L^n y_t = y_{t-n}$.

Ao longo desse capítulo discutiremos as características dessa ST e os passos para modelá-la utilizando o software R, discutindo quais são os possíveis pacotes disponibilizados pelo programa. Conforme observaremos, essa é uma série temporal não estacionária nas partes sazonal e não sazonal e na variância. Aprenderemos a identificar essas características e qual é a maneira adequada de corrigí-las para fazermos uso da metodologia proposta por Box & Jenkins.

Ao ler esse capítulo, pretende-se que o leitor esteja apto a modelar uma ST "não complexa", seguindo a proposta de Box & Jenkins, utilizando o software R. Para atingir tal objetivo, além dessa introdução, esse capítulo está organizado da seguinte forma: na seção 2, entitulada Preliminares, vamos definir o nosso diretório de trabalho e comentar sobre os pacotes que precisamos instalar para estimar e analisar de maneira correta o modelo SARIMA; na seção 3, vamos explorar a ST de vendas de passagens aéreas observando sua tendência, variância e padrão sazonal; na seção 4, vamos aprofundar o nosso conhecimento sobre a ST que estamos trabalhando e discutir quais são os procedimentos que devemos adotar para modelá-la; na seção 5, baseando-se no ciclo iterativo proposto por Box & Jenkins (1970), iremos modelar a nossa ST e prevê-la 12 passos à frente; na seção 6, mostraremos como exportar as previsões para arquivos *.xlsx* e *.csv*; e, por fim, na seção 7 faremos as considerações finais.

Preliminares

Nessa seção discutiremos a definição do diretório de trabalho e comentaremos sobre os pacotes de séries temporais que iremos utilizar ao longo da modelagem.

Definição do diretório

Primeiramente, é preciso definir no R o diretório de trabalho. Isso é feito com a função `setwd()` como se segue:

```
> setwd("digitar o endereço neste espaço")
```

Instalação dos pacotes necessários

O próximo passo é instalar alguns pacotes do R utilizando a função `install.packages()`. Entre parênteses vem o nome do pacote entre aspas. O leitor deve digitar o seguinte no *console*:

- `install.packages("urca")` - *Unit root and cointegration tests for time series data* (Pfaff et al., 2016).
- `install.packages("TSA")` - *Time Series Analysis* (Chan & Ripley, 2012).
- `install.packages("forecast")` - *Forecasting Functions for Time Series and Linear Models* (Hyndman et al., 2012).
- `install.packages("lmtest")` - *Testing Linear Regression Models* (Zeileis & Hothorn, 2002).
- `install.packages("normtest")` - *Tests for Normality* (Gavrilov & Pusev, 2014).
- `install.packages("FinTS")` - *Companion to Tsay (2005) Analysis of Financial Time Series* (**?**).
- `install.packages("xlsx")` - *Read, write, format Excel 2007 and Excel 97/2000/XP/2003 files* (Dragulescu, 2014).

Após a instalação, é preciso usar a função `require()` para carregar os pacotes, mas faremos isso ao longo do texto para que fique claro para o leitor em quais pontos estamos usando os pacotes.

Análise Exploratória da ST Vendas de Passagens Aéreas

Nessa seção vamos ler a ST de vendas de passagens aéreas, analisar a sazonalidade e fazer uma decomposição clássica da série temporal para saber quais são os principais componentes da mesma.

Leitura da ST no R

Por ser uma ST conhecida, o R já a disponibiliza na sua base de dados, tornando-se muito fácil a sua leitura. Basta executar o seguinte comando:

```
> data(AirPassengers)
```

Após ler a ST, façamos um gráfico.

```
> ts.plot(AirPassengers, ylab = "Vendas de Passagens Aéreas", xlab = "Anos")
```

Observando o gráfico da ST de vendas de passagens aéreas (Figura 5.1), podemos perceber que há uma tendência crescente do número de passageiros. As oscilações de picos e vales podem ser relacionadas às estações do ano, nas quais, mais especificamente, temos períodos de férias, feriados, etc. Essas oscilações, como observadas, acontecem anualmente, o que nos faz acreditar que há presença de sazonalidade. Do começo do ano a outubro percebemos um comportamento crescente, seguido de um comportamento decrescente da série, que permanece até dezembro. Isso se repete todos os anos.

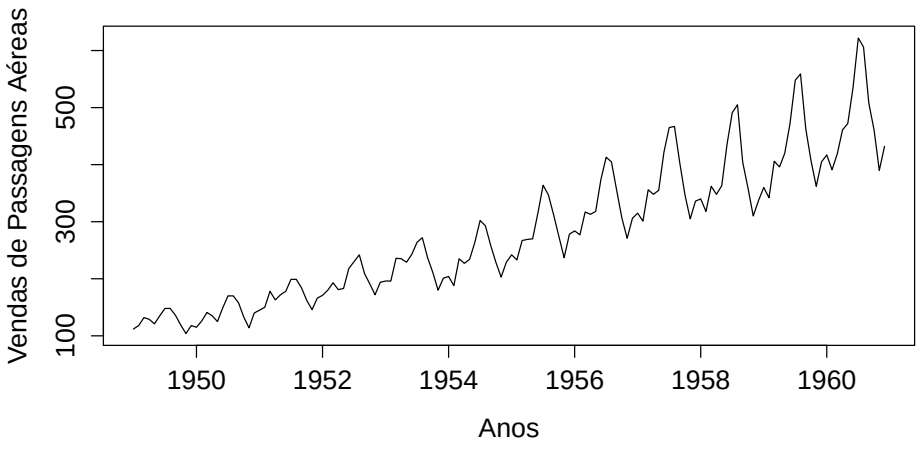


Figura 5.1: ST Mensal de Vendas de Passagens Aéreas (em milhares)

Nesse sentido, apenas observando o gráfico podemos "levantar" as seguintes hipóteses sobre essa ST:

- **Tendência:** parece haver aumento do número de passageiros transportados pela Pam Am ao longo dos anos. Esse comportamento é coerente com a teoria econômica, pois espera-se que ao longo do tempo a empresa cresça e, consequentemente, aumente as vendas de passagens aéreas.
- **Variância:** observa-se que, além do aumento do número de passagens vendidas, a distância entre os meses com maiores e menores vendas também está aumentando, indicando aumento da variância. Fato este também coerente com a teoria econômica pois ao aumentar o volume de vendas, espera-se maiores oscilações em relação ao valor médio.
- **Sazonalidade:** verifica-se um comportamento sazonal das vendas de passagens aéreas. Isto é, nos meses de março (feriado de Páscoa) e julho (Dia da Independência e férias escolares) há um aumento nas vendas quando comparado com os meses anteriores. Além disso, parece que a sazonalidade é crescente ao longo do tempo.

Observe que a análise gráfica nos permitiu conhecer a nossa ST e é uma fase muito importante para esse tipo de modelagem. Obviamente, como bons econometristas que somos, iremos testar estatisticamente todos os pontos levantados anteriormente. Antes disso, vamos tentar entender um pouco mais o comportamento sazonal da nossa ST.

Uma análise um pouco mais profunda da sazonalidade

O gráfico *monthplot* ajuda a detectar visualmente a presença de sazonalidade na ST. Como se pode verificar, esta ST apresenta média e variância não constantes, indícios de não estacionariedade na parte sazonal da ST.

```
> monthplot(AirPassengers,ylab = "Vendas de Passagens Aéreas", xlab = "Meses")
```

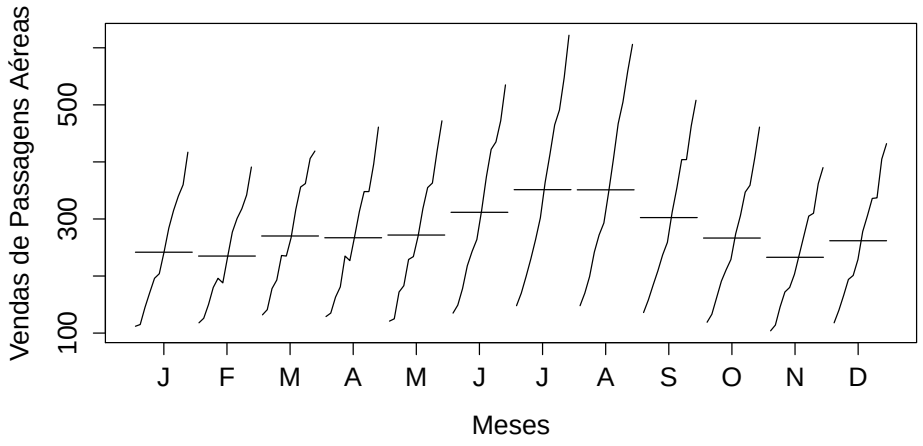


Figura 5.2: ST de Vendas de Passagens Aéreas por mês (em milhares)

Observando o gráfico (Figura 5.2), podemos ver que o número médio de passageiros (traços horizontais) aumenta nos meses de férias (indício de sazonalidade). Analisando os traços verticais, verifica-se um aumento contínuo na venda de passagens aéreas ano a ano, indício de não estacionariedade na parte sazonal da Série Temporal.

Decomposição da ST

Por fim, seguindo a decomposição clássica, decompõe-se a ST utilizando filtros de médias móveis em três componentes principais:

- tendência + ciclo

- sazonalidade

- resíduo (componente irregular, inovação)

```
> plot(decompose(AirPassengers))
```

Conforme observado no gráfico da ST (Figura 5.3), verifica-se que o número mínimo de passagens vendidas foi de 104 (Nov-1949) e o máximo de 622 (Jul-1960). Ao observar o componente de tendência,

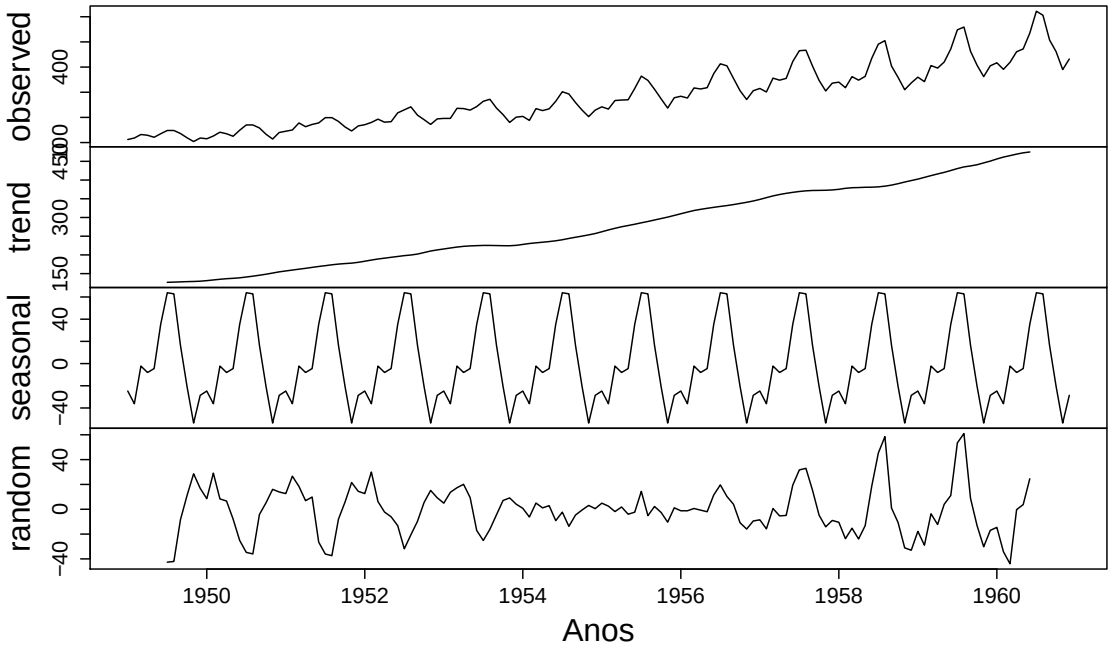


Figura 5.3: Decomposição da ST de Vendas de Passagens Aéreas (em milhares)

observa-se que a ST é fortemente afetada por esse componente (em torno de 85%). Com relação ao componente sazonal, verifica-se que o mesmo também está presente nessa ST e gira em torno de 10%. Sobrando uma pequena parte de componente irregular, o qual é levemente "contaminado" pela parte sazonal, mostrando que o método de decomposição utilizado não é muito eficiente.

Essa análise é interessante pois mostra que, basicamente, precisamos modelar as componentes de tendência e sazonalidade (em torno de 95% da ST), componentes "bem" modelados pelos modelos SARIMA(p,d,q)(P,D,Q). Este fato mostra porque essa ST é tão utilizada para exemplificar o uso dessa metodologia.

Prosseguindo, nosso próximo passo será testar estatisticamente as percepções levantadas anteriomente. Isto é, a ST de vendas de passagens aéreas é realmente não estacionária na parte não sazonal? E na parte sazonal? Como faremos para "corrigir" esses "problemas"?

Conhecendo a ST antes de iniciar a modelagem BJ

Para responder aos questionamentos feitos na seção anterior, abordaremos os seguintes tópicos:

1. Testar a estacionariedade da parte não sazonal
2. Testar a estacionariedade da parte sazonal

Testando a estacionariedade da parte não sazonal

Há, basicamente, quatro maneiras de observar se a ST em estudo é ou não estacionária:

- Análise gráfica;
- Comparação da média e da variância da ST para diferentes períodos de tempo;
- Observação a FAC (Função de Autocorrelação);
- Testes de Raiz Unitária.

Já vimos que a análise gráfica nos mostrou indícios de não estacionariedade. Fica claro também que, se "fatiássemos" a ST e calculássemos as médias de cada ano, observaríamos uma tendência de alta nas médias, indicando não estacionariedade das mesmas.

Outra maneira de ver a não estacionariedade da ST é visualizando o gráfico da FAC. A figura a seguir mostra que as autocorrelações plotadas pela FAC não decrescem exponencialmente ou de forma senoidal conforme descrito pela teoria de Box & Jenkins. Esse é mais um indicativo de que a ST é não estacionária[1].

```
> require(TSA)
> acf(AirPassengers, lag.max = 36, drop.lag.0 = T)
```

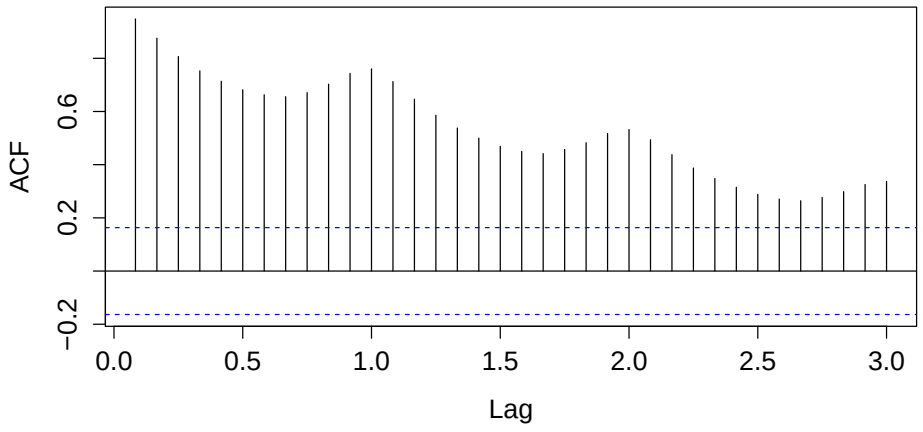


Figura 5.4: FAC: ST de Vendas de Passagens Aéreas (em milhares)

Nesse momento, o leitor atento pode estar se fazendo a seguinte pergunta: para que tantas

[1] O pacote **TSA** permite excluir o lag zero da FAC.

maneiras de se observar a estacionariedade se, ao observar o gráfico da ST (Figura 5.4), já está claro que a mesma é não estacionária? A resposta a esse questionamento é que nenhuma das maneiras vistas até o momento para verificar se a ST é ou não estacionária nos dá uma resposta "clara" (com significância estatística). Mais ainda, tais métodos não nos dizem quantas diferenciações precisaremos fazer na ST em estudo para torná-la estacionária e qual é o tipo de não estacionariedade (determinística ou estocástica). Para obter essas respostas precisamos testar a estacionariedade através dos testes de raiz unitária (RU).

Os testes de raíz unitária[2] (RU) foram uma grande revolução na Econometria na década de 1980. Existe uma grande quantidade de testes e, basicamente, todos têm a mesma ideia, isto é, a hipótese nula é que a série temporal possui uma raiz unitária (a ST é não-estacionária) e a hipótese alternativa é que a série é estacionária, com exceção do teste KPSS que tem as hipóteses alternadas. Abaixo podemos ver alguns exemplos de testes de raiz unitária:

- Augmented Dickey Fuller (ADF) (Dickey & Fuller, 1979)
- Phillips-Perron (PP) (Phillips & Perron, 1988)
- Kwiatkowski-Phillips-Schmidt-Shin (KPSS) (Kwiatkowski et al., 1992)
- Dickey Fuller GLS (DF-GLS) (Elliott et al., 1996)
- Elliott–Rothenberg-Stock point optimal (ERS) (**?**)

Apesar de haver uma grande quantidade de testes, nesse capítulo abordaremos apenas o teste de Dickey Fuller Aumentado (ADF), que tem as seguintes hipóteses:

H_0: a ST possui uma RU $\Leftrightarrow$ a série é não estacionária

H_1: a ST não possui RU $\Leftrightarrow$ a série é estacionária

A regra de rejeição da hipótese nula funciona da seguinte forma: se o valor observado para a estatística de teste for inferior ao valor crítico, rejeitamos a hipótese nula e, portanto, concluímos que a ST é estacionária de acordo com o nível de confiança adotado previamente. Caso contrário, a ST será não estacionária[3]. A distribuição da estatística de teste do teste ADF foram tabulados por MacKinnon (1996).

Como existem várias especificações consistentes com a não-estacionariedade, irão existir várias formas de testá-la. Na prática, a questão importante é escolher a forma do teste de RU adequada para

[2]Para maiores detalhes ver Hamilton (1994).

[3]Para maiores detalhes sobre processos não estacionários e os testes de Raíz Unitária de Dickey Fuller e Phillips Perron, recomenda-se consultar (Hamilton, 1994, capítulos 15, 16 e 17)

a ST em questão. O teste ADF apresenta as seguintes formas:

- Raiz unitária + constante + tendência determinística (R: *type = "trend"*)
- Raiz unitária + constante (R: *type = "drift"*)
- Raiz unitária (R: *type = "none"*)

Para executar o teste no R, usaremos a função `ur.df()` do pacote **urca** (Pfaff et al., 2016). Os principais argumentos dessa função são:

```
> ur.df(y, type = c("none", "drift", "trend"), lags = 1,
+       selectlags = c("Fixed", "AIC", "BIC"))
```

- `y`: ST em que será testada a raiz unitária;
- `type`: tipo da especificação do teste que o usuário deseja realizar;
- `lags`: número de defasagens a serem usadas para captar o comportamento da ST e, consequentemente, corrigir o problema da autocorrelação residual;
- `selectlags`: a função pode definir automaticamente, baseada em um critério de informação, o número de lags a serem inclusos dado um valor máximo no argumento *lags*.

Antes de iniciar o teste é importante observar que o número de lags que serão incluídos na equação do teste ADF será definido com base na análise dos resíduos da regressão e não somente nos critérios de informação.

Dando início aos testes, vamos testar a estacionariedade da ST considerando a equação sem tendência e com constante. É importante você saber que testamos a equação com tendência determinística antes, porém o parâmetro dessa variável não foi significativo. Nessa fase o parâmetro mais difícil e importante de definir é o lag, isto é, você precisa encontrar um número de lags que corrija a autocorrelação dos resíduos e seja parcimonioso com relação ao número de parâmetros da equação do modelo.

Estipulamos, inicialmente, o lag máximo como 24 e o critério de informação a minimizar sendo o AIC. A seguir observamos o resultado do teste ADF e a FAC do resíduos, a qual mostra que não há presença de autocorrelação.

```
> require(urca)
> adf.drift <- ur.df(y = AirPassengers, type = c("drift"),
+                    lags = 24, selectlags = "AIC")
> acf(adf.drift@res, lag.max = 36, drop.lag.0 = T)
> adf.drift@teststat #estatística de teste
             tau2     phi1
statistic 1.85818 7.914366
```

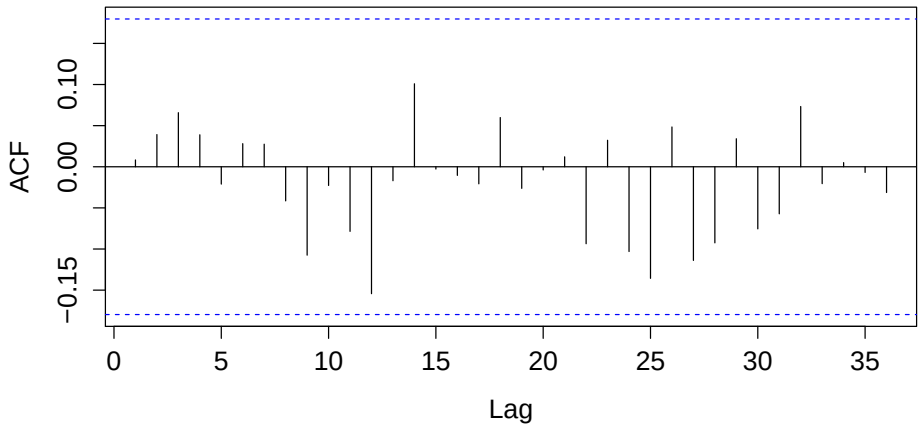


Figura 5.5: FAC dos Resíduos: ST de Vendas de Passagens Aéreas (em milhares)

```
> adf.drift@cval #valores tabulados por MacKinnon (1996)
      1pct  5pct 10pct
tau2 -3.46 -2.88 -2.57
phi1  6.52  4.63  3.81
```

Ao analisar a estatística de teste (tau2 = 1,8582) notamos que seu valor é superior ao valor crítico associado ao nível de confiança de 95% (-2,88). Dessa forma, conclui-se que a ST não é estacionária (não rejeição da hipótese nula).

O leitor pode visualizar mais informações sobre o teste de RU, como a equação ajustada por exemplo, usando a função `summary(adf.drift)`.

Ao concluir que a ST tem raiz unitária, precisamos descobrir o número de diferenciações necessárias para torná-la estacionária. É importante observar que esse é apenas um exercício para que o leitor observe o comportamento da ST e da FAC antes e após a diferenciação, pois, como veremos nas próximas seções, faremos as "correções" de não estacionariedade da ST na própria função que estimará o modelo SARIMA.

Dado que a nossa ST é não estacionária, vamos tentar torná-la estacionária fazendo uma diferenciação e vamos observar o gráfico (Figura 5.6) e a FAC (Figura 5.7) novamente.

```
> ts.plot(diff(AirPassengers, lag = 1, differences = 1))
> acf(diff(AirPassengers, lag = 1, differences = 1),
+     lag.max = 36, drop.lag.0 = T)
```

Observe que ao aplicar a diferenciação, a ST aparenta estar estacionária na média, mas a variância é crescente ao longo do tempo. Como sabemos, um dos pressupostos da teoria Box & Jenkins é que a ST seja também estacionária na variância. Para tal, iremos passar o log na ST em questão

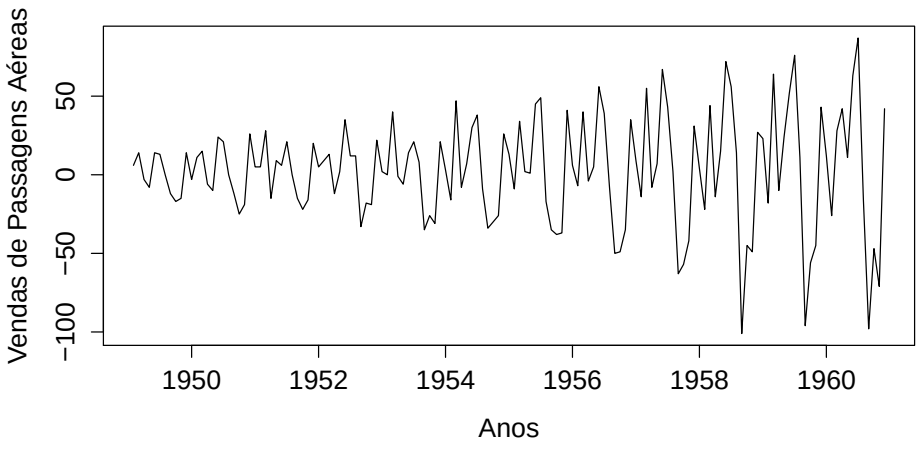


Figura 5.6: ST de Vendas de Passagens Aéreas (em milhares) com uma diferença

(Figura 5.8).

```
> ts.plot(diff(log(AirPassengers),lag = 1,differences = 1))
> acf(diff(log(AirPassengers), lag = 1, differences = 1),
+     lag.max=48, drop.lag.0=T)
```

Note agora que temos uma série temporal estacionária tanto na média quanto na variância. Ao analisarmos a FAC (Figura 5.9), a pergunta que fica é: essa FAC é adequada para idenficarmos a estrutura do nosso modelo SARIMA?

Avaliando a estacionariedade da parte sazonal

Com relação a pergunta feita na seção anterior, o leitor atento já observou que nos lags sazonais[4] a função de autocorrelação também apresenta um decrescimento lento, indicando que a ST é não estacionária na parte sazonal[5].

Para corrigir esse problema precisamos diferenciar a parte sazonal, para isso diferenciaremos a ST já diferenciada na parte não sazonal. Tal procedimento é feito mudando o parâmetro *lag* da função `diff()` para 12, conforme pode ser observado abaixo:

```
> acf(diff(diff(log(AirPassengers), lag = 1, differences = 1),
+          lag = 12, differences = 1), lag.max = 48, drop.lag.0 = T)
```

Observe que agora a FAC apresenta cortes bruscos nos lags 1 e 12 (Figura 5.10) e não apresenta mais decrescimento lento nem na parte sazonal nem na não sazonal.

[4]Observe que na FAC gerada pelo R os lags sazonais são 1(=12), 2(=24), 3(=36), etc.

[5]Existem testes estatísticos para avaliar a presença de não estacionariedade sazonal, um dos mais conhecidos é o teste de HEGY (Hylleberg et al., 1990).

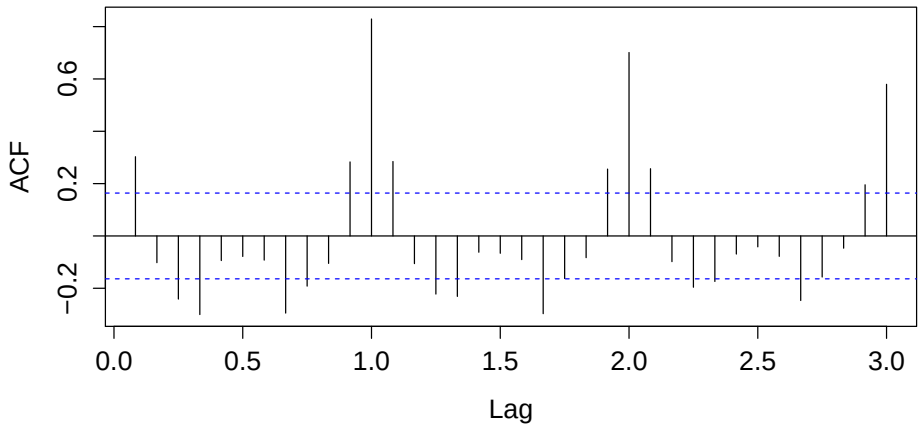


Figura 5.7: FAC: ST de Vendas de Passagens Aéreas (em milhares) com uma diferença

Vamos refazer o teste de RU para confirmar a estacionariedade da ST após aplicar as transformações anteriores. O valor da estatística de teste (tau2 = -4,0398) é inferior ao valor crítico (-2,88) ao nível de significância de 95%. Assim, podemos concluir que a série é estacionária.

```
> # Teste de RU na ST com diferenças sazonal e não sazonal
> adf.drift2 <- ur.df(y = diff(diff(log(AirPassengers), lag = 1), lag = 12),
+                               type = "drift", lags = 24, selectlags = "AIC")
> adf.drift2@teststat #estatística de teste
               tau2     phi1
statistic -4.039891 8.160779
> adf.drift2@cval #valores tabulados por MacKinnon (1996)
      1pct  5pct 10pct
tau2 -3.46 -2.88 -2.57
phi1  6.52  4.63  3.81
> acf(adf.drift2@res, lag.max = 36, drop.lag.0 = T)
```

Ao analisar a FAC para os resíduos do teste ADF (Figura 5.11), o leitor pode notar que alguns lags aparecem significativos, porém não são relevantes (apresentam correlação muito baixa). Dessa forma, confirmamos a validade do teste e podemos começar a nossa modelagem.

Modelando a ST

Séries temporais podem ser estacionárias ou não estacionárias, estocásticas ou determinísticas. Um processo estocástico Gaussiano é considerado fracamente estacionário se a média e a autocovariância não dependem do tempo, a última dependendo somente da distância temporal entre as observações (Hamilton, 1994).

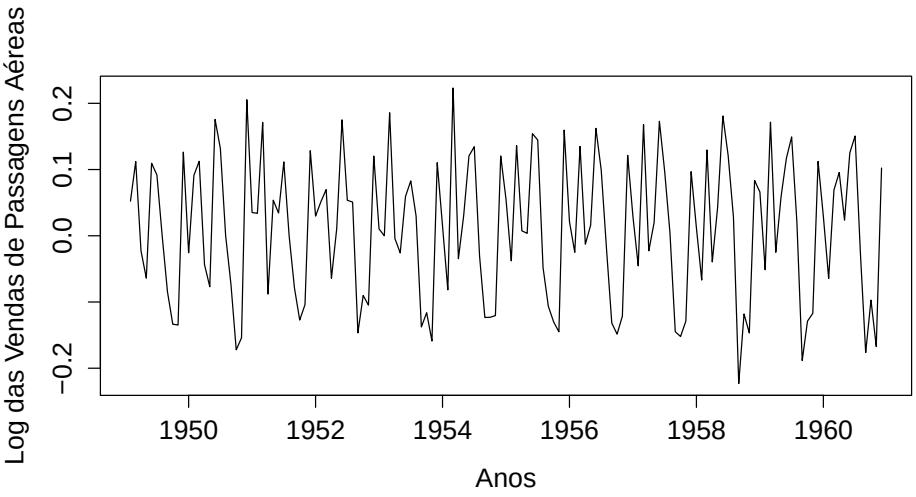


Figura 5.8: Logarítimo da ST de Vendas de Passagens Aéreas (em milhares) com uma diferença

Os modelos de Box & Jenkins são usados para séries originalmente estacionárias ou tornadas estacionárias por meio de diferenciação. Geralmente, as séries econômicas são não estacionárias, devendo ser diferenciadas até que fiquem estacionárias.

A metodologia Box & Jenkins para séries temporais estacionárias e construção dos modelos ARIMA segue um ciclo iterativo composto por cinco partes (Granger & Newbold, 1976):

1. **Especificação**: a classe geral de estruturas SARIMA(p,d,q)(P,D,Q) é analisada.
2. **Identificação**: com base na FAC e FACP amostrais e outros critérios são definidos os valores de p,q e P,Q.
3. **Estimação**: os parâmetros do modelo identificado são estimados e testados estatisticamente sobre sua significância.
4. **Diagnóstico**: faz-se uma análise dos resíduos (devem ser ruído branco) e testes de verificação (Ljung-Box) para ver se o modelo sugerido é adequado. Em seguida, verifica-s os modelos que apresentam menores valores para os critérios AIC e BIC. Caso haja problemas no diagnóstico, volta-se à identificação.
5. **Modelo definitivo**: para previsão ou controle. Verificar quais modelos têm as melhores medidas RMSE (*Root Mean Square Error*) e MAPE (*Mean Absolute Percentage Error*) (este não vale para dados próximos de zero, sendo preferível a utilização de outro método para a análise dos erros), por exemplo.

Um processo ARIMA(p,d,q) é um ARMA diferenciado d vezes até estar estacionário. Os modelos

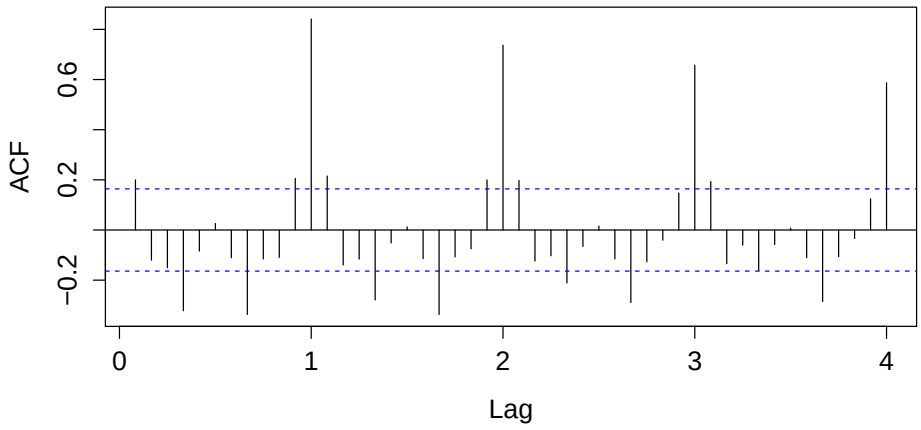


Figura 5.9: FAC: Logarítimo da ST de Vendas de Passagens Aéreas (em milhares) com uma diferença

SARIMA são usados para séries temporais que apresentam comportamento periódico em s espaços de tempo, isto é, quando ocorrem desempenhos semelhantes a cada intervalo de tempo Box & Jenkins (1970). Este é o caso da série a ser trabalhada neste capítulo.

Identificação

Como sabemos, o primeiro passo para identificar o nosso modelo SARIMA é observando a FAC e a FACP. Como os modelos propostos por Box & Jenkins (1970) são da década de 1970, o esforço computacional para estimar o modelo era muito grande, portanto essa fase era fundamental para se ter um modelo adequado à ST em análise. Atualmente, graças aos avanços computacionais, observar a FAC e a FACP é útil, principalmente, para se ter uma ideia inicial do modelo a ser testado, pois, como veremos mais adiante, o ideal é escolher um modelo que minimize os critérios de informação.

Assim, vamos observar a FAC e a FACP (função de autocorrelação parcial) da ST de vendas de passangens aéreas diferenciada na parte sazonal e não sazonal e com transformação logarítmica.

```
> layout(1:2)
> acf(diff(diff(log(AirPassengers), lag = 1, differences = 1),
+          lag = 12, differences = 1), lag.max = 48, drop.lag.0 = T)
> pacf(diff(diff(log(AirPassengers), lag = 1, differences = 1),
+          lag = 12, differences = 1), lag.max = 48)
```

Observando os gráficos (Figura 5.12) com um pouco de boa vontade podemos pensar nos seguintes modelos:

- **SARIMA(1,1,1)(1,1,1)** - corte brusco na FAC e na FACP nas partes sazonais e não sazonais;

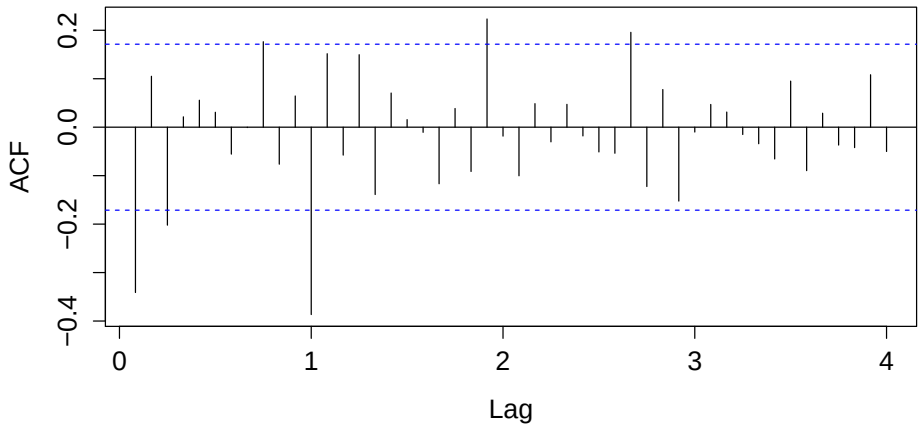


Figura 5.10: Logarítimo da ST de Vendas de Passagens Aéreas (em milhares) com diferença sazonal e não sazonal

- **SARIMA(0,1,1)(0,1,1)** - corte brusco na FAC e decrescimento das partes sazonais e não sazonais.

Uma vez identificado os possíveis modelos, passa-se para o próximo passo: a estimação.

Estimação

Para estimar o modelo, deve-se testar as possibilidades dos SARIMAs que idealizamos a partir da visualização da FAC e da FACP. Para tanto, utilizaremos a função `Arima()` do pacote **forecast**. Com relação ao método de estimação dos parâmetros neste trabalho, usaremos o *default* do R, que utiliza o método de Máxima Verossimilhança, denotado como **ML** (*Maximum Likelihood*).

Dessa forma, o primeiro modelo a ser estimado será uma SARIMA$(1,1,1)(1,1,1)_{12}$. Observe que na função `Arima()` a variável de entrada é a ST original, mas ajustar o argumento *lambda* em zero permite que seja feita a tranformação logarítmica na série. Também não é necessário diferenciar a ST antecipadamente pois o própria função faz isso.

```
> library("forecast")
> fit.air <- Arima(AirPassengers, order = c(1,1,1), seasonal = c(1,1,1),
+                  method = "ML", lambda = 0)
> fit.air
```

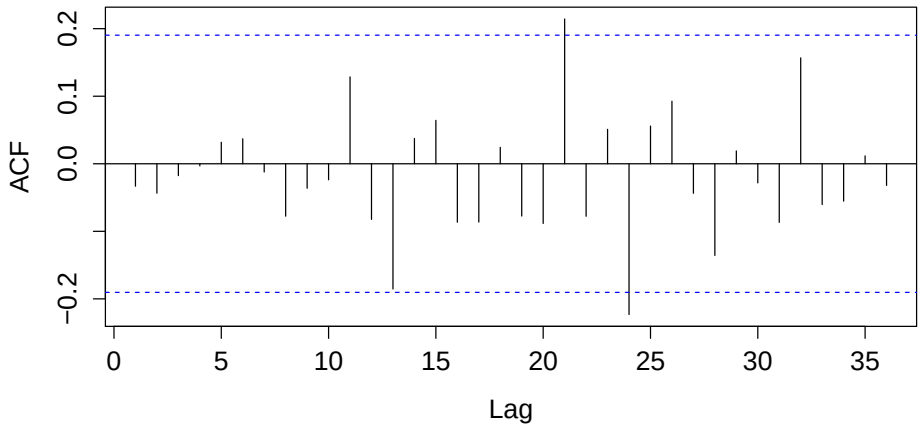
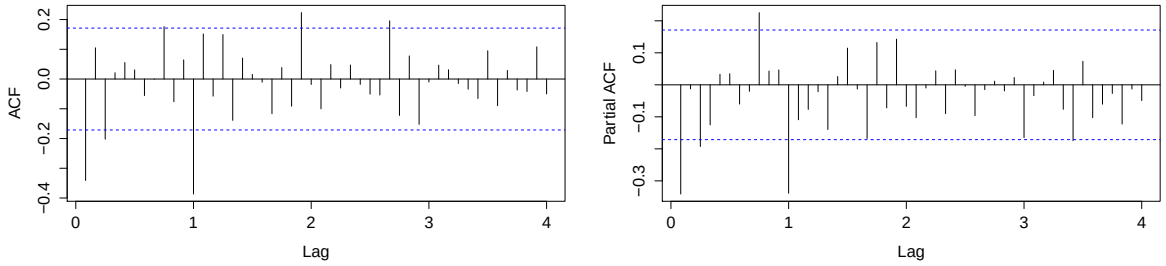


Figura 5.11: FAC : Diferença do Logarítimo da ST de Vendas de Passagens Aéreas (em milhares), com 1 diferença



(a) FAC (b) FACP

Figura 5.12: Logarítimo da ST de Vendas de Passagens Aéreas (em milhares) com diferença sazonal e não sazonal

```
Series: log(AirPassengers)
ARIMA(1,1,1)(1,1,1)[12]

Coefficients:
         ar1      ma1     sar1    sma1
      0.1668  -0.5616  -0.0994  -0.497
s.e.  0.2458   0.2114   0.1540   0.136

sigma^2 estimated as 0.00138:  log likelihood=245.16
AIC=-480.31   AICc=-479.83   BIC=-465.93
```

Para verificar, de forma rápida, se os parâmetros do modelo são significativos, desenvolvemos uma função no R chamada `t.test()`, o código da função está disponibilizado a seguir. Consideramos

nessa função o nível de confiança de 95%.

```
> # função de teste de significância dos parâmetros
> t.test <- function(modelo_arima){
+   # estatística t
+   coef <- modelo_arima$coef
+   se <- sqrt(diag(modelo_arima$var.coef))
+   t <- abs(coef/se)
+   # Teste t
+   ok <- t > qt(0.975, length(modelo_arima$x) -
+                  sum(modelo_arima$arma[c(1,2,3,4,6,7)]))
+   resul <- data.frame(Coef = coef, sd = se, t = t, rej_H0 = ok)
+   return(resul)
+ }
> # teste de significância para o modelo SARIMA(1,1,1)(1,1,1)12
> t.test(fit.air)
             Coef        sd         t rej_H0
ar1   0.16679124 0.2457980 0.6785705  FALSE
ma1  -0.56163441 0.2114211 2.6564723   TRUE
sar1 -0.09938487 0.1539918 0.6453907  FALSE
sma1 -0.49700743 0.1360485 3.6531644   TRUE
```

Como podemos observar, os parâmetros da parte AR não sazonal e sazonal são não significativos, logo, tais parâmetros não devem permanecer no modelo. Então, estes foram retirados e o modelo foi reestimado.

```
> fit.air <- Arima(AirPassengers, order = c(0,1,1), seasonal = c(0,1,1),
+                  method = "ML", lambda=0)


> fit.air
Series: AirPassengers
ARIMA(0,1,1)(0,1,1)[12]

Coefficients:
          ma1     sma1
      -0.4018  -0.5569
s.e.   0.0896   0.0731

sigma^2 estimated as 0.001371:  log likelihood=244.7
AIC=-483.4   AICc=-483.21   BIC=-474.77
> t.test(fit.air)
           Coef         sd        t rej_H0
ma1  -0.4018268 0.08964405 4.482470   TRUE
sma1 -0.5569466 0.07309948 7.619023   TRUE
```

Conforme pode ser observado, temos um modelo SARIMA$(0,1,1)(0,1,1)_{12}$ onde todos os parâmetros são significativos e que minimiza os critérios de informação (BIC, AIC e $AICc$)[6].

[6]Para maiores detalhes sobre os critérios de informação ver Akaike (1973) e Schwarz et al. (1978).

Diagnóstico

Após definir a "melhor" estrutura e estimar os parâmetros do modelo, outra etapa fundamental é a fase de diagnóstico do modelo. Nesta fase as seguintes características dos resíduos precisam ser analisadas e confirmadas:

- Ausência de autocorrelação linear;
- Ausência de heterocedasticidade condicional;
- Normalidade.

Para uma visão geral dos resíduos, utiliza-se a função `tsdiag()`. Esta disponibiliza a distribuição dos resíduos padronizados, a função de autocorrelação dos resíduos e os p-valores da estatística do teste Ljung-Box. Conforme podemos observar no primeiro gráfico a seguir, os dados aparentam estar distribuídos simetricamente em torno da média zero, indicação de distribuição normal. Observe também que não temos nenhuma informação discrepante (muito fora do intervalo [-3,3]). A única exceção é o resíduo de janeiro de 1954, neste caso, poderíamos testar se a venda de passagens aéreas nesse mês é um *outlier* ou não[7].

O segundo gráfico disponibilizado pela função `tsdiag()` é a FAC dos resíduos. Este gráfico é extremamente útil para observar se há a presença de autocorrelação linear nos resíduos. Conforme verificamos, não há nenhum *lag* significante, logo, toda a parte linear da ST de vendas de passagens aéreas foi capturada pelo modelo SARIMA$(0,1,1)(0,1,1)_{12}$.

O terceiro gráfico mostra o p-valor da estatística Ljung-Box para diferentes defasagens após a defasagem 14. De acordo com o gráfico, verificamos que não rejeitamos a hipótese nula da não existência de dependência serial para todas as defasagens. Tal resultado está em consonância com a análise feita anteriormente, isto é, não há dependência linear nos resíduos. Contudo, este gráfico não é confiável uma vez que os graus de liberdade usados para calcular os p-valores são baseados nos *lags* e não em (*lags* - (p+q)). Isto é, o processo usado para calcular os p-valores não leva em conta o fato de os resíduos terem sido gerados a partir de um modelo ajustado. Portanto, precisamos tomar cuidado ao observar esse gráfico.

```
> diag <- tsdiag(fit.air, gof.lag = 20)
```

Bem, conforme observamos a função `tsdiag()` já nos deu bastante informação sobre os nossos resíduos, entretanto, vamos realizar alguns testes estatísticos específicos para cada uma das características em decorrência do problema da estatística Ljung-Box e da necessidade de testarmos a normalidade

[7] Existem testes específicos para a detecção de *outliers*. Para maiores detalhes ver Chang et al. (1988) e Tsay (1988).

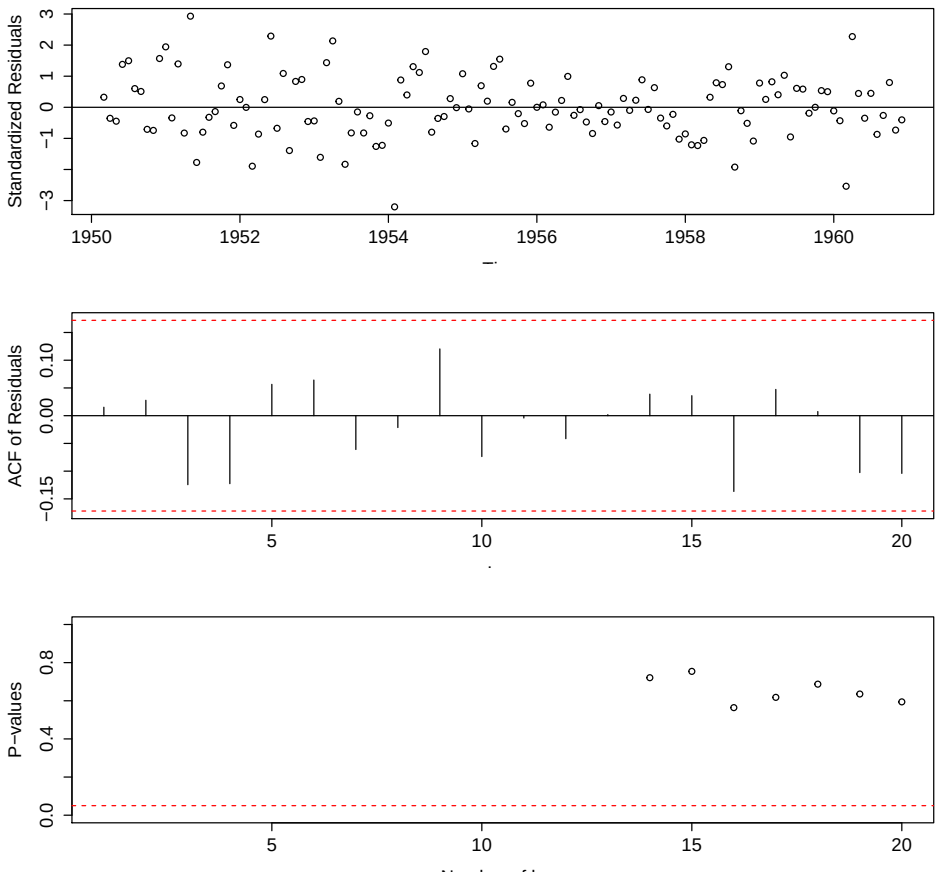


Figura 5.13: Características dos Resíduos

e a homocedasticidade dos resíduos.

Primeiramente, vamos testar a autocorrelação linear dos resíduos através do teste de **?**. Como sabemos o teste de Ljung Box nos dá a presença ou não de autocorrelação serial dos resíduos para o "L" primeiros *lags*. Outro teste de autocorrelação residual muito conhecido é o teste de Durbin & Watson[8], que testa a autocorrelação dos resíduos apenas para o primeiro lag.

```
> Box.test(x = fit.air$residuals, lag = 24,
+          type = "Ljung-Box", fitdf = 2)

        Box-Ljung test

data:  fit.air$residuals
X-squared = 26.446, df = 22, p-value = 0.233
```

Conforme podemos observar, o resultado do teste de Ljung Box mostra que a 95% de confiança

[8] Durbin & Watson (1950); Durbin & Watson (1951).

não rejeitamos a hipótese nula de não existência de autocorrelação serial até o *lag* 24[9]. É importante observar o argumento *fitdf*, neste caso igual a 2 ($p+q$), pois o teste é feito nos resíduos de um modelo SARIMA com dois parâmetros.

Confirmada a ausência de autocorrelação linear nos resíduos, vamos testar a estacionariedade da variância. Para tal, faremos o teste Multiplicador de Lagrange para heterocedasticidade condicional autorregressiva (ARCH LM) (Engle, 1984) disponível no pacote **FinTS**.

```
> require(FinTS)
> ArchTest(fit.air$residuals,lags = 12)
        ARCH LM-test; Null hypothesis: no ARCH effects

data:  fit.air$residuals
Chi-squared = 14.859, df = 12, p-value = 0.2493
```

Conforme mostrado pelo teste, a hipótese nula é que não há presença de efeito ARCH. Dessa forma, dado o valor do p-valor, não rejeitamos a hipótese nula a 95% de confiança, logo, a variância é estacionária.

Por fim, precisamos testar a normalidade do nosso resíduo. Para tal, faremos o teste de Jarque & Bera (1980) baseando-se no pacote **normtest**.

```
> require(normtest)
> jb.norm.test(fit.air$residuals, nrepl=2000)
        Jarque-Bera test for normality

data:  fit.air$residuals
JB = 5.2265, p-value = 0.0555
```

Os resultados mostram que a 95% de confiança não rejeitamos a hipótese nula de normalidade. Feito o diagnóstico dos resíduos, o nosso próximo passo será fazer as previsões.

Previsão

Após fazermos o diagnóstico dos resíduos e concluirmos que estamos modelando toda a parte linear da ST de vendas de passagens aéreas, o nosso próximo passo é fazer a previsão. Afinal de contas, esse é nosso objetivo final. Nessa etapa, basicamente, queremos conhecer a nossa previsão, saber qual é o intervalo de confiança (neste caso, 95%) e analisar as métricas de desempenho do modelo.

Para a previsão utilizaremos o pacote **forecast** e a função com o mesmo nome. Observe que, ao usar esta função, precisamos definir os seguintes parâmetros: (a) *object*: *output* do modelo SARIMA estimado; (b) h: horizonte de previsão (quantos passos à frente queremos prever); e (c) *level*: nível de confiança que desejamos para o nosso intervalo de confiança.

[9]Definido pelo próprio usuário.

```
> require(forecast)
> plot(forecast(object = fit.air, h=12, level = 0.95))
```

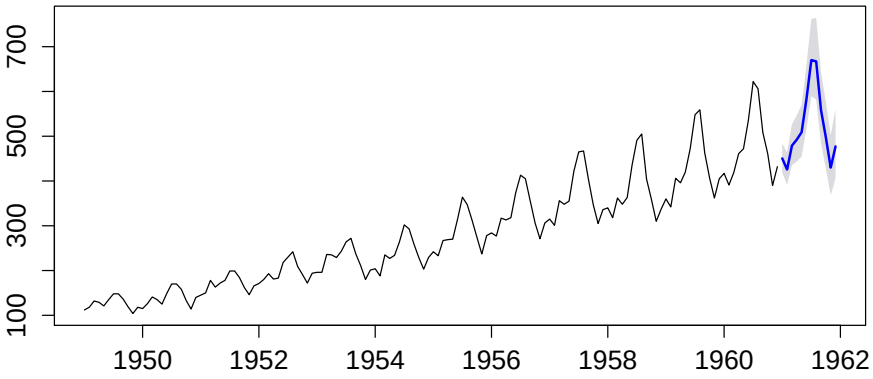


Figura 5.14: Previsão da ST de Vendas de Passagens Aéreas

Observando o gráfico (Figura 5.14), parece que fizemos uma "boa" previsão. Porém, uma maneira mais adequada de certificar isso é analisando as métricas de previsão. Conforme podemos observar, as métricas a seguir confirmam a análise gráfica. Analisando o MAPE, por exemplo, que é uma medida percentual do módulo dos erros e que não é contaminada pela escala da ST, observamos que o erro de previsão está apenas em 2,62%, o que é muito bom![10].

```
> accuracy(fit.air)
Training set error measures:
                      ME       RMSE        MAE        MPE      MAPE      MASE
Training set 0.000573059 0.03504882 0.02626034 0.01098891 0.4752816 0.2169522
```

Exportando as Previsões

Imagine o seguinte: você trabalha em uma empresa na área financeira e seu chefe lhe pede a previsão das vendas de um determinado produto para os próximos 12 meses. Ainda, imagine também que ninguém na sua empresa conheça o R (não é tão dificíl de imaginar isso, certo?!).

A solução para o primeiro problema você já tem e já aprendeu ao longo desse capítulo. A solução para o segundo problema pode ser treinar toda a equipe da área financeira para trabalhar com o R ou

[10]Dois pontos que gostaria de destacar aqui: primeiro, que uma análise da previsão fora da amostra seria importante para corroborar a performance do nosso modelo. Segundo, essa ideia de bom ou ruim é muito relativa, isto é, é sempre bom termos um modelo *benchmark* para compararmos nossas previsões.

então extrair as previsões e os intervalos de confiança estimados para um programa mais conhecido como o Excel através de um arquivo *.csv* ou *.xlsx*.

Como veremos, essa tarefa é muito fácil de fazer no R e pode ser executada com apenas uma linha de comando.

- Em formato *.csv*:
  ```
  > write.csv2(data.frame(prev),"previsao.csv")
  ```
- Em formato *.xlsx*:
  ```
  > require(xlsx)
  > write.xlsx(data.frame(prev),"previsao.xlsx")
  ```

Considerações finais

Neste capítulo aprendemos empiricamente como modelar uma série temporal mensal com base na metodologia proposta por Box & Jenkins utilizando o software R. Aprendemos como fazer uma análise exploratória de uma ST, quais são os possíveis "problemas" que ela pode ter para ser modelada utilizando o arcabouço proposto por Box & Jenkins e como "consertar" esses problemas, através, por exemplo, da diferenciação da ST.

Foram abordados também alguns "pacotes" úteis para esse tipo de modelagem, discutimos algumas funções e chamamos a atenção para algumas limitações das mesmas. Apesar de ter sido uma experiência interessante, sabemos que ainda ficaram faltando alguns pontos a serem abordados, como por exemplo, não tratamos da identificação e "correção" de possíveis *outliers*, não mostramos como "corrigir" a presença de heterocedasticidade condicional nos resíduos, quando ela existir, etc.

Nesse sentido, é importante que o leitor que estiver usando esse manual para construir o seu modelo SARIMA tenha ciência de suas limitações e busque, sempre que possível, aprofundar o seu conhecimento sobre o assunto.

Ajuste Sazonal

Daiane Marcolino de Mattos

Pedro Costa Ferreira

Introdução

Iniciaremos esse capítulo com uma breve descrição sobre a composição de uma série temporal. Segundo a *decomposição clássica*, uma série temporal pode ser decomposta em quatro componentes não observáveis: tendência, sazonalidade, ciclo e erro. A sazonalidade, que é o objeto de estudo desse capítulo, é causada por movimentos oscilatórios de mesma periodicidade que ocorrem em período intra-anual, como variações climáticas, férias escolares, feriados fixos, entre outros. A ocorrência desses eventos pode levar a conclusões inadequadas a respeito da série temporal em estudo. Por exemplo, a oferta de emprego costuma aumentar no final do ano devido às festas natalinas, isto é, há uma demanda maior por bens e serviços, elevando o nível de contratações de pessoas. Porém, como a maioria das vagas é temporária, geralmente, há diminuição no nível de pessoal ocupado no período seguinte. Para a análise econômica, o importante é detectar a diferença entre o que ocorre periodicamente e o que de fato ocorre de diferente naquele período específico, possibilitando observar a tendência e o ciclo da variável.

Dessa forma, precisamos de uma ferramenta adequada que consiga remover a componente sazonal. A remoção da sazonalidade de uma série temporal é chamada de ajuste sazonal ou dessazonalização. Nesse capítulo, nos dedicaremos a aprender como removê-la utilizando o programa de ajuste sazonal X-13ARIMA-SEATS (ou simplesmente X-13) desenvolvido e mantido por U.S. Census Bureau. O programa é mundialmente utilizado por órgãos de estatística e em alguns lugares ainda em sua versão anterior (X12-ARIMA). Iremos implementá-lo no software R e o aplicaremos na série temporal do índice de produção industrial geral do Brasil, esta estimada pelo Instituto Brasileiro de Geografia e Estatística (IBGE).

Embora nos dediquemos a apresentar o X-13, é importante que o leitor saiba que existem outras metodologias para remover a sazonalidade de uma série temporal. Veja alguns exemplos:

a. *Seasonal Dummies* (Zellner, 1979);
b. *Holt-Winters* (Rasmussen, 2004));
c. *Structural Models* (Harvey & Shepard, 1993; Plosser, 1979; Koopman et al., 2009);
d. *Dainties* (Fok et al., 2005);
e. *TRAMO-SEATS* (Gómez & Maravall, 1998; Hungarian Central Statistical Office, 2007);
f. *X-11, X-11ARIMA, X-12ARIMA* (Shiskin et al., 1967; Findley et al., 1998).

Para que o objetivo do capítulo seja atingido, dividimos o capítulo em outras quatro seções: na seção 6.2 apresentamos um breve resumo sobre o X-13; nas seções 6.3 e 6.4 o leitor verá o passo-a-passo sugerido para dessazonalizar uma série temporal no R; na seção 6.5 aplicaremos todo o procedimento aprendido na série temporal do índice de produção industrial geral do Brasil; e na seção 6.6 relatamos algumas considerações finais.

Breve resumo sobre o X-13ARIMA-SEATS

O X-13ARIMA-SEATS, criado em julho de 2012, é um programa de ajuste sazonal desenvolvido por U.S Census Bureau com o apoio do Bank of Spain. O programa é a junção dos softwares X12-ARIMA e TRAMO/SEATS com melhorias. As melhorias incluem uma variedade de novos diagnósticos que ajudam o usuário a detectar e corrigir inadequações no ajuste, além de incluir diversas ferramentas que repararam problemas de ajuste e permitiram um aumento na quantidade de séries temporais que podem ser ajustadas de maneira adequada (U.S. Census Bureau, 2015).

Um procedimento contido no X-13ARIMA-SEATS que merece destaque é o pré-ajuste da série temporal, isto é, uma correção antes de ser feito, de fato, o ajuste sazonal. Alguns eventos atípicos e/ou não sazonais como, por exemplo, efeitos do calendário (*trading days, working days, moving holidays*, etc), greves, catástrofes, entre outros, podem afetar o padrão sazonal da série temporal e, consequentemente, gerar um ajuste de qualidade inferior. O tratamento desses eventos (pré-ajuste) deve ser feito, se necessário. Um exemplo da aplicação de ajuste sazonal a partir do X-13ARIMA-SEATS com a utilização de *trading days* pode ser encontrado em Livsey et al. (2014).

Caso o leitor queira aprofundar-se a respeito do programa, recomendamos, além da literatura oficial, uma nota técnica produzida por nós sobre o mesmo tema, dessa vez aplicado às séries temporais da Sondagem da Indústria da Transformação (FGV/IBRE) (Ferreira et al., 2015).

Instalação dos pacotes necessários

Para que seja possível dessazonalizar uma série temporal no R com o X-13, é necessário apenas instalar e carregar o pacote **seasonal**, este desenvolvido por Sax (2015a). Nas primeiras versões implementadas, o usuário deveria fazer o download do programa executável do X-13 no site do Census (`https://www.census.gov/srd/www/x13as/`). Atualmente, no entanto, o download desse arquivo é feito automaticamente ao instalar o pacote, como é feito no código a seguir.

```
> install.packages("seasonal")
> library(seasonal)
```

Sugerimos que após a instalação e o carregamento do pacote, o leitor verifique se, de fato, poderá utilizar o X-13. Para isso, utilize a função `checkX13()` do pacote **seasonal**. Se a instalação funcionou corretamente, a função retorna a mensagem vista a seguir.

```
> checkX13()
seasonal is using the X-13 binaries provided by x13binary
X-13 installation test:
- X13_PATH correctly specified
- binary executable file found
- command line test run successful
- command line test produced HTML output
- seasonal test run successful
Congratulations! 'seasonal' should work fine!
```

Algoritmo de ajuste sazonal

Após executar os passos da seção 6.3, podemos iniciar as etapas de um ajuste sazonal que, segundo a literatura sugere, são:

1. Análise Gráfica;
2. Execução o X-13ARIMA-SEATS no modo automático;
3. Avaliação do ajuste sazonal em (2);
4. Correção do ajuste sazonal em (2) (se necessário).

A análise gráfica de uma série temporal permite visualizar suas características para uma boa modelagem, por exemplo: seu padrão sazonal, quebras estruturais, possíveis outliers, se há necessidade (e possibilidade) de usar transformação logarítmica nos dados.

O X-13 funciona, basicamente, em duas etapas: pré-ajuste e ajuste sazonal. Na primeira, o software pode corrigir a série de efeitos determinísticos. É nesta etapa que o usuário pode especificar, por exemplo, *outliers* e efeitos do calendários (Páscoa, Carnaval, etc). Na segunda etapa é feita a estimação da componente sazonal e dessazonalização.

A execução do programa no modo automático pode trazer um ajuste sazonal de boa qualidade. Nesse modo, o programa verifica, entre outras coisas, se há necessidade de transformação log nos dados; se existem *outliers (additive, level shift e temporary change)*; a ordem do modelo ARIMA; se há efeitos de calendário). Essas verificações automáticas podem poupar o tempo do usuário e ajudá-lo na

escolha de um bom modelo, principalmente na etapa do pré-ajuste. No entanto, este modelo precisa ser avaliado e o X-13ARIMA-SEATS fornece algumas ferramentas[1] para essa finalidade:

- **QS statistic:** verifica a existência de sazonalidade em uma série temporal. A Tabela 6.1 resume em quais séries temporais o programa calcula o teste de sazonalidade. Em um bom ajuste sazonal, o diagnóstico dado pela estatística QS permitiria concluir indícios de sazonalidade somente na série original (corrigida por *outliers*) e não nas restantes.

  É importante saber que se a série possui mais de 8 anos de dados mensais (96 observações) o teste de sazonalidade é executado na série temporal inteira e também nos últimos oito anos; caso contrário, é executado apenas na série completa.

Codificação	**Significado**
qsori	série original
qsorievadj	série original corrigida por *outliers*
qsrsd	resíduos do modelo SARIMA
qssadj	série com ajuste sazonal
qssadjevadj	série com ajuste sazonal corrigida por *outliers*
qsirr	componente irregular
qsirrevadj	componente irregular corrigida por *outliers*

Tabela 6.1: Séries temporais disponíveis para o diagnóstico dado pela estatística QS

- **Ljung-Box statistic:** o teste de Ljung & Box (1978) verifica a existência de autocorrelação (hipótese nula) em uma série temporal. O X-13 mostra o resultado desse teste aplicado aos resíduos do modelo SARIMA estimado na defasagem 24. Espera-se que os resíduos não sejam autocorrelacionados.
- **Shapiro-Wilk statistic:** o teste de Shapiro & Wilk (1965) verifica se a distribuição de um conjunto de dados é normal (hipótese nula). O X-13 mostra o resultado desse teste aplicado aos resíduos do modelo SARIMA. Espera-se que os resíduos sigam distribuição normal.
- **Gráfico SI ratio:** útil para verificar se a decomposição das componentes da série temporal foi feita adequadamente. Espera-se que os fatores sazonais acompanhem o SI (componentes sazonal e irregular agregadas[2]), indicando que o SI não é dominado pela componente irregular.

[1] Há uma gama de recursos oferecidos pelo X-13ARIMA-SEATS que não ainda não foram explorados nesse capítulo. Mais informações ver X13-ARIMA-SEATS Reference Manual Acessible HTML Output Version (U.S. Census Bureau, 2015).

[2] Se for utilizada a decomposição aditiva (sem transformação log) então SI é a soma da componente sazonal e da componente irregular S+I. Caso contrário, usa-se a multiplicação: S×I.

- **Gráfico Spectral**[3]**:** é uma ferramenta que alerta se a série temporal possui influência de efeitos sazonais e de *trading days*. O gráfico é feito para a série original, para a série com ajuste sazonal (se o ajuste sazonal for executado), para a série da componente irregular e para os resíduos do modelo SARIMA. Se o objetivo é realizar um ajuste sazonal na série temporal, então é esperado a identificação de efeitos sazonais no gráfico spectral da série original. Se o ajuste sazonal foi feito adequadamente, espera-se que tais efeitos não sejam encontrados nas séries disponíveis restantes.

Após a análise de todas as ferramentas de diagnóstico, caso alguma não conformidade seja detectada no modelo automático, o usuário deve reajustar o modelo e diagnosticá-lo novamente. Algumas alterações que podem ajudar a melhorar o ajuste são: rever a necessidade de transformação nos dados (isso pode estabilizar a variância); modificar a ordem do modelo SARIMA; e inserir ou retirar outliers e/ou variáveis de regressão.

Aplicação no Índice de Produção Industrial

Nessa seção o leitor aprenderá a dessazonalizar a série temporal do Índice de Produção Industrial (IBGE, 2015b). Escolhemos essa variável pois a produção industrial de uma região geralmente é afetada pelas épocas do ano. Por volta de outubro, por exemplo, é esperado um aumento no nível de produção devido às comemorações natalinas em dezembro. Nos meses seguintes, no entanto, é esperado uma queda desse nível. Com ajuste sazonal, a série de produção industrial poderá ser interpretada sem os efeitos do calendário, permitindo realizar comparações entre os meses de forma adequada.

O índice de produção industrial geral do Brasil é estimado mensalmente pelo Instituto Brasileiro de Geografia e Estatística (IBGE) pela Pesquisa Industrial Mensal - Produção Física (PIM-PF) desde a década de 1970. Os dados podem ser descritos como um índice sem ajuste sazonal com base fixa em 2012 (média de 2012 = 100) e compreendem o espaço de tempo de janeiro de 2002 a dezembro de 2014, totalizando 156 observações.

Os dados podem ser extraídos do sistema SIDRA do IBGE e também em `https://github.com/pedrocostaferreira`. Após o download em formato .csv, leia o arquivo no R com a função `read.csv2()` e, em seguida, transforme-o em um objeto de séries temporais utilizando a função `ts()` como é feito nos próximos comandos:

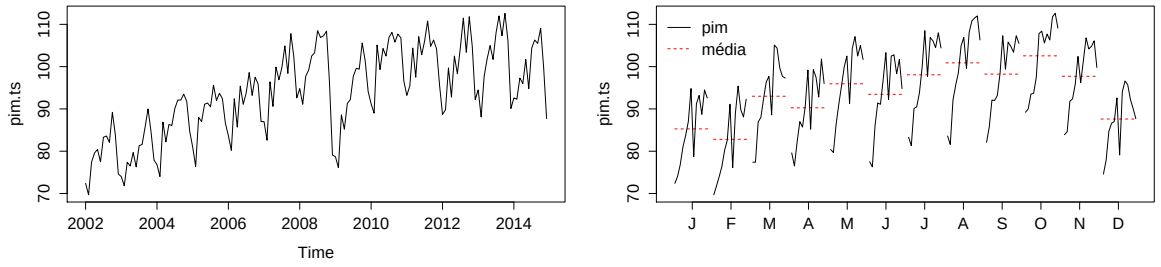
```
> pim <- read.csv2("pimpf.csv")
> pim.ts <- ts(pim, start = c(2002,1), freq = 12)
```

[3] Veja mais detalhes sobre o gráfico spectral em U.S. Census Bureau (2015) (capítulo 6).

Agora podemos executar os quatro passos do algoritmo da seção 6.4.

Análise Gráfica

Para esboçar de forma simples o gráfico de uma série temporal, utiliza-se a função `plot()`. Outro gráfico que pode ajudar a entender o comportamento de séries temporais é dado pela função `monthplot()`. Nele é possível comparar a série histórica de cada mês do ano. A análise gráfica da série temporal (Figura 6.1) permite supor que o índice de produção industrial:



(a) Série histórica (`plot`)

(b) Série histórica por mês (`monthplot`)

Figura 6.1: Análise gráfica do índice de produção industrial (IBGE,2015)

a. tem característica sazonal, uma vez que de janeiro a outubro o índice tem comportamento crescente e nos outros dois meses decrescente. Esse comportamento se repete em todos os anos;

b. apresentava uma tendência crescente antes da crise econômica (de 2008) e parece estar estável (sem crescimento ou quedas expressivas) após tal acontecimento;

c. parece ter variação constante, não necessitando que os dados sejam transformados para estabilizá-la;

d. foi extremamente afetado pela crise econômica de 2008. Notamos que o índice atingiu um valor discrepante em relação ao comportamento habitual.

Para esboçar a Figura 6.1, o leitor pode executar os comandos a seguir.

```
> plot(pim.ts)
> monthplot(pim.ts, col.base = 2, lty.base = 2)
> legend("topleft", legend = c("pim", "média"),
+        lty = c(1,2), col = c(1,2), bty = "n")
```

Execução do X-13ARIMA-SEATS no modo automático

A função `seas()` do pacote **seasonal**[4] desempenhará o papel de efetuar o ajuste sazonal tanto no modo automático como com especificações definidas de acordo com a necessidade do usuário.

seas(x, xreg = NULL, xtrans = NULL, seats.noadmiss = "yes", transform.function = "auto", regression.aictest = c("td", "easter"), ...)

Especificamos os principais argumentos da função `seas()` a seguir.

- `x`: série temporal de interesse;
- `arima.model`: permite especificar o modelo SARIMA para a série de interesse;
- `outlier`: permite definir se o programa deve ou não detectar automaticamente *outliers*;
- `regression.variables`: permite especificar *outliers* e variáveis de calendário como Páscoa, *trading days*, ano bissexto, entre outras variáveis[5] já disponibilizadas pelo X-13;
- `regression.aictest`: permite definir se o programa deve ou não detectar automaticamente variáveis de regressão;
- `transform.function`: permite especificar a transformação que deve ser aplicada na série de interesse como, por exemplo, `log` (transformação logarítmica), `none` (nenhuma transformação) ou `auto` (o programa define se deve ser ou não aplicada a transformação log).

A função `seas()`, no entanto, requer apenas a série temporal em que se pretende fazer o ajuste sazonal para o seu funcionamento. Os outros argumentos, nem todos exemplificados anteriormente, funcionarão no modo automático. Logo, para executar o ajuste sazonal em um objeto de série temporal `x` no modo automático, o usuário pode utilizar `seas(x)`.

```
> (ajuste <- seas(pim.ts))
```

[4] Mais detalhes sobre o pacote ver Sax (2015b).

[5] Outras variáveis pré-definidas podem ser encontradas em X-13ARIMA-SEATS Reference Manual Acessible HTML Output Version (U.S. Census Bureau, 2015, cap. 7, pág. 144-147).

```
Call:
seas(x = pim.ts)

Coefficients:
Mon             Tue             Wed             Thu
0.0055644       0.0053722       0.0022402       0.0053296
Fri             Sat             Easter[1]       LS2008.Dec
0.0004133      -0.0002806      -0.0242756      -0.1334241
AO2011.Feb      AO2014.Feb      MA-Seasonal-12
0.0612627       0.0646419       0.6791109
```

No modelo ajustado automaticamente foram detectados efeitos da Páscoa, de dias da semana e também de *outliers*. Foi detectado um outlier *level shift* no mês de dezembro de 2008 (`LS2008.Dec`), mês extremamente afetado pela crise econômica. Outros dois *outliers*, não esperados visualmente pela análise da Figura 6.1, foram detectados em fevereiro de 2011 (`AO2011.Feb`) e 2014 (`AO2014.Feb`). Esses são do tipo aditivo. O próximo passo é avaliar esse ajuste.

Avaliação do ajuste automático

Para avaliar o ajuste sazonal feito no modo automático, vamos precisar dos resultados do tópico 3 apresentados na seção 6.4. Um breve resumo desses resultados são obtidos pela função `summary()`.

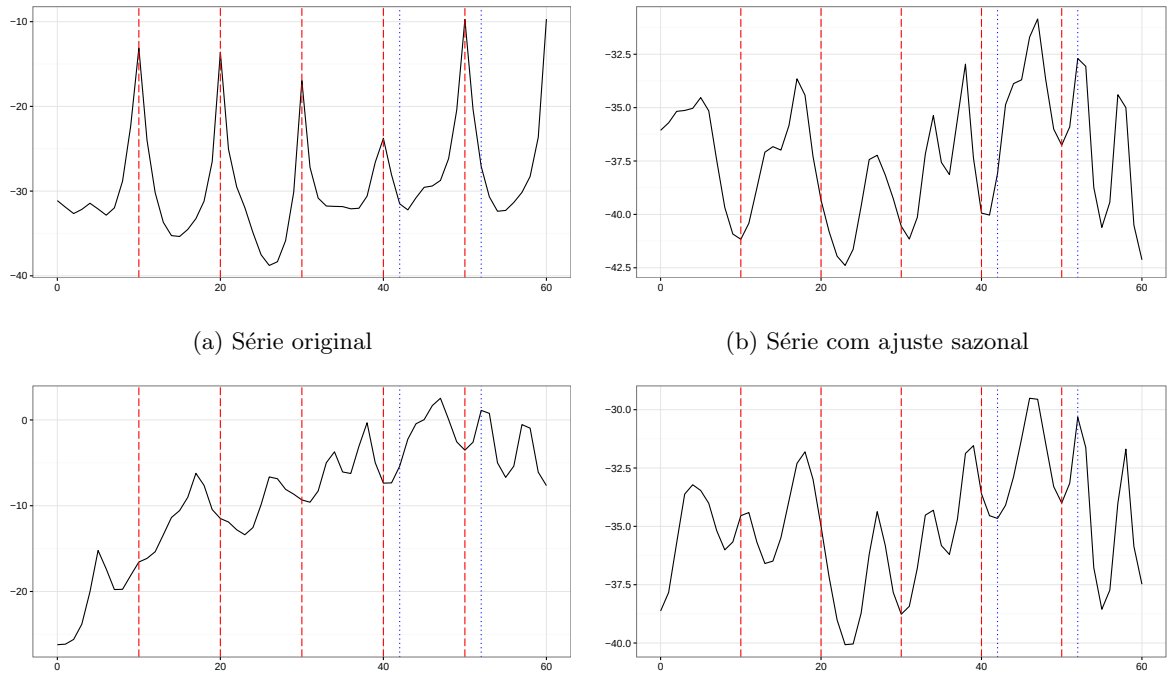
```
> summary(ajuste)
Call:
seas(x = pim.ts)

Coefficients:
Estimate Std. Error z value Pr(>|z|)
Mon             0.0055644  0.0023379   2.380   0.0173 *
Tue             0.0053722  0.0023763   2.261   0.0238 *
Wed             0.0022402  0.0022930   0.977   0.3286
Thu             0.0053296  0.0023305   2.287   0.0222 *
Fri             0.0004133  0.0023074   0.179   0.8578
Sat            -0.0002806  0.0023164  -0.121   0.9036
Easter[1]      -0.0242756  0.0043743  -5.550 2.86e-08 ***
LS2008.Dec     -0.1334241  0.0182831  -7.298 2.93e-13 ***
AO2011.Feb      0.0612627  0.0129063   4.747 2.07e-06 ***
AO2014.Feb      0.0646419  0.0140587   4.598 4.27e-06 ***
MA-Seasonal-12  0.6791109  0.0695104   9.770  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

SEATS adj.  ARIMA: (0 1 0)(0 1 1)  Obs.: 156  Transform: log
AICc: 613.9, BIC: 647.1  QS (no seasonality in final):    0
Box-Ljung (no autocorr.): 23.32   Shapiro (normality): 0.9814 *
```

O leitor pode ver que foi ajustado um modelo SARIMA$(0\,1\,0)(0\,1\,1)$. O parâmetro `MA-Seasonal-12` foi significativo ao considerar nível de significância de 5%. O mesmo pode ser dito para o efeito dos

três *outliers* e da variável que reflete a Páscoa (`Easter[1]`). Embora nem todos os dias da semana sejam significativos considerando nível de 5% de significância, três dias foram (`Mon, Tue` e `Thu`), e isso é suficiente para mantê-los no modelo e concluir que há indícios de que a produção industrial seja afetada pelos dias da semana.

O leitor também pode observar que a hipótese de normalidade dos resíduos foi rejeitada com 95% de confiança e a transformação log foi aplicada na série original, embora no início tenhamos acreditado que isso não era necessário ao analisar a série graficamente. O teste de Ljung & Box, sugere não haver evidências de autocorrelação residual até o lag 24.

Além desses resultados, criaremos um gráfico spectral (Figura 6.2) para analisar se há efeitos da sazonalidade e de *trading days*. O gráfico pode ser feito no R após extrair o *spectral output* utilizando a função `series()` do pacote **seasonal** e transformá-lo em um objeto da classe `data.frame`. Você pode usar o código, a seguir, para extrair o *spectral output* para a série original[6] (`sp0`).



(a) Série original

(b) Série com ajuste sazonal

(c) Série da componente irregular

(d) Resíduos do modelo ARIMA

Figura 6.2: Análise spectral do ajuste sazonal

```
> spec.orig <- data.frame(series(ajuste, "sp0"))
```

O gráfico pode ser feito utilizanto o pacote **ggplot2** com as configurações abaixo.

[6]Para observar a codificação para as outras séries além da série original ver (U.S. Census Bureau, 2015, cap. 7, pág. 194).

```
> library(ggplot2)
> ggplot(aes(x = 0:60, y = X10.Log.Spectrum_AdjOri.),
+        data = spec.orig, colour = "black") +
+   geom_line() +
+   geom_vline(colour = "red", xintercept = c(10, 20, 30, 40, 50), linetype = 5) +
+   geom_vline(colour = "blue", xintercept = c(42, 52),  linetype = 3) +
+              ylab(" ") + xlab(" ") + theme_bw() +
+   ggtitle("Spectral plot of the first-differenced original series") +
+   theme(plot.title = element_text(lineheight=2, face="bold",size = 16))
```

O gráfico spectral (Figura 6.2) mostra indícios de efeitos de sazonalidade na série original (Figura 6.2a), visto que nas frequências sazonais (10, 20, 30, 40, 50) a série temporal toma forma de picos bem definidos. Já nas frequências de *trading days* (linha pontilhada em azul, aproximadamente 42 e 52) não se pode concluir o mesmo pois não há forma de picos. Para as outras três séries não foi detectado picos de sazonalidade, porém, há leves indícios de efeitos de *trading days*, o que é estranho pois foi incluído o efeito de *trading days* no modelo do ajuste.

As suposições de sazonalidade na série original não são rejeitadas com a análise da estatística QS. Os testes de sazonalidade nas demais séries (Tabela 6.1 da seção 6.4), podem ser vistos utilizando a função `qs()` do pacote **seasonal**:

```
> qs(ajuste)
                    qs   p-val
qsori        162.39311 0.00000
qsorievadj   236.41305 0.00000
qsrsd          0.02336 0.98839
qssadj         0.00000 1.00000
qssadjevadj    0.00000 1.00000
qsirr          0.00000 1.00000
qsirrevadj     0.00000 1.00000
qssori        81.19590 0.00000
qssorievadj  136.51723 0.00000
qssrsd         0.00000 1.00000
qsssadj        0.00000 1.00000
qsssadjevadj   0.00000 1.00000
qssirr         0.00000 1.00000
qssirrevadj    0.00000 1.00000
```

Uma vez que a série `pim.ts` apresenta mais de 96 observações, o teste de sazonalidade foi calculado para a série completa e para os 8 anos mais recentes. Nota-se que o p-valor é pequeno tanto para série original (`qsori`) como para a série original corrigida por outliers (`qsorievadj`), isto é, não há evidências de que a série do índice de produção industrial não seja sazonal. Nas demais séries, pelo p-valor ser próximo de 1, conclui-se o contrário: há evidências de não sazonalidade nas séries temporais. Assim, se tratando de sazonalidade, o X-13ARIMA-SEATS cumpriu bem o seu dever de removê-la.

Mais uma ferramenta para avaliar a qualidade do ajuste sazonal é dado pelo gráfico *SI ratio*. Para esboçá-lo, utiliza-se a função `monthplot()`.

```
> monthplot(ajuste, col.base = 1, lty.base = 2, lwd.base = 2)
> legend("topleft", legend = c("SI", "Fator sazonal", "Média Fator Sazonal FS"),
+        cex = 0.7, lty = c(1,1,2), col = c(4,2,1), lwd = c(1,2,2), bty = "n")
```

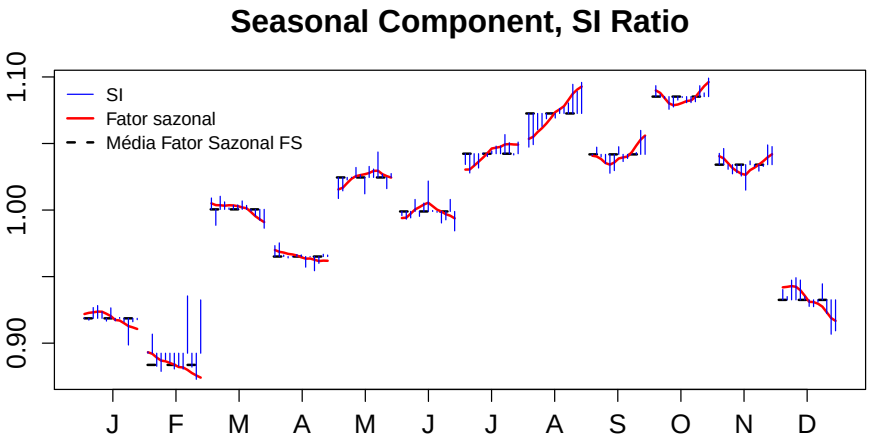


Figura 6.3: Fatores sazonais e SI *ratio*

Na Figura **??** as linhas azuis (verticais) referem-se à componente SI ratio (componentes sazonal e irregular agregadas). As linhas vermelhas (realçadas) representam os fatores sazonais e a linha tracejada é a média dos fatores sazonais naquele mês. Pode-se observar que o fator sazonal tende a acompanhar o SI ratio. Isso significa que a componente SI ratio não é dominada pela componente irregular, isto é, os erros tem um comportamento estável em torno de zero e a decomposição das componentes não observáveis da série temporal foi feita adequadamente. No entanto, note que para o mês de fevereiro (em que dois outliers foram encontrados) o SI ratio é dominado pela componente irregular.

Pode-se concluir, então, que o ajuste sazonal automático já forneceu bons resultados, porém, como alguns pressupostos necessários (normalidade dos resíduos) não foram confirmados estatisticamente, o modelo precisa ser especificado com mais detalhes.

Correção do ajuste automático

Após a análise do ajuste sazonal automático, verificamos que o modelo precisava ser corrigido. O IBGE, utilizando o método X-12-ARIMA, adiciona ao modelo de ajuste sazonal, além de efeitos de

trading days e Páscoa, o efeito do Carnaval (IBGE, 2015c). Esse efeito também será acrescentado e esperamos que as alterações corrijam a normalidade dos resíduos e o novo modelo tenha um critério de informação inferior ao do modelo automático.

Para criar a variável de Carnaval, vamos utilizar a função `genhol()` do **seasonal** que apresenta os seguintes argumentos:

genhol(x, start = 0, end = 0, frequency = 12, center = "none")

- `x`: um vetor da classe `Date`, contendo as datas de ocorrência do feriado;
- `start`: inteiro, desloca o ponto inicial do feriado. Use valores negativos se o efeito começa antes da data específica;
- `end`: inteiro, desloca o ponto final do feriado. Use valores negativos se o efeito termina antes da data específica;
- `frequency`: inteiro, frequência da série temporal resultante.

Para os argumentos `start` e `end` escolhemos colocar uma janela de 3 dias antes e um dia após o feriado, uma vez que no Brasil o feriado dura quase uma semana, mas o leitor pode se sentir livre para alterar esses argumentos. Ao argumento `frequency` foi atribuído 12 pois os dados são mensais.

```
> dates <- c("02/12/2002","03/04/2003","02/24/2004","02/08/2005",
+            "02/28/2006","02/20/2007","02/05/2008","02/24/2009",
+            "02/16/2010","03/08/2011","02/21/2012","02/12/2013",
+            "03/04/2014","02/17/2015","02/09/2016","02/28/2017")
> carnaval.date <- as.Date(dates,  "%m/%d/%Y")
> carnaval <- genhol(carnaval.date, start = -3, end = 1, frequency = 12)
```

Para acrescentar a variável `carnaval` ao ajuste sazonal, precisamos especificar o argumento `xreg = carnaval`. Os *trading days* poderiam ser específicos para cada dia da semana, como foi feito no ajuste automático, ou apenas uma variável que combinasse essas informações específicas (ver nota de rodapé 5). Ambos os tipos foram testados com e sem o efeito do ano bissexto. E a opção que melhor[7] caracterizou o modelo foi apenas uma variável que indicasse efeitos do dia da semana mais o efeito de anos bissextos e esta é indicada por `td1coef`. É importante o leitor saber que a variável que representa os *trading days* é dada pelo programa e não é calculada considerando os feriados do calendário brasileiro.

```
> ajuste_novo <- seas(pim.ts, transform.function = "none",
+                     xreg = carnaval, regression.variables = "td1coef")
> summary(ajuste_novo)
```

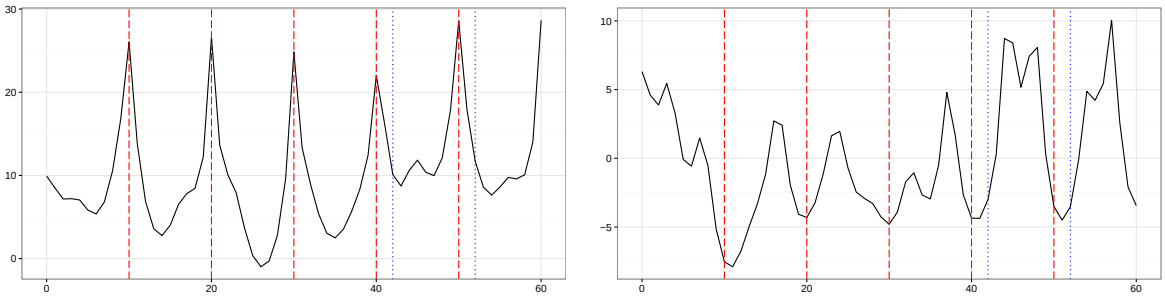
[7] Como melhor, consideramos os parâmetros significativos e a redução do critério de informação BIC.

```
Call:
seas(x = pim.ts, xreg = carnaval, transform.function = "none",
regression.variables = "td1coef")

Coefficients:
Estimate Std. Error z value Pr(>|z|)
carnaval           -3.01264    0.48992  -6.149 7.78e-10 ***
Leap Year           2.41786    0.73006   3.312 0.000927 ***
Weekday             0.35307    0.03052  11.570  < 2e-16 ***
Easter[1]          -2.98575    0.41402  -7.212 5.53e-13 ***
AO2008.Nov         -6.81580    1.49880  -4.548 5.43e-06 ***
LS2008.Dec        -17.10157    1.79419  -9.532  < 2e-16 ***
AO2011.Dec          5.06899    1.18449   4.279 1.87e-05 ***
AR-Nonseasonal-01  -0.28248    0.07944  -3.556 0.000377 ***
MA-Seasonal-12      0.52212    0.07772   6.718 1.85e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

SEATS adj.  ARIMA: (1 1 0)(0 1 1)  Obs.: 156  Transform: none
AICc: 582.5, BIC: 610.5  QS (no seasonality in final):0.3954
Box-Ljung (no autocorr.): 20.63   Shapiro (normality): 0.9914
```

Com as mudanças inseridas na função `seas()`, o modelo SARIMA também foi modificado de `(0 1 0)(0 1 1)` para `(1 1 0)(0 1 1)` com todos os parâmetros significativos com 95% de confiança. As variáveis de regressão também foram significativas considerando o mesmo nível de confiança e, assim, podemos concluir que a quantidade de dias da semana e o ano bissexto influenciam na produção industrial brasileira.

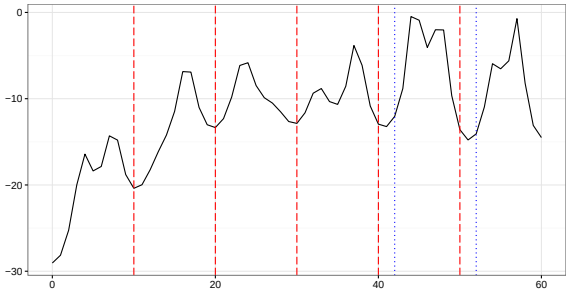
O leitor também deve ter reparado que as alterações nos permitiram concluir que os resíduos seguem distribuição normal, que os critérios de informação AICc e BIC são consideravelmente inferiores ao do ajuste automático e que o programa encontrou mais um outlier além dos relacionados a crise econômica de 2008 (`AO2011.Dec`). O teste de Ljung & Box não mostrou autocorrelação residual até o lag 24.

A análise do gráfico spectral (Figura 6.4), de forma diferente da análise feita no tópico 3, não mostra indícios de efeitos de *trading days* na série com ajuste sazonal. Os fatores sazonais (Figura 6.5) aparentam ter um comportamento mais suave do que o do ajuste automático. Repare que o mês de fevereiro foi melhor captado depois da correção. Note também o impacto dos *outliers* nos fatores sazonais de novembro de 2008 e de dezembro de 2011: a componente SI assume um valor discrepante comparado aos outros valores do mesmo mês.

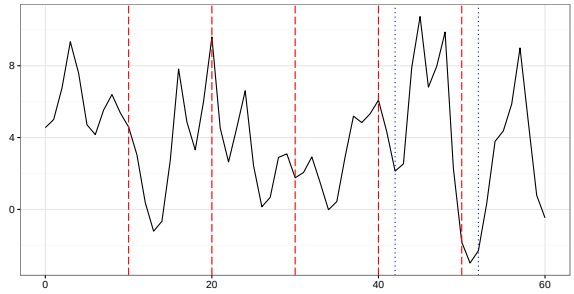


(a) Série original

(b) Série com ajuste sazonal



(c) Série da componente irregular



(d) Resíduos do modelo ARIMA

Figura 6.4: Análise spectral do ajuste sazonal corrigido.

A conclusão a respeito do teste de sazonalidade para o novo ajuste é semelhante à conclusão para o ajuste automático, também classificando este ajuste como adequado.

```
> qs(ajuste_novo)
              qs   p-val
qsori        162.39311 0.00000
qsorievadj   230.00377 0.00000
qsrsd          0.00000 1.00000
qssadj         0.39540 0.82062
qssadjevadj    0.00000 1.00000
qsirr          0.00000 1.00000
qsirrevadj     0.00000 1.00000
qssori        81.19590 0.00000
qssorievadj  130.82652 0.00000
qssrsd         0.00000 1.00000
qsssadj        0.00132 0.99934
qsssadjevadj   0.00000 1.00000
qssirr         0.00000 1.00000
qssirrevadj    0.00000 1.00000
```

Por fim, temos o gráfico do índice de produção industrial com ajuste sazonal pelo X-13ARIMA-SEATS em que o leitor pode notar o comportamento decrescente do índice nos últimos meses.

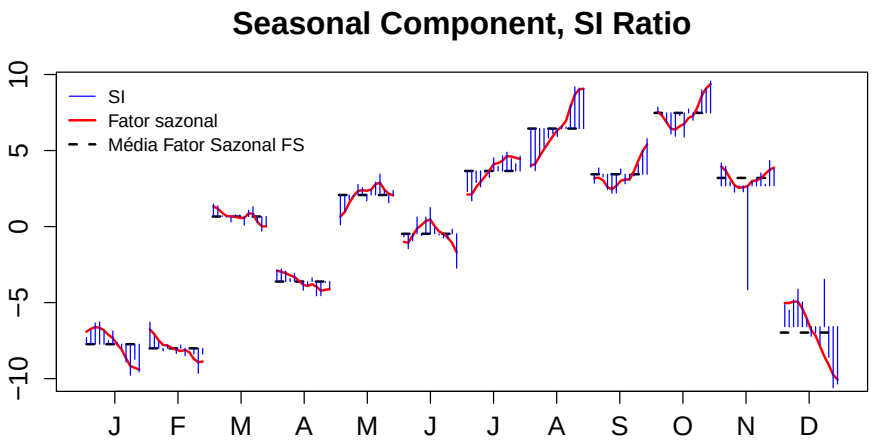


Figura 6.5: Fatores sazonais e SI *ratio* para o modelo corrigido

```
> plot(ajuste_novo, main = "")
> legend("topleft", legend = c("Observada", "Com ajuste sazonal"),
+        lty = 1, col = c(1,2), lwd = c(1,2), bty = "n")
```

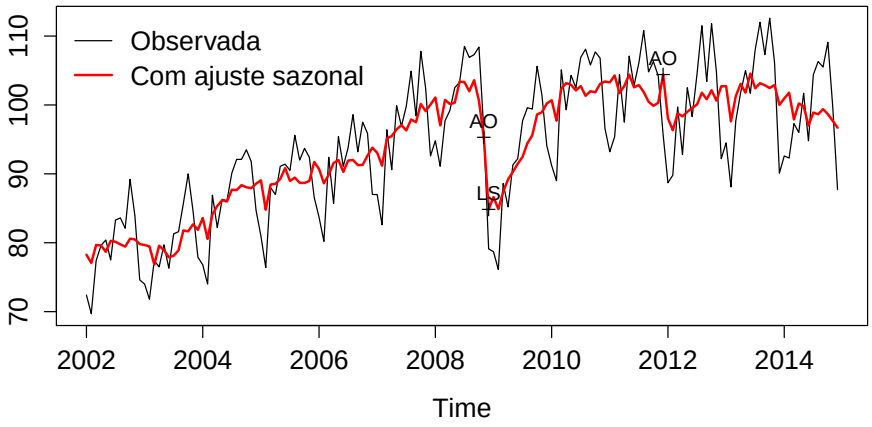


Figura 6.6: Índice de produção industrial geral do Brasil com ajuste sazonal

Considerações Finais

Neste capítulo aprendemos o que é o ajuste sazonal e a sua finalidade. Vimos que não há apenas uma maneira de dessazonalizar uma série temporal, embora tenhamos focado em apenas uma:

o programa de ajuste sazonal X-13ARIMA-SEATS do US Census Bureau. Aprendemos os passos necessários de como executar o X-13 no R e métodos de avaliar a qualidade do ajuste sazonal a partir de diversos diagnósticos. Além disso, vimos a utilidade da etapa de pré-ajuste no X-13, que permitiu a inserção de outras variáveis, aprimorando as avaliações do diagnóstico.

Apesar de ter sido uma experiência interessante, sabemos que ainda ficaram faltando alguns pontos a serem abordados. Por exemplo, o ajuste de diversas séries temporais simultaneamente, pois sabemos que há interesse em fazer isso para outras séries temporais além do índice de produção industrial. Também não abordamos as previsões da série com ajuste sazonal.

Nesse sentido, é importante que o leitor que estiver usando esse manual para dessazonalizar séries temporais tenha ciência de suas limitações e busque, sempre que possível, aprofundar o seu conhecimento sobre o assunto consultando outras fontes.

Parte III

Análise de Séries Temporais: Modelos Multivariados

Box & Jenkins com função de transferência

Daiane Marcolino de Mattos

Pedro Costa Ferreira

Introdução

Este capítulo é dedicado à apresentação dos modelos de Box & Jenkins com variáveis auxiliares. A utilização de variáveis auxiliares nesses modelos por meio de funções de transferência pode aperfeiçoar a modelagem e a previsão de séries temporais (ST).

O tema será exposto utilizando um exemplo clássico de Box & Jenkins (BJ) em que a produção de CO_2 (dióxido de carbono) é influenciada pela Taxa de Alimentação de Metano. O exemplo é aplicado de forma simples e objetiva no software R, não necessitando que o leitor entenda profundamente sobre as variáveis. No entanto, saiba que esse tipo de modelagem é baseada na relação causal entre as STs, portanto o leitor precisa ter conhecimento sobre o tema de interesse ao generalizar esse estudo para outras STs.

Para que você acompanhe o objetivo desse capítulo, este foi dividido em outras quatro seções: na seção 7.2 apresentamos a definição de Função de Transferência (FT); na seção 7.3 informamos os pacotes necessários para a modelagem no R e também as variáveis que utilizaremos; na seção 7.4 o leitor aprenderá sobre a metodologia; e na seção 7.5 discutimos algumas considerações finais.

Definição

Os modelos de Box & Jenkins (1970) podem incorporar variáveis auxiliares e a forma como essas variáveis auxiliares (X_{it}) influenciam a variável resposta (Y_t), isto é, como os movimentos dessas variáveis afetam o percurso da variável resposta, é dada por uma função de transferência $f(X_t)$:

$$Y_t = f(X_t) + \varepsilon_t \tag{7.1}$$

onde ε_t pode ser um ruído branco ou um modelo ARIMA completo.

A função $f(X_t)$ pode agrupar valores passados e/ou presentes de uma ou mais séries temporais, que podem ser do tipo quantitativo ou binário (*dummy*) e essa distinção implica na forma de identificação de $f(X_t)$. No caso de uma ST quantitativa, a forma genérica de uma $f(X_t)$ pode ser denotada pela equação 7.2:

$$f(X_t) = \frac{(w_0 + w_1 L + w_2 L^2 + ... + w_s L^s)}{(1 - \delta_1 L - \delta_2 L^2 - ... - \delta_r L^r)} X_{t-b} \tag{7.2}$$

Certamente o leitor notou que para identificar $f(X_t)$ é necessário descobrir os valores de r, s e b e estimar os parâmetros $w_i, i = 0, ..., s$ e $\delta_j, j = 1, ..., r$. Esclareceremos, na seção seguinte, como isso pode ser feito aplicando a metodologia a um exemplo clássico extraído de Box & Jenkins (1970).

Dados e pacotes necessários

Para a modelagem de Box & Jenkins com função de transferência (BJFT) no R, os seguintes pacotes devem ser instalados:

- **devtools** (usado na extração dos dados);
- **forecast** (usado na estimação e previsão de modelos ARIMA);
- **TSA** (usado na estimação de modelos ARIMA com função de transferência);
- **tseries** (usado para executar teste de normalidade em séries temporais);
- **FinTS** (usado para executar testes de heterocedasticidade em séries temporais).

É importante o leitor saber que outros pacotes também serão utilizados na modelagem BJFT. No entanto, não há necessidade de instalá-los pois já estão incluídos na versão base do R, como exemplo podemos citar o pacote **stats** que é utilizado para calcular funções de autocorrelação e extrair resíduos de modelos, e o pacote **graphics** usado para elaborar gráficos.

Para identificar e estimar a função de transferência $f(X_t)$, será utilizado um exemplo clássico extraído de Box & Jenkins (1970). Tal exemplo investiga a otimização adaptativa de um aquecedor a gás, isto é, foi utilizado uma combinação de ar e metano para formar uma mistura de gases contendo CO_2 (dióxido de carbono). A alimentação de ar foi mantida constante, mas a taxa de alimentação de metano poderia ser variada de qualquer maneira desejada. Após a combinação, a concentração de CO_2 resultante nos gases de exaustão foi medida. A finalidade do exemplo é encontrar a forma como a variável Taxa de Alimentação de Metano (X_t) se relaciona com Concentração de CO_2 (Y_t). Cada variável representa uma série temporal de 296 observações e os dados podem ser baixados diretamente pelo R executando as linhas de comando a seguir.

```
> devtools::source_url("http://git.io/vCXJC")
> gas
```

As colunas do objeto `gas` referem-se à variável independente X_t e à variável dependente Y_t, nessa ordem. Observamos na Figura 7.1 o comportamento de ambas as séries em análise. Como visto no capítulo sobre a modelagem BJ, as séries não apresentam um comportamento estacionário em todo o

espaço de tempo, com variações não constantes e uma leve tendência[1] de queda para X_t e crescimento para Y_t.

```
> plot(gas, main = "")
```

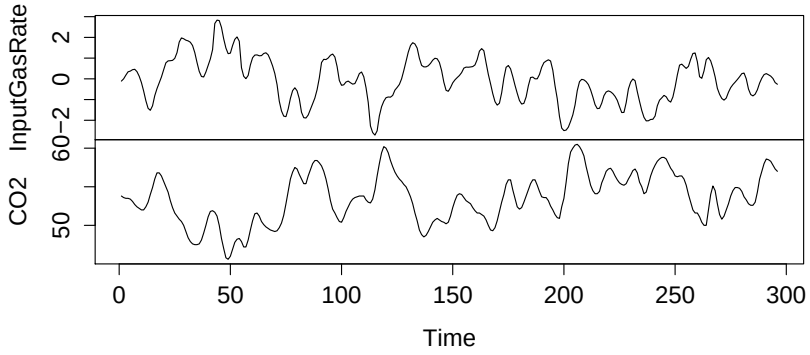


Figura 7.1: Input Gas Rate (X) e CO_2 (Y)

Metodologia

Agora que o leitor já foi apresentado às STs, apresentamos os passos de identificação da $f(X_t)$, que segundo a metodologia BJFT são:

1. Calcular a função de correlação cruzada entre Y_t e X_t;
2. Identificar r, s e b;
3. Estimar o modelo com função de transferência;
4. Verificar se o modelo é adequado.

Discutiremos cada etapa nas seções seguintes.

Função de correlação cruzada entre Y e X

Como vimos na definição de FT, para identificá-las, inicialmente, basta estipularmos valores para r, s e b (lembre-se que estamos trabalhando apenas com séries quantitativas). A identificação desses valores é feita calculando a função de correlação cruzada (CCF) entre Y_t e as variáveis auxiliares

[1] Lembre-se que para confirmar essa afirmação o mais correto é fazer um teste de raiz unitária.

(apenas uma neste exemplo). A CCF entre as séries temporais Y_t e X_t mostra as correlações entre elas para diferentes defasagens no tempo, sendo definida na equação 7.3 como:

$$ccf(k) = \frac{c_{xy}(k)}{s_x s_y}, \quad k = 0, \pm 1, \pm 2, ... \tag{7.3}$$

onde s_x e s_y representam, respectivamente, os desvios-padrão de X_t e Y_t; $c_{xy}(k)$ representa a covariância entre as duas variáveis no lag k:

$$c_{xy}(k) = \begin{cases} \frac{1}{n}\sum_{t=1}^{n-k}(x_t - \bar{x})(y_{t+k} - \bar{y}), & k = 0, 1, 2, ... \\ \frac{1}{n}\sum_{t=1}^{n+k}(y_t - \bar{y})(x_{t-k} - \bar{x}), & k = 0, -1, -2, ... \end{cases}$$

Para $k > 0$, a CCF mostra o relacionamento entre X no tempo t e Y no tempo futuro $t+k$. Em contrapartida, para valores negativos de k, tem-se o relacionamento entre X no tempo t e Y no tempo passado $t-k$.

É importante saber que a CCF é afetada pela autocorrelação de X_t e Y_t e, se as STs não forem estacionárias, o resultado da CCF não refletirá realmente o grau de associação entre elas Hamilton (1994); Phillips & Perron (1988). Para corrigir esse problema, Box & Jenkins sugeriram o método de pré-branqueamento[2].

O pré-branqueamento permite eliminar a estrutura de tendência (determinística ou estocástica) presente numa série temporal. O método consiste nas seguintes etapas:

(a) Ajustar um modelo ARIMA para a série independente X_t;

(b) Filtrar Y_t pelo modelo encontrado em (a), isto é, o modelo de Y_t é o mesmo modelo de X_t (com os mesmos parâmetros estimados);

(c) Salvar os resíduos dos dois modelos;

(d) Calcular a CCF entre os resíduos obtidos em (c).

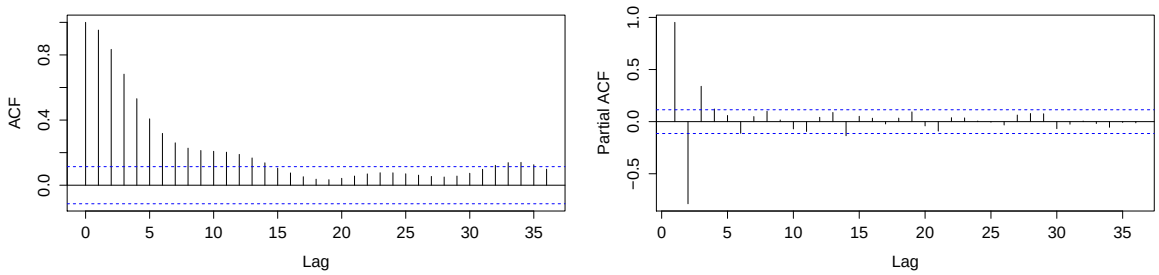
A seguir discute-se cada etapa separadamente aplicando-as às STs disponíveis por BJ.

[2]Existe também o pré-branqueamento duplo, em que é ajustado para cada variável seu próprio modelo ARIMA. No entanto, por tornar as duas séries um ruído branco, a correlação entre elas pode ser puramente devida ao acaso.

(a) *Ajustar um modelo ARIMA para a série independente* X_t:

Já aprendemos a identificar as ordens de um modelo ARIMA para X_t usando a função de autocorrelação (FAC) e a função de autocorrelação parcial (FACP)[3].

```
> acf(gas[,"InputGasRate"], lag.max = 36)
> pacf(gas[,"InputGasRate"], lag.max = 36)
```



(a) FAC: Input Gas Rate (X)

(b) FACP: Input Gas Rate (X)

Figura 7.2: FAC e FACP: Input Gas Rate (X)

A queda exponencial da FAC e o corte brusco da FACP no lag 3 (Figura 7.2) sugerem um modelo ARIMA(3,0,0). Ao ajustar o modelo ARIMA (função `Arima()` do pacote **forecast**), a constante não foi significativa, sendo esta excluída do modelo. Veja os códigos a seguir para a estimação do modelo.

```
> library("forecast")
> (modelo_x <- Arima(gas[,"InputGasRate"],
+                    order = c(3,0,0), include.mean = F))
Series: gas[, "InputGasRate"]
ARIMA(3,0,0) with zero mean

Coefficients:
ar1      ar2     ar3
1.9696  -1.3659  0.3399
s.e.  0.0544   0.0985  0.0543

sigma^2 estimated as 0.03531:  log likelihood=72.52
AIC=-137.04   AICc=-136.9   BIC=-122.27
```

[3]Um método que auxilia na identificação de modelos ARIMA são os critérios de informação, como por exemplo AIC Akaike (1973) e BIC Schwarz et al. (1978).

(b) *Filtrar Y_t pelo modelo encontrado em (a):*

Para filtrar Y_t por meio do modelo de X_t também usaremos a função `Arima()`, no entanto, agora acrescentaremos o argumento `model` indicando o modelo já estimado anteriormente.

```
> (modelo_y <- Arima(gas[,"CO2"], model = modelo_x))
Series: gas[, "CO2"]
ARIMA(3,0,0) with zero mean

Coefficients:
ar1      ar2     ar3
1.9696  -1.3659  0.3399
s.e.  0.0000   0.0000  0.0000

sigma^2 estimated as 9.56:  log likelihood=-756.47
AIC=1514.94   AICc=1514.96   BIC=1518.63
```

(c) *Salvar os resíduos dos dois modelos:*

Os resíduos podem ser obtidos pela função `resid()` e serão salvos em novos objetos `alpha` e `beta` para os modelos de X e Y, respectivamente.

```
> alpha <- resid(modelo_x)
> beta <- resid(modelo_y)
```

(d) *Calcular a CCF entre os resíduos obtidos em (c):*

Uma vez que as variáveis foram filtradas, podemos calcular a CCF (Figura 7.3) entre os resíduos utilizando a função `ccf()` do pacote **stats**. A CCF nessa figura mostra o relacionamento entre Y_t e os lags defasados de X_t a partir dos coeficientes de correlação.

Veja que não há correlação significativa entre Y_t e X_t no tempo presente (t = 0) e a primeira correlação significativa é dada para t = 3, ou seja, entre Y no tempo presente e X defasada em 3 lags.

```
> ccf(beta, alpha, xlim = c(0,20))
```

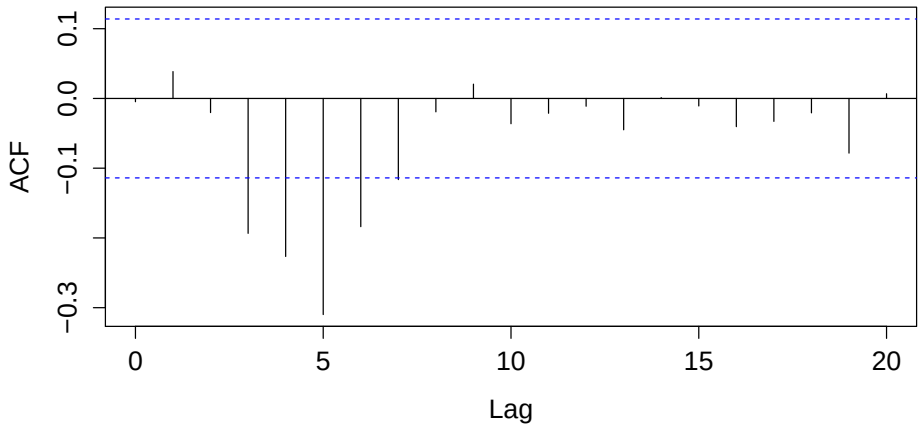


Figura 7.3: CCF: Input Gas Rate (X) and CO_2 (Y)

Identificar r, s e b

Com a CCF estimada, pode-se descobrir as ordens r, s e b:

- b: refere-se ao primeiro lag significativo. Representa a primeira defasagem de X que entrará no modelo. Neste caso, $b = 3$.
- s: número de *lags* crescentes depois de b. Representa as próximas defasagens de X que entrarão no modelo. Logo, $s = 2$.
- r: por haver queda exponencial[4] após os lags crescentes, $r = 1$.

Portanto, o modelo contém X_{t-3}, X_{t-4} e X_{t-5} e a $f(X_t)$ é definida como

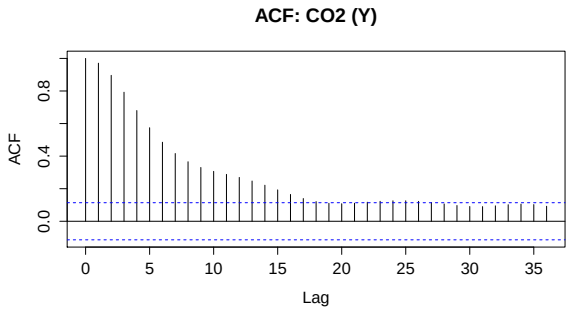
$$f(X_t) = \frac{(w_0 + w_1 L + w_2 L^2)}{(1 - \delta_1 L)} X_{t-3}$$

Estimação do modelo com função de transferência

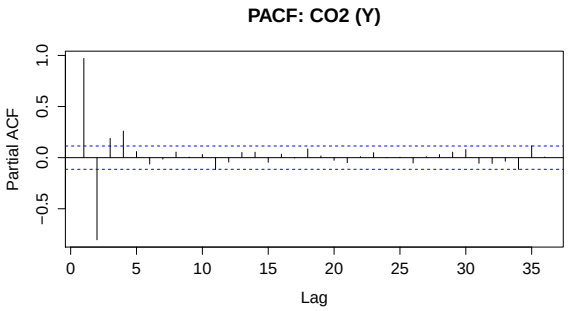
Uma vez que identificamos a forma de $f(X_t)$, passaremos para a etapa de estimação. Inicialmente, é preciso identificar a ordem do modelo ARIMA para a série Y_t, assim como foi feito para X_t.

```
> acf(gas[,"CO2"], lag.max = 36, main = "FAC: CO2 (Y)")
> pacf(gas[,"CO2"], lag.max = 36, main = "FACP: CO2 (Y)")
```

[4]Queda exponencial na CCF indica parâmetros no denominador (r), enquanto picos indicam parâmetros no numerador (s). Veja um exemplo em Box & Jenkins (1970) (figura 10.6).



(a) FAC: CO_2 (Y)



(b) FACP: CO_2 (Y)

Figura 7.4: FAC e FACP: CO_2 (Y)

A queda exponencial da FAC e o corte brusco da FACP no lag 2 (Figura 7.4) sugerem um modelo ARIMA(2,0,0).

Finalmente, estimaremos o modelo com função de transferência utilizando a função `arimax()` do pacote **TSA**.

Sintaxe: arimax(x, order = c(0, 0, 0), seasonal = list(order = c(0, 0, 0), period = NA), xreg = NULL, include.mean = TRUE, transform.pars = TRUE, fixed = NULL, init = NULL, method = c("CSS-ML", "ML", "CSS"), n.cond, optim.control = list(), kappa = 1e+06, io = NULL, xtransf, transfer = NULL)

Os argumentos que utilizaremos são:

- `x`: série dependente (Y_t);
- `order`: ordem do modelo ARIMA para Y_t;
- `xtransf`: série independente (X_t) já defasada no lag b;
- `transfer`: valores de r e s na forma `list(c(r,s))`.

Para defasar a variável X_t em três lags, o leitor pode usar a função `lag()` do pacote **stats**.

```
> x_novo <- lag(gas[,"InputGasRate"], k = -3)
```

Como três valores de X_t foram "perdidos" para estimar o modelo, é preciso cortar os três primeiros valores de Y_t para os dois conjuntos de dados terem o mesmo tamanho.

```
> gas_novo <- na.omit(cbind(x_novo, gas[,"CO2"]))
> colnames(gas_novo) <- c("InputGasRate", "CO2")
> head(gas_novo)
```

```
InputGasRate  CO2
      -0.109 53.5
       0.000 53.4
```

```
0.178 53.1
0.339 52.7
0.373 52.4
0.441 52.2
```

Com os dados na forma correta, estima-se um modelo para a variável dependente CO_2.

```
> (modelo_ft <- arimax(x = gas_novo["CO2"], order = c(2,0,0),
+                      xtransf = gas_novo["InputGasRate"],
+                      transfer = list(c(1,2))) )

Call:
arimax(x = gas_novo[, "CO2"], order = c(2, 0, 0), xtransf = gas_novo[, "InputGasRate"],
transfer = list(c(1, 2)))

Coefficients:
        ar1      ar2  intercept  T1-AR1   T1-MA0   T1-MA1   T1-MA2
     1.5272  -0.6288    53.3618  0.5490  -0.5310  -0.3801  -0.5180
s.e. 0.0467   0.0495     0.1375  0.0392   0.0738   0.1017   0.1086

sigma^2 estimated as 0.0571:  log likelihood = 2.08,  aic = 9.83
```

O modelo apresentado na saída do R pode ser representado pela seguinte equação:

$$Y_t = 53.4 + \frac{(-0.5310 - 0.3801L - 0.5180L^2)}{(1 - 0.5490L)} X_{t-3} + \frac{1}{1 - 1.5272L + 0.6288L^2} e_t$$

Uma vez estimado, o modelo sempre precisa ser avaliado. Se ao final verificarmos que o modelo não é adequado, então sua identificação foi incorreta e precisaremos corrigir essa etapa. Veremos na seção seguinte como avaliar a adequação do modelo BJFT.

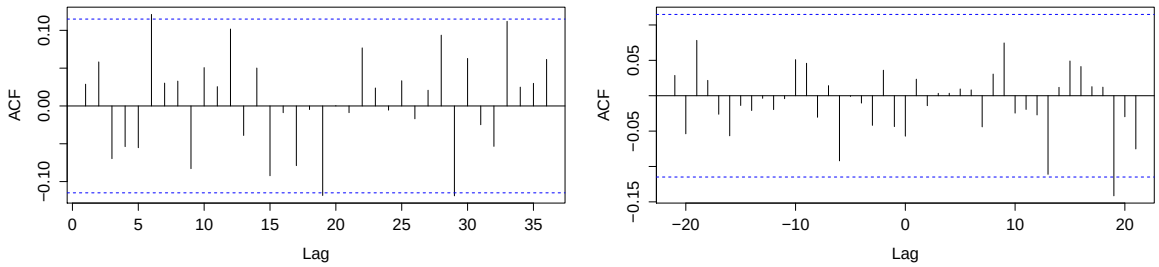
Verificar se o modelo é adequado

Para avaliar se o modelo ajustado é adequado, vamos executar as seguintes análises:

- Calcular autocorrelação dos resíduos;
- CCF entre os resíduos e a variável auxiliar X_t pré-branqueada.

Pretendemos não encontrar padrões de correlação, pois isso sugere que o modelo não esteja bem especificado e que, consequentemente, deve ser modificado. A seguir temos a FAC dos resíduos e o teste de autocorrelação de **?** utilizando a função `Box.test()` do pacote **stats**.

```
> residuos <- resid(modelo_ft)
> acf(residuos, na.action = na.omit, lag.max = 36)
> ccf(residuos, alpha, na.action = na.omit)
> Box.test(residuos, type = "Ljung-Box", lag = 24)

Box-Ljung test

data:  residuos
X-squared = 27.969, df = 24, p-value = 0.2613
```



(a) FAC: Resíduos

(b) CCF: Resíduos vs. Alpha

Figura 7.5: ACF Resíduos e CCF entre Resíduos e Alpha

A partir da análise das funções de correlação (Figura 7.5) e do teste de autocorrelação residual de Ljung-Box, podemos concluir que não há evidências de autocorrelação residual e, portanto, esse modelo BJFT está adequado. O gráfico dos valores observados versus ajustados pelos modelos com e sem função de transferência é exposto na Figura 7.6.

```
> modelo_y <- Arima(gas[,"CO2"], order = c(2,0,0), include.mean = T)
> ajustados <- fitted(modelo_y)
> ajustados_ft <- fitted(modelo_ft)
> ts.plot(gas[,"CO2"], ajustados, ajustados_ft, lty = c(1,3,2),
+         lwd = c(1,3,2), col = c(1, "orangered","dodgerblue"))
> legend("bottomright", col = c(1,"dodgerblue", "orangered"),
+        legend = c("Observados", "Ajustados c/ FT", "Ajustados s/ FT"),
+        lty = c(1,2,3), lwd = c(1,2,2), cex = 0.7)
>
```

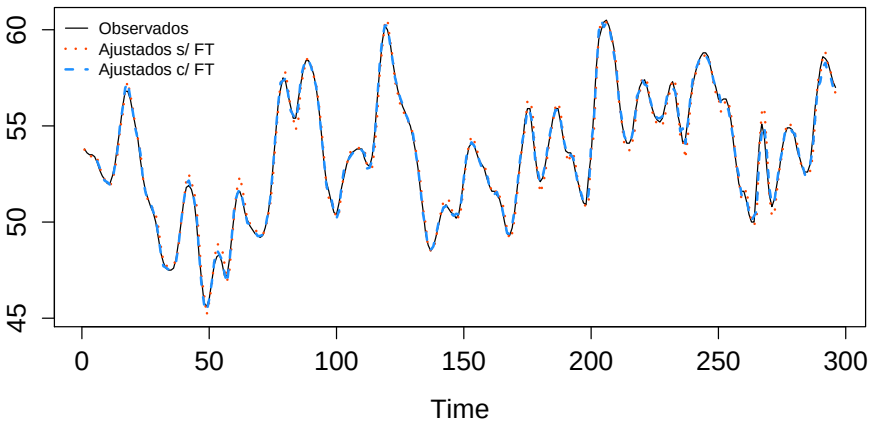


Figura 7.6: TS observada (CO_2) e valores ajustados com e sem FT

A partir da análise dos modelos com e sem FT, vemos que ambos são adequados para a modelagem de CO_2 dentro da amostra. Se você aplicar a função `summary()` para os dois modelos, verá que o MAPE (*Mean Absolute Percent Error*) para o modelo com função de transferência é de 0,3% e para o outro modelo é de 0,5%, confirmando o melhor desempenho do modelo BJFT dentro da amostra. Outra forma de verificar qual dos dois modelos é mais eficiente é analisar suas respectivas previsões para fora da amostra. No entanto, a função `arimax()` não suportava previsões até o momento de edição desse livro.

Considerações finais

Neste capítulo aprendemos empiricamente, com base na metodologia proposta por Box & Jenkins, como modelar uma série temporal utilizando outra variável que possui um relacionamento causal com a variável de interesse. Aprendemos os passos para aplicar a metodologia, bem como filtrar as STs utilizadas para que as etapas possam ser aplicadas corretamente.

Foram abordados os pacotes úteis para esse tipo de modelagem, discutimos algumas funções e chamamos a atenção para algumas limitações das mesmas. Apesar de ter sido uma experiência interessante, sabemos que ainda ficaram faltando alguns pontos a serem abordados, como, por exemplo, a previsão das séries para fora da amostra e a identificação de variáveis do tipo binário na função de transferência.

Nesse sentido, é importante que o leitor que estiver usando esse manual para construir o seu modelo ARIMA com função de transferência tenha ciência de suas limitações e busque, sempre que possível, aprofundar o seu conhecimento sobre o assunto.

Regressão Dinâmica

Ingrid Christyne Luquett de Oliveira

Pedro Costa Ferreira

Introdução

Modelos dinâmicos estudam a relação entre variáveis observadas em instantes de tempo diferentes. Podemos, por exemplo, investigar se o Índice de Preços ao Consumidor Amplo (IPCA)[1] em um determinado mês influencia a maneira como os consumidores brasileiros formam suas expectativas de inflação em meses subsequentes. Sob esta ótica, pretendemos neste capítulo explorar as implicações do emprego do modelo clássico de regressão linear em variáveis observadas ao longo do tempo e apresentar a metodologia de regressão dinâmica como alternativa ao uso dos modelos usuais.

As seções que seguem dividem-se da seguinte forma: na Seção 8.2 descrevemos o modelo clássico de regressão linear, seus pressupostos e as consequências em violá-los; posteriormente, a Seção 8.3 aborda especificamente a presença de correlação serial nos erros do modelo clássico de regressão e expõe maneiras para contornar o problema; sob outra perspectiva, na Seção 8.4 tratamos as violações dos pressupostos como uma especificação inadequada do modelo e apresentamos os modelos autoregressivos com defasagens distribuídas; explorando o contexto de variáveis não-estacionárias, a Seção 8.5 discute o modelo de correção de erro; a Seção 8.6 apresenta uma aplicação dos modelos na análise da formação da expectativa de inflação por parte dos consumidores brasileiros baseada no IPCA; por fim, a Seção 8.7 resume todos os modelos apresentados.

Modelo clássico de regressão linear

Em diferentes contextos estamos interessados em estudar se o comportamento de uma determinada variável (dependente) é influenciado por uma ou mais variáveis (explicativas). A estrutura desta relação pode assumir diferentes formas e, em alguns casos, apresenta comportamento linear. Os modelos que assumem estrutura linear entre variável dependente e variáveis explicativas são chamados modelos de regressão linear ou modelos lineares.

Considere a variável dependente Y_t, observada ao longo do tempo, e k variáveis explicativas $\{X_{1,t}, X_{2,t}, \ldots X_{k,t}\}$. O modelo linear usualmente encontrado na literatura pode ser escrito como

$$Y_t = \beta_0 + \beta_1 X_{1,t} + \beta_2 X_{2,t} + \cdots + \beta_k X_{k,t} + \varepsilon_t, \tag{8.1}$$

onde β_0 é um nível global, os β_j's, $j \in \{1, 2, \ldots, k\}$, são os parâmetros correspondentes aos respectivos

[1] Divulgado pelo Instituto Brasileiro de Geografia e Estatística (IBGE).

efeitos isolados de cada $X_{j,t}$ sobre Y_t, e ε_t é o erro do modelo no tempo t. Ao longo do capítulo a equação (8.1) será evetualmente referida como modelo estático.

A construção dos modelos de regressão linear é fundamentada na aceitação dos seguintes pressupostos sobre o erro ε_t:

1. **Exogeneidade estrita:** as variáveis explicativas X_k são estritamente exógenas com respeito ao termo de erro ε_t de maneira que

$$E(\varepsilon_t \mid X) = 0, \;\; t = 1,2,\ldots,T \tag{8.2}$$

   onde X inclui todas as k variáveis explicativas e todos os instantes de tempo t:

$$X = \begin{bmatrix} X_{1,1} & X_{2,1} & \cdots & X_{k-1,1} & X_{k,1} \\ X_{1,2} & X_{2,2} & \cdots & X_{k-1,2} & X_{k,2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ X_{1,T} & X_{2,T} & \cdots & X_{k-1,T} & X_{k,T} \end{bmatrix}.$$

2. **Ausência de colinearidade perfeita:** nenhuma variável explicativa $X_{k,t}$ é constante ou pode ser expressa como uma função linear de outras covariáveis.
3. **Homocedasticidade:** a variância do erro é a mesma para todas as observações, ou seja, $Var(\varepsilon_t \mid X) = \sigma^2$, $t = 1,2,\ldots,T$.
4. **Ausência de correlação serial:** os termos de erro são independentes (condicionalmente a X), ou seja, $Cov(\varepsilon_t, \varepsilon_{t-s} \mid X) = 0$, $s = 1,2,\ldots,T-1$.
5. **Normalidade:** os ε_t's são identicamente distribuídos com $\varepsilon_t \sim N(0,\sigma^2)$.

Se as duas primeiras condições forem satisfeitas, o estimador de mínimos quadrados ordinários (MQO) será não viesado. Ainda, caso vigorem as suposições 3 e 4 o estimador de MQO da variância do estimador dos coeficientes do modelo também será não viesado. Caso os pressupostos de 1 a 4 sejam satisfeitos, o estimador de MQO será o melhor estimador linear não viesado (em inglês, *BLUE*). Por fim, se o pressuposto 5 for observado, tem-se que os β_k's seguem distribuição gaussiana e a razão entre cada coeficiente e seu erro padrão segue distribuição t-Student.

Os pressupostos expostos acima são razoáveis em contextos onde as observações são independentes. Na análise de séries temporais, entretanto, algumas suposições frequentemente não são satisfeitas. Em particular, a suposição de que as variáveis explicativas são independentes de toda a história de Y

(exogeneidade estrita) e que choques em Y no período t não persistem em $t+1$ (correlação serial) são usualmente violadas.

O emprego de modelos estáticos em séries temporais requer, portanto, que sejamos capazes de lidar com a violação dos pressupostos 1 e 4. A suposição de exogeneidade estrita pode ser relaxada em situações onde as variáveis em análise são ergódicas, sendo necessário somente independência contemporânea entre os erros e X (exogeneidade fraca), ou seja, $E(\varepsilon_t \mid X_{1,t}, X_{2,t}, \ldots, X_{k,t}) = 0$. Se, além disso, a amostra for grande o suficiente, o estimador de mínimos quadrados terá as propriedades assintóticas desejadas.

Caso as suposições 1-3 sejam satisfeitas, mesmo na presença de correlação serial no vetor de erros $\varepsilon = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_T\}$, os estimadores de MQO serão consistentes. Entretanto, o mesmo não ocorre com respeito à eficiência, ou seja, existe algum estimador cuja variância é menor do que a variância do estimador de MQO (Pindyck & Rubinfeld, 1998). Como consequência, a inferência acerca dos coeficientes do modelo pode conduzir a conclusões incorretas uma vez que as distribuições usadas para testar as hipóteses sobre os coeficientes não estarão corretas.

Correlação serial

Vimos que a presença de correlação serial nos erros afeta a inferência do modelo, impossibilitando a realização dos testes usuais sobre os parâmetros. Devemos, portanto, verificar se os pressupostos do modelo estático são satisfeitos antes de tirar qualquer conclusão. A Subseção 8.3.1 aborda metodologias para diagnosticar a presença de correlação serial enquanto a Subseção 8.3.2 apresenta caminhos para a correção do problema.

Testes de correlação serial

A violação do pressuposto 4 ocorre quando $Cov(\varepsilon_t, \varepsilon_{t-s}) \neq 0$ para algum $s \neq 0$. Para testar a hipótese nula de ausência de correlação serial de ordem s nos erros ($H_0 : Cov(\varepsilon_t, \varepsilon_{t-s}) = 0$) é necessário que os estimadores de ε_t sejam consistentes. A escolha natural é utilizar os resíduos $\hat{\varepsilon}_t$ do modelo (8.1), estimado via MQO, de forma que os testes envolverão a análise da correlação entre $\hat{\varepsilon}_t$ e $\hat{\varepsilon}_{t-s}$ para s positivo até algum valor máximo arbitrado p. A literatura dispõe de ínumeros testes de correlação serial, todavia focaremos na descrição de dois deles: (i) Durbin-Watson[2] e (ii) Breusch-Godfrey[3].

[2] Durbin & Watson (1950, 1951, 1971)
[3] Breusch (1978), Godfrey (1978)

Teste de Durbin-Watson

A estatística de Durbin-Watson é dada por

$$d = \frac{\sum_{t=2}^{T} (\hat{\varepsilon}_t - \hat{\varepsilon}_{t-1})^2}{\sum_{t=1}^{T} \hat{\varepsilon}_t^2}, \tag{8.3}$$

onde $\hat{\varepsilon}_t$, $t = 1, \ldots, T$ são os resíduos da estimação de (8.1) por mínimos quadrados ordinários.

Após algumas aproximações, tem-se que $d \approx 2(1 - \hat{\rho})$, onde $\hat{\rho}$ é a correlação de primeira ordem de $\hat{\varepsilon} = \{\hat{\varepsilon}_1, \ldots, \hat{\varepsilon}_T\}$. Como $-1 \leq \hat{\rho} \leq 1$, a estatística d pertence ao intervalo entre 0 e 4, sendo a ausência de correlação serial correspondente a d próximo a 2.

Dados os limites inferior e superior da região de rejeição, d_L e d_U respectivamente, podemos concluir sobre a correlação serial segundo a Figura 8.1.

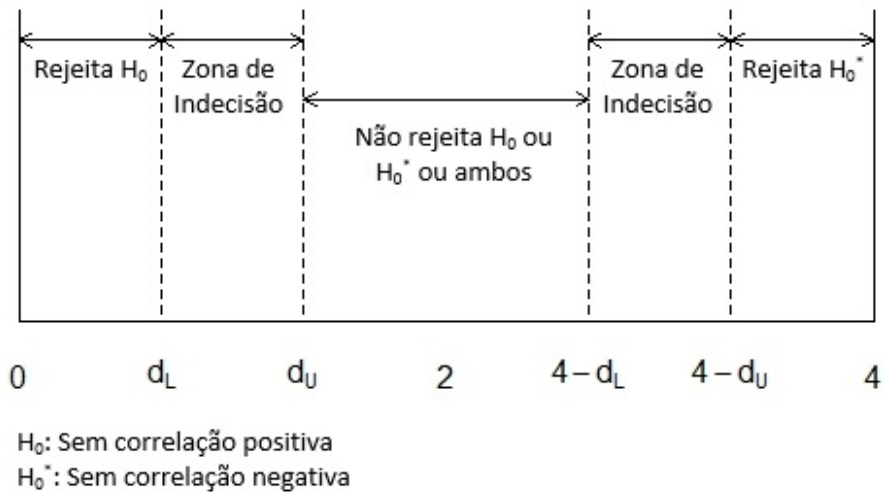


Figura 8.1: Regiões do teste de Durbin-Watson

A aplicação do teste de Durbin-Watson apresenta, entretanto, algumas limitações: *(i)* os valores críticos d_L e d_U dependem das covariáveis do modelo e não podem ser obtidos para o caso geral; *(ii)* o teste é inválido na presença da variável dependente defasada no lado direito da equação (8.1)[4]; e *(iii)* o teste somente aborda correlação de primeira ordem, não sendo aplicável a correlações de maior ordem.

Existem no R diferentes maneiras de obter a estatística do teste de Durbin-Watson, entre as quais podemos citar: `dwtest()` no pacote **lmtest** (Zeileis & Hothorn, 2002), `durbinWatsonTest()` no

[4] Durbin (1970, p. 200) propõe uma modificação do teste de Durbin-Watson que permite a inclusão da variável dependente defasada no modelo.

pacote **car** (Fox & Weisberg, 2011), `test.DW()` no pacote **dcv** (Li, 2010), etc.

Teste de Breusch-Godfrey ou Teste LM

O segundo teste de correlação serial é o teste de Breusch-Godfrey, que permite a inclusão de variáveis dependentes defasadas no modelo além de poder ser usado para testar correlações de ordem p, $p \geq 1$.

A hipóteste nula do teste é de que os erros são ruídos brancos. Se de fato essa hipótese for verdadeira, então os resíduos $\hat{\varepsilon}_t$ obtidos da estimação da equação (8.1) via MQO são independentes dos resíduos defasados $\hat{\varepsilon}_{t-1}$, ..., $\hat{\varepsilon}_{t-p}$. Deste modo, para avaliar a presença de correlação serial de ordem p o teste baseia-se no modelo

$$\hat{\varepsilon}_t = \gamma_1 \hat{\varepsilon}_{t-1} + \cdots + \gamma_p \hat{\varepsilon}_{t-p} + \beta_0 + \beta_1 X_{1,t} + \beta_2 X_{2,t} + \cdots + \beta_k X_{k,t} + \nu_t, \tag{8.4}$$

onde $\hat{\varepsilon} = \{\hat{\varepsilon}_1, \ldots, \hat{\varepsilon}_T\}$ são os resíduos da equação (8.1) e ν_t é o erro do modelo.

A estatística de teste é calculada como

$$BG(p) = (T-p)R^2, \tag{8.5}$$

onde R^2 é o coeficiente de determinação do modelo (8.4). A ausência de correlação serial e o baixo poder explicativo de $\hat{\varepsilon}$ pelas variáveis independentes implicam em valores pequenos do coeficiente R^2 e, consequentemente, da estatística $BG(p)$, levando à não rejeição da hipótese nula de que os resíduos são ruídos brancos.

Computacionalmente, a estatística do teste de Breusch-Godfrey pode ser obtida no R pela função `bgtest()` do pacote **lmtest** (Zeileis & Hothorn, 2002).

Correção da correlação serial

Identificada a presença de correlação serial nos erros através dos métodos apresentados na seção anterior, precisamos encontrar alternativas ao modelo estático para lidar com tal problema. Nos limitaremos, nesta seção, à descrição de duas abordagens: *(i)* inclusão de estrutura para os termos de erro no modelo (8.1); e *(ii)* aplicação do método de Newey-West para correção dos erros padrão das estimativas de MQO.

Assuma que as suposições do modelo de regressão linear são satisfeitas, porém os erros não são

independentes ao longo do tempo. Supondo um processo autoregressivo de primeira ordem, denotado por AR(1), para descrever o comportamento dos erros de regressão, temos o modelo

$$Y_t = \beta_0 + \beta_1 X_{1,t} + \beta_2 X_{2,t} + \cdots + \beta_k X_{k,t} + \varepsilon_t \tag{8.6a}$$

$$\varepsilon_t = \rho \varepsilon_{t-1} + \nu_t, \tag{8.6b}$$

onde $0 \leq \mid \rho \mid < 1$, ν_t tem distribuição $N(0, \sigma_\nu^2)$ e é independente de ν_j, para $j \neq t$ bem como é independente de ε_t, $\forall t$. De modo similar, $\varepsilon_t \sim N(0, \sigma_\varepsilon^2)$ porém os termos de erro são correlacionados ao longo do tempo. O termo ρ reflete a correlação entre ε_t e ε_{t-1} de modo que $\rho = 0$ implica ausência de autocorrelação nos erros e $\mid \rho \mid$ próximo a um resulta em correlação serial de primeira ordem. Cabe destacar que, pela construção do modelo em (8.6a)-(8.6b), o efeito de ε_t será sentido em todos os períodos posteriores, com magnitude descrescente ao longo do tempo.

Podemos reescrever as equações (8.6a)-(8.6b) em uma única equação. Para tal, multiplicamos o termo $(1 - \rho L)$ em todos os termos da equação (8.6a), obtendo-se 8.7a

$$Y_t^* = \beta_1(1-\rho) + \beta_1 X_{1,t}^* + \beta_2 X_{2,t}^* + \cdots + \beta_k X_{k,t}^* + \nu_t, \tag{8.7a}$$

onde L é o operador de defasagem tal que $LY_t = Y_{t-1}$ e

$$Y_t^* = Y_t - \rho Y_{t-1}, \tag{8.7b}$$

$$X_{j,t}^* = X_{j,t} - \rho X_{j,t-1} \tag{8.7c}$$

$$\nu_t = \varepsilon_t - \rho \varepsilon_{t-1}. \tag{8.7d}$$

Por construção, o modelo em (8.7a) possui erros independentes e identicamente distribuídos com média 0 e variância constante. Deste modo, se ρ for conhecido podemos aplicar o método de mínimos quadrados ordinários para obter estimativas eficientes de todos os parâmetros do modelo. Cabe a ressalva de que o intercepto é estimado como $\beta_0^* = \beta_0(1-\rho)$, de onde obtemos β_0 por $\beta_0 = \frac{\beta_0^*}{1-\rho}$. É usual, entretanto, não conhecermos o valor de ρ, sendo necessária a adoção de procedimentos para estimação desse parâmetro.

Um primeiro método atribui-se a Cochrane & Orcutt (1949) e consiste na estimação iterativa de ρ considerando o conceito de correlação entre termos adjacentes normalmente atribuído a esse parâmetro. O procedimento é descrito pelos seguintes passos:

1. Estimar os parâmetros do modelo (8.1) via mínimos quadrados ordinários;
2. Obter os resíduos $\hat{\varepsilon}_t$, $t = 1, \ldots, T$, a partir do passo 1;
3. Estimar ρ via MQO na equação $\hat{\varepsilon}_t = \rho \hat{\varepsilon}_{t-1} + \nu_t$;
4. Transformar as variáveis do modelo usando $X^*_{j,t} = X_{j,t} - \hat{\rho} X_{j,t-1}$, $j \in 1, \ldots, k$, e $Y^*_t = Y_t - \hat{\rho} Y_{t-1}$;
5. Estimar os coeficientes da equação (8.7a) via MQO;
6. Obter os resíduos do modelo ajustado em 5;
7. Repetir os passos 3 - 6 até que a diferença entre as estimativas de ρ para duas iterações consecutivas seja inferior a algum limite de convergência arbitrado.

Note que o procedimento de Cochrane-Orcutt elimina a primeira observação ao transformar as variáveis (passo 4), o que não implica em perda significativa de informação à medida que a amostra cresce. Computacionalmente, podemos obter as estimativas dos coeficientes por meio da função `cochrane.orcutt()` no pacote **orcutt** (Spada et al., 2012).

Outro método de estimação do modelo (8.6a)-(8.6b) foi introduzido por Prais & Winsten (1954) como uma modificação do procedimento de Cochrane-Orcutt no sentido de que não é necessária a exclusão da primeira observação. O algoritmo de estimação assemelha-se ao anteriormente apresentado, com a exceção de que no passo 4 as variáveis no primeiro instante de tempo são construídas como $Y^*_1 = \sqrt{1 - \hat{\rho}^2}\, Y_1$ e $X^*_{j,1} = \sqrt{1 - \hat{\rho}^2}\, X_{j,1}$, $j \in \{1, \ldots, k\}$. Este método se mostra mais eficiente em amostras pequenas, visto que não elimina nenhuma observação da amostra. No R encontra-se disponível a função `prais.winsten()` do pacote **prais** (Mohr, 2015).

Os procedimentos tratados acima dependem da suposição de que os resíduos são estimadores consistentes do termo de erro, o que requer estimativas consistentes dos coeficientes usados no cálculo dos resíduos. Um caso importante onde os resíduos não são estimados consistentemente aparece em contextos onde a variável dependente defasada é usada como regressora no modelo e os erros são autocorrelacionados. Para situações dessa natureza, os resíduos obtidos via MQO não podem ser usados para estimar ρ. Uma alternativa aos métodos anteriores aparece em Hildreth & Lu (1960), cujo estimador procura o valor de ρ, $-1 \leq \rho \leq 1$, que minimiza a soma dos quadrados dos resíduos no modelo (8.7a).

Todas as metodologias descritas exigem cautela em sua realização, uma vez que os algoritmos podem resultar em mínimos locais ao invés de globais.

A segunda abordagem para lidar com a correlação serial dos erros segue em direção oposta

ao apresentado anteriormente, tornando a inferência válida através da correção dos erros padrão dos estimadores de MQO pelo método descrito em Newey & West (1987) ao invés de introduzir uma estrutura para o erro. É importante frisar que tal correção somente é válida em casos onde o termo de erro não é correlacionado com nenhuma das variáveis explicativas do modelo. A implementação do método pode ser realizada através do pacote **sandwich** (Zeileis, 2004) do R, todavia maior detalhamento do método encontra-se fora escopo do presente capítulo.

Exemplo com dados artificiais

Apresentaremos agora um breve exemplo com dados artificiais para ilustrar a situação em que o modelo de regressão clássico não satisfaz a suposição de ausência de correlação serial dos erros.

A seguir temos o início da nossa base de dados, que contém 4 variáveis ao longo do tempo num total de 30 observações (cons, price, income, temp).

```
> head(dados)
  cons price income temp
1 0.39  0.27     78   41
2 0.37  0.28     79   56
3 0.39  0.28     81   63
4 0.42  0.28     80   68
5 0.41  0.27     76   69
6 0.34  0.26     78   65
```

Iniciamos o exemplo estimando a regressão

$$\text{Cons}_t = \beta_0 + \beta_1 \text{Price}_t + \beta_2 \text{Income}_t + \beta_3 \text{Temp}_t + \varepsilon_t \tag{8.8}$$

através da função `lm()`. Como o modelo não considera defasagens da variável dependente no lado direito de (8.8), podemos estimar $\boldsymbol{\varepsilon} = \{\varepsilon_1, \ldots, \varepsilon_T\}$ consistentemente a partir dos resíduos da estimação de (8.8) via MQO.

```
> # Estimando a Regressão Linear Clássica
> reg <- lm(cons ~ price + income + temp, data = dados)
```

Lembrando que tanto o teste de Durbin-Watson quanto o de Breusch-Godfrey tem como hipótese nula a ausência de correlação serial dos erros, conduzimos no R ambos os testes considerando autocorrelação de primeira ordem e os resíduos da regressão acima. Assumindo nível de significância igual a 5%, os dois testes rejeitam a hipótese nula uma vez que os p-valores de ambos são inferiores a 0,05 ($\text{p-valor}_{DW} = 0{,}04\%$ e $\text{p-valor}_{BG} = 4{,}7\%$).

```
> # Testando correlação serial de primeira ordem
> require(lmtest)
> # Durbin-Watson
> dw_reg <- dwtest(cons ~ price + income + temp, data = dados)
> dw_reg
             Durbin-Watson test

     data:  cons ~ price + income + temp
     DW = 1.0384, p-value = 0.0003768
     alternative hypothesis: true autocorrelation is greater than 0
> # Breusch-Godfrey
> bg_reg <- bgtest(cons ~ price + income + temp, data = dados)
> bg_reg
             Breusch-Godfrey test for serial correlation of order up to 1

     data:  cons ~ price + income + temp
     LM test = 3.928, df = 1, p-value = 0.04749
```

Devido à presença de correlação serial de primeira ordem nos erros do modelo (8.8), surge a necessidade de incluirmos uma estrutura em ε_t que reflita tal comportamento. Assumimos, então, o modelo

$$\text{Cons}_t = \beta_0 + \beta_1 \text{Price}_t + \beta_2 \text{Income}_t + \beta_3 \text{Temp}_t + \varepsilon_t \tag{8.9a}$$

$$\varepsilon_t = \rho \varepsilon_{t-1} + \nu_t. \tag{8.9b}$$

Explicitamos na Seção 8.3.2 dois procedimentos de estimação do modelo (8.9a)-(8.9b): Cochrane-Orcutt e Prais-Winsten. Ambos foram implementados no R e os resultados são apresentados a seguir:

```
> # Estimação do modelo com estrutura no erro
> # Cochrane-Orcutt
> require(orcutt)
> co_reg <- cochrane.orcutt(reg)
> co_reg
```

```
$Cochrane.Orcutt

Call:
lm(formula = YB ~ XB - 1)

Residuals:
      Min        1Q    Median        3Q       Max
-0.072610 -0.012760  0.000906  0.015209  0.077841

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
XB(Intercept)  0.2424113  0.2531469   0.958   0.3474
XBprice       -1.1927018  0.6873988  -1.735   0.0950 .
XBincome       0.0031820  0.0014969   2.126   0.0436 *
XBtemp         0.0035572  0.0005339   6.662 5.56e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0317 on 25 degrees of freedom
Multiple R-squared:  0.9837,        Adjusted R-squared:  0.9811
F-statistic: 378.3 on 4 and 25 DF,  p-value: < 2.2e-16

$rho
[1] 0.3788941

$number.interaction
[1] 12
```

```
> # Prais-Winsten
> require(prais)
> pw_reg <- prais.winsten(cons ~ price + income + temp, data = dados)
> pw_reg
```

```
[[1]]

Call:
lm(formula = fo)

Residuals:
      Min        1Q    Median        3Q       Max
-0.079171 -0.017838 -0.000846  0.010029  0.080424

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
Intercept  5.709e-01  2.594e-01   2.201 0.036834 *
price     -1.297e+00  6.756e-01  -1.920 0.065873 .
income     6.857e-05  1.928e-03   0.036 0.971899
temp       3.076e-03  6.847e-04   4.492 0.000128 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03255 on 26 degrees of freedom
Multiple R-squared:  0.9435,        Adjusted R-squared:  0.9348
F-statistic: 108.5 on 4 and 26 DF,  p-value: 7.947e-16

[[2]]
       Rho Rho.t.statistic Iterations
 0.7018387        4.481754         49
```

Na Tabela 8.1 resumimos os valores estimados dos parâmetros pelos três diferentes métodos abordados. Note que os coeficientes para o modelo clássico de regressão e para o método de Cochrane-Orcutt são bastante similares para todo β_k, $k = 0,1,2,3$, mesmo que ρ seja estimado diferente de zero. Já o procedimento de Prais-Winsten apresenta resultados ligeiramente diferentes. Como o número de observações é pequeno, o segundo procedimento parece mais eficiente ao incluir a primeira observação.

Parâmetro	Prais-Winsten	Cochrane-Orcutt	Regressão Clássica
β_0	0.5709	0.2424	0.2672
β_1	-1.2973	-1.1927	-1.2528
β_2	0.0001	0.0032	0.0032
β_3	0.0031	0.0036	0.0034
ρ	0.7018	0.3790	-

Tabela 8.1: Estimativas

Nesse exemplo tentamos ilustrar a implementação dos testes de correlação serial e dos procedimentos de estimação do modelo estático com estrutura autoregressiva de primeira ordem para os erros. Ainda que os dados não sejam reais e pouca, ou nenhuma, interpretação possa ser dada a eles, nosso intuito foi expor a sintaxe das funções no R e prover base para maior aprofundamento do leitor.

Modelos autoregressivos com defasagens distribuídas

Até o momento tratamos a correlação serial de $\varepsilon = \{\varepsilon_1, \ldots, \varepsilon_T\}$ como uma violação das suposições do modelo clássico de regressão linear. Podemos, em contrapartida, enxergar tal comportamento dos erros como um indício de incorreção na especificação do modelo.

O impacto sobre Y de um choque em determinada variável explicativa X pode não ocorrer imediatamente, sendo Y afetado somente após alguns instantes de tempo, ou mesmo Y pode afetar seu próprio valor em tempos posteriores. A omissão dessa dinâmica pode induzir correlação serial nos erros, sendo interessante nessas circunstâncias optar pelo uso de modelos dinâmicos.

Quando a dinâmica do modelo é ditada pelo comportamento das variáveis independentes defasadas nos referimos a modelos com defasagens distribuídas, sendo as defasagens responsáveis por explicar Y. Se somente os valores passados de Y determinam seu valor em t, a dinâmica de Y pode ser descrita segundo modelos autoregressivos. É possível, ainda, combinar os dois modelos anteriores em uma única equação, originando os chamados modelos autoregressivos com defasagens distribuídas (do inglês, ADL - *Autoregressive Distributed Lag*)[5].

A forma geral do modelo ADL com p defasagens para Y e q defasagens para X, denotado por ADL(p,q), é dada por

$$\phi(L)Y_t = \alpha + \theta(L)X_t + \nu_t, \tag{8.10}$$

onde $\phi(L) = 1 - \phi_1 L - \phi_2 L^2 - \cdots - \phi_p L^p$ cujas raízes não pertencem ao círculo unitário, o que significa dizer que Y é estacionário, $\theta(L) = \theta_0 + \theta_1 L + \theta_2 L^2 + \cdots + \theta_q L^q$ e L é um operador de defasagem tal que $L^k Y_t = Y_{t-k}$. Podemos reescrever (8.10) como

$$Y_t = \alpha + \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + \theta_0 X_t + \theta_1 X_{t-1} + \cdots + \theta_q X_{t-q} + \nu_t. \tag{8.11}$$

Supondo que os erros $\nu = \{\nu_1, \ldots, \nu_T\}$ são ruídos brancos, o modelo (8.11) pode ser estimado via mínimos quadrados ordinários. A função `dynlm()` (Zeileis, 2016) do pacote de mesmo nome permite a inclusão de defasagens e diferenças das variáveis no modelo, retornando as estimativas de MQO dos parâmetros.

[5] O capítulo se restringe à descrição do modelo ADL, sendo os outros dois modelos casos particulares deste. Mais detalhes sobre os modelos podem ser vistos em Greene (2003).

Escolhendo as defasagens p e q:

O emprego do modelo ADL requer a especificação a priori do número de defasagens de X e Y no modelo. Entretanto, são raras as situações onde a teoria nos informa sobre os valores exatos de p e q, sendo necessário determiná-los empiricamente. Diferentes métodos estão disponíveis para avaliar o número apropriado de defasagens no modelo, não existindo um "método correto". A escolha é, portanto, usualmente feita pela combinação de métodos.

Um primeiro método trata da especificação da defasagem através de testes de significância dos parâmetros. Podemos começar com um número elevado de defasagens e avaliar a significância do coeficiente associado à maior defasagem. Caso esse coeficiente seja estatisticamente significativo, optamos por esse modelo, em contrapartida, se o coeficiente não for significativo, estimamos o modelo com uma defasagem a menos e continuamos o processo até que o coeficiente associado à maior defasagem seja significativo. O caminho contrário também pode ser empregado, ou seja, começamos pelo modelo com a menor defasagem e inserimos novas defasagens até que o coeficiente para a variável de maior defasagem não seja mais significativo.

Outro método de determinação de p e q envolve o cálculo de critérios de informação. Tais critérios mensuram a quantidade de informação sobre a variável dependente contida no conjunto de variáveis independentes, considerando o erro padrão das estimativas dos coeficientes e penalizando pelo número de parâmetros do modelo. A literatura dispõe de uma gama de critérios de informação, figurando entre os mais utilizados o critério de informação de Akaike (AIC) e o critério de informação bayesiano (BIC). Por meio dessa metodologia, escolhemos o número de defasagens com base no modelo que retorna o menor valor desses critérios. Cabe a ressalva que o cálculo dessas medidas deve considerar o mesmo intervalo de tempo para todos os modelos a fim de torná-los comparáveis.

Os procedimentos apresentados acima podem ser aplicados tanto na escolha de p, defasagens da variável dependente, quanto de q, defasagens das variáveis independentes. É importante enfatizar, ainda, que usualmente nenhuma defasagem até p e q é omitida. No R, a função `glmulti()` do pacote **glmulti** (Calcagno, 2013) recebe as variáveis dependentes e independentes do modelo e seleciona automaticamente o melhor modelo segundo o critério de informação escolhido.

Modelo de correção de erro

Por todo este capítulo descrevemos diferentes metodologias para lidar com variáveis observadas ao longo do tempo. Os modelos apresentados supõem que tais variáveis são estacionárias. É comum, entretanto, encontrarmos situações onde esse pressuposto não é satisfeito como, por exemplo, contextos onde há uma quebra estrutural em determinado instante de tempo ou mesmo quando a variável apresenta tendência.

Supor estacionariedade das variáveis em modelos de regressão linear quando de fato esse pressuposto é violado pode conduzir a conclusões inapropriadas. Mesmo que duas variáveis não sejam relacionadas, ocasionalmente a estimação via mínimos quadrados ordinários resulta em coeficiente de determinação (R^2) elevado, significando que a covariável conseguiu explicar bem a variável dependente, e estatísticas de teste que levam à conclusão de significância dos parâmetros. Esse é um exemplo de regressão espúria, introduzido por Granger & Newbold (1974).

Existem diferentes abordagens para variáveis não estacionárias, porém nos concentraremos naquelas para variáveis integradas. Lembre-se que uma variável Z é dita integrada de ordem k se sua k-ésima diferença é estacionária, ou seja, $Z_t \sim I(k)$ se $\Delta^k Z_t = (1-L)^k Z_t$ e então $\Delta^k Z_t \sim I(0)$.

Considere um modelo de regressão com apenas duas variáveis integradas de primeira ordem, $Y_t \sim I(1)$ e $X_t \sim I(1)$, dado por $Y_t = \phi X_t + \nu_t$. Uma alternativa para contornar os problemas induzidos pela não estacionariedade consiste em realizar a regressão tomando a primeira diferença das variáveis, isto é, $\Delta Y_t = \beta \Delta X_t + \eta_t$, onde ΔY_t e ΔX_t são estacionárias. Um aspecto negativo dessa metodologia decorre da perda de eventual informação de longo prazo entre as variáveis.

Um caso de especial interesse em econometria surge quando a combinação linear de duas variáveis integradas de primeira ordem resulta em um termo de erro estacionário. Retornando ao exemplo do parágrafo anterior, teríamos $Y_t, X_t \sim I(1)$ mas $\nu_t \sim I(0)$. Este caso define o conceito de cointegração, que em economia usualmente está relacionado ao conceito de equilíbrio de longo prazo. A presença de cointegração entre X e Y permite que os modelos clássicos de regressão sejam estimados corretamente pelo método de mínimos quadrados ordinários.

Quando duas séries são cointegradas, perturbações em qualquer uma delas provocam alterações em sua relação de longo prazo. A taxa à qual o sistema retorna ao equilíbrio após tais perturbações

pode ser estimada a partir dos chamados modelos de correção de erro (ECM)[6], escritos como

$$\Delta Y_t = \beta \Delta X_t + \gamma(Y_t - \phi X_t) + \eta_t, \tag{8.12}$$

onde a taxa de retorno ao equilíbrio é γ, $\gamma < 0$, e β nos informa sobre os efeitos de curto prazo de X_t sobre Y_t. Note que o modelo de correção de erro é "balanceado" no sentido que todas as variáveis no lado direito de (8.12) são estacionárias e, portanto, o método de mínimos quadrados é aplicável.

Equivalência entre ADL(1,1) e ECM:

Considere o modelo ADL(1,1) dado pela equação

$$Y_t = \beta_0 X_t + \beta_1 X_{t-1} + \rho Y_{t-1} + \eta_t. \tag{8.13}$$

Subtraindo Y_{t-1} em ambos os lados e substituindo X_t por $\Delta X_t + X_{t-1}$ temos

$$\Delta Y_t = \beta_0 \Delta X_t + (\beta_0 + \beta_1) X_{t-1} + (\rho - 1) Y_{t-1} + \eta_t. \tag{8.14}$$

Rearrumando a equação (8.14) encontramos

$$\Delta Y_t = \beta_0 \Delta X_t + (\rho - 1)\left[Y_{t-1} + \left(\frac{\beta_0 + \beta_1}{\rho - 1} \right) X_{t-1} \right] + \eta_t. \tag{8.15}$$

Fazendo $\gamma = (\rho - 1)$, $\phi = \left(\frac{\beta_0 + \beta_1}{\rho - 1} \right)$ e $\beta = \beta_0$ em (8.15) chegamos ao modelo de correção de erro na equação (8.12).

Note que a equivalência não é exata, visto que alguns parâmetros são escritos em função de outros.

Estimação do modelo:

O modelo de correção de erro apresentado em (8.12) pode ser estimado de duas maneiras diferentes. O primeiro método consiste no procedimento de Engle-Granger e segue as seguintes etapas:

[6] Os modelos de correção de erro não são aplicados, necessariamente, a variáveis não-estacionárias (Keele & De Boef, 2004).

1. Estimar o modelo $Y_t = \phi X_t + \varepsilon_t$;
2. A partir dessas estimativas obter os resíduos $\hat{\varepsilon}_t = Y_t - \hat{\phi} X_t$;
3. Estimar o modelo $\Delta Y_t = \beta_0 + \beta_1 \Delta X_t + \gamma \hat{\varepsilon}_{t-1} + \nu_t$.

Na segunda metodologia nos valeremos da equivalência entre o modelo ADL(1,1) e o ECM e obteremos as estimativas dos coeficientes do modelo de equação única

$$\Delta Y_t = \beta \Delta X_t + \gamma Y_{t-1} - \gamma \phi X_{t-1}. \tag{8.16}$$

Aplicação à expectativa de enflação dos consumidores

Apresentados os conceitos relacionados à regressão dinâmica, trataremos nessa seção da implementação desses modelos à expectativa de inflação dos consumidores brasileiros bem como da aplicação dos testes de estacionariedade e correlação cabíveis.

Desde 2005, o Instituto Brasileiro de Economia (FGV\IBRE) inclui na Sondagem do Consumidor uma pergunta quantitativa sobre a expectativa de inflação individual para os próximos 12 meses. Pretendemos estimar um modelo de correção de erro baseado nos dois procedimentos da Seção 8.5, com a expectativa de inflação sendo explicada pelo Índice de Preços ao Consumidor Amplo (IPCA - IBGE). Os dados aplicados no exemplo pertencem ao período entre setembro de 2005 e dezembro de 2013 e são parcialmente ilustrados na Tabela 8.2.

Data	Expectativa de Inflação do Consumidor	IPCA
Set/05	9.44	6.04
Out/05	9.50	6.36
Nov/05	9.13	6.22
⋮	⋮	⋮
Nov/2013	9.01	5.77
Dez/2013	9.25	5.91

Tabela 8.2: Resumo dos dados

O primeiro passo de nossa análise contempla o teste de estacionariedade das variáveis em questão. A literatura dispõe de diversos testes, entretanto somente realizaremos o teste de Dickey-Fuller

Aumentado (ADF)[7]. Esse teste tem como hipótese nula a presença de raiz unitária e pode ser aplicado no R através da função `ur.df()` do pacote **urca** (Pfaff et al., 2016).

Iniciaremos os testes de estacionariedade com a expectativa de inflação. A seguir temos os resultados do teste ADF aplicado a essa variável, assumindo defasagem máxima igual a 12 e adotando o AIC como critério de escolha da defasagem. O teste retornou defasagem igual a 1 e estatística 0,5713. Como a estatística de teste é maior que o valor crítico ao nível de 5% de confiança, a saber -1,95, aceitamos a hipótese nula e, portanto, a variável em questão não é estacionária.

```
> # Expectativa de Inflação
> adf_expinf <- ur.df(expinf_cons, type = "none", lags = 13, selectlags = "AIC")
> summary(adf_expinf)

###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################
Test regression none

Call:
lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)

Residuals:
     Min       1Q   Median       3Q      Max
-1.00779 -0.27690  0.02725  0.25452  1.07022

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
z.lag.1     0.003152   0.005518   0.571   0.5694
z.diff.lag -0.190529   0.104648  -1.821   0.0722 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4045 on 84 degrees of freedom
Multiple R-squared:  0.04016,        Adjusted R-squared:  0.01731
F-statistic: 1.757 on 2 and 84 DF,  p-value: 0.1788


Value of test-statistic is: 0.5713
Critical values for test statistics:
     1pct  5pct 10pct
tau1 -2.6 -1.95 -1.61
```

[7]Para mais detalhes sobre testes de estacionariedade consultar Enders (2008).

Para avaliar se o teste ADF foi conduzido corretamente, a Figura 8.2 analisa a presença de autocorrelação nos resíduos do modelo utilizado pelo teste e a consequente necessidade de inclusão de mais defasagens no modelo. Como podemos observar, não existe autocorrelação significativa de nenhuma ordem e o teste parece correto.

```
> acf(adf_expinf@res, main = "")
```

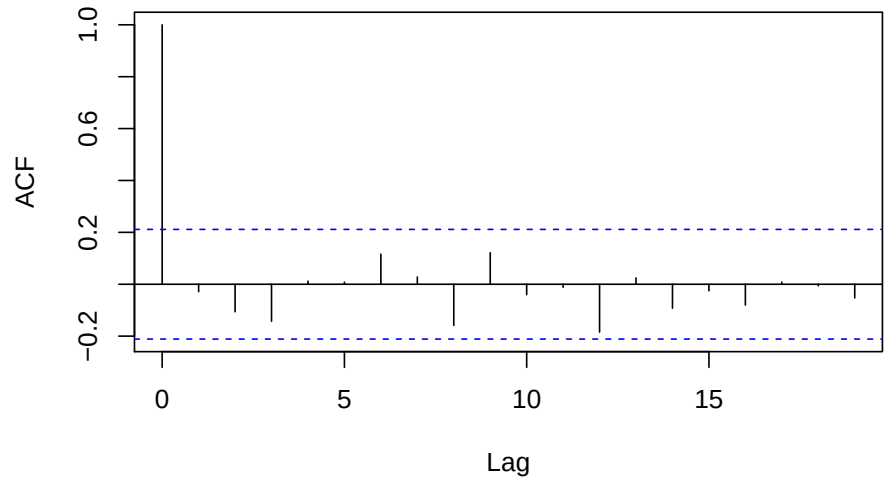


Figura 8.2: ACF dos resíduos do teste ADF - expectativa de inflação

Passaremos à análise do IPCA. A seguir apresentamos os resultados do teste ADF e a correspondente função de autocorrelação dos resíduos do teste (Figura 8.3). Novamente assumimos defasagem máxima igual a 12 e seleção do lag via AIC.

```
> # IPCA
> adf_ipca <- ur.df(ipca, type = "none", lags = 12, selectlags = "AIC")
> summary(adf_ipca)
```

```
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################
Test regression none
Call: lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)

Residuals:
     Min       1Q   Median       3Q      Max
-0.54453 -0.10688  0.04011  0.15284  0.41034

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
z.lag.1    0.0004985  0.0042074   0.118    0.906
z.diff.lag 0.5559689  0.0905489   6.140 2.55e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2096 on 85 degrees of freedom
Multiple R-squared:  0.3104,        Adjusted R-squared:  0.2942
F-statistic: 19.13 on 2 and 85 DF,  p-value: 1.378e-07

Value of test-statistic is: 0.1185
Critical values for test statistics:
     1pct  5pct 10pct
tau1 -2.6 -1.95 -1.61
```

```
> acf(adf_ipca@res, main = "")
```

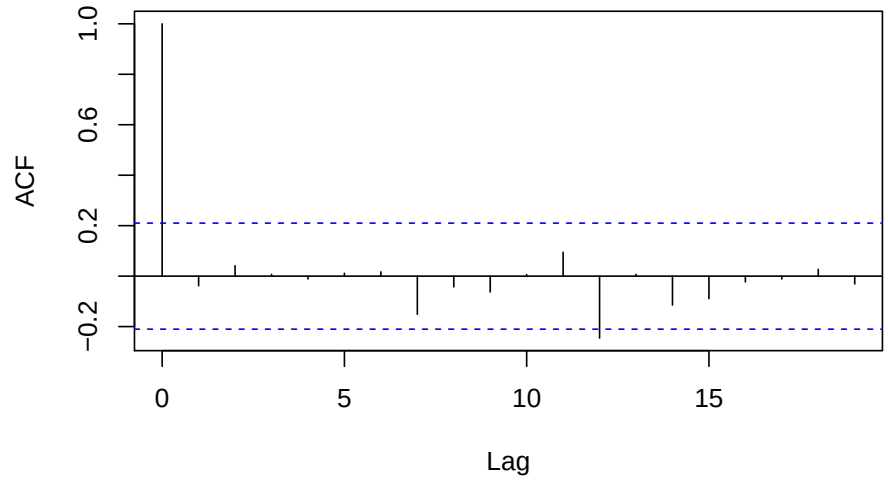


Figura 8.3: ACF dos resíduos do teste ADF - IPCA

O teste retornou o modelo com apenas uma defasagem como o de menor AIC e estatística de teste (0,1185) que nos leva à conclusão de não rejeição da hipótese nula, ou seja, a variável IPCA não é estacionária. Corroborando os resultados encontrados, a função de autocorrelação dos resíduos (Figura 8.3) não fornece indícios de incorreção do teste, visto que a autocorrelação não é significativa para nenhuma ordem.

Tendo concluído pela não estacionaridade das variáveis em estudo, os modelos clássicos de regressão linear não podem ser empregados. O próximo passo é, então, verificar se as variáveis são cointegradas. Esse teste avaliará se os resíduos da regressão $\text{ExpInf}_t = \alpha + \beta \text{IPCA}_t + \varepsilon_t$ são estacionários via teste ADF.

```
> # Expectativa de Inflação x IPCA
> ajuste_coin1 <- lm(expinf_cons ~ ipca - 1)
> summary(ajuste_coin1)
```

```
Call: lm(formula = expinf_cons ~ ipca - 1)

Residuals:
    Min      1Q  Median      3Q     Max
-2.8746 -0.4482  0.0905  0.8496  2.1819

Coefficients:
     Estimate Std. Error t value Pr(>|t|)
ipca  1.47992    0.01979   74.78   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.06 on 99 degrees of freedom
Multiple R-squared:  0.9826,        Adjusted R-squared:  0.9824
F-statistic:  5593 on 1 and 99 DF,  p-value: < 2.2e-16
```

```
> adf_coin1 <- ur.df(ajuste_coin1$residuals, "none", lags = 12, selectlags = "AIC")
> summary(adf_coin1)
```

```
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################
Test regression none
Call: lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)

Residuals:
     Min       1Q   Median       3Q      Max
-1.37125 -0.31229  0.07097  0.36439  0.84176

Coefficients:
           Estimate Std. Error t value Pr(>|t|)
z.lag.1    -0.14800    0.05163  -2.867  0.00523 **
z.diff.lag  0.07835    0.10577   0.741  0.46092
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4942 on 85 degrees of freedom
Multiple R-squared:  0.0884,        Adjusted R-squared:  0.06695
F-statistic: 4.121 on 2 and 85 DF,  p-value: 0.01957

Value of test-statistic is: -2.8666

Critical values for test statistics:
     1pct  5pct 10pct
tau1 -2.6 -1.95 -1.61
```

```
> acf(adf_coin1@res, main = "")
```

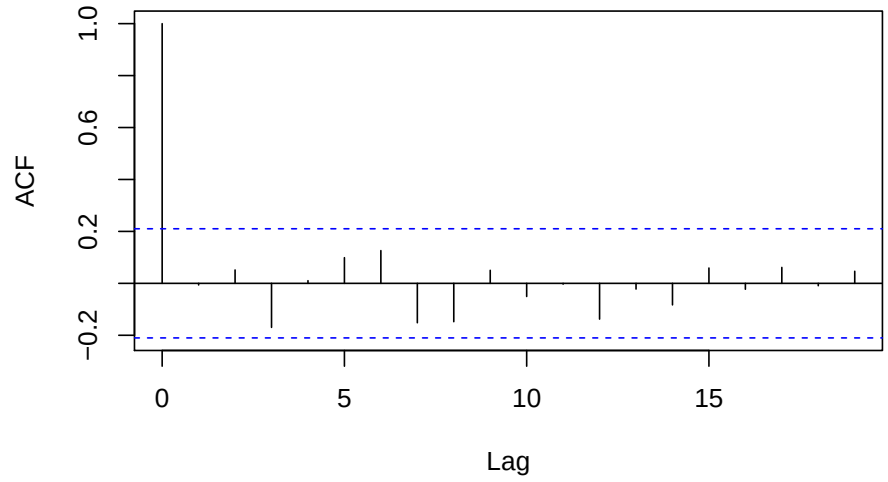


Figura 8.4: ACF dos resíduos do teste ADF - regressão

A análise da estatística do teste ADF (-2,8666) nos leva à rejeição da hipótese nula e os resíduos

do teste não apresentam autocorrelação significativa (Figura 8.4). Portanto, concluímos que as variáveis são cointegradas. Assim, podemos estimar o modelo de correção de erro para encontrar as relações de longo e curto prazo entre elas. A Tabela 8.3 apresenta os valores estimados dos parâmetros usando as duas metodologias descritas na Seção 8.5. Como esperado, ambos os procedimentos resultam em estimativas similares.

Parâmetro	Única Equação	Duas Equações
β	0.2862	0.2860
γ	-0.0933	-0.0933
ϕ	1.4830	1.4800

Tabela 8.3: Estimativas do Modelo de Correção de Erros

Para finalizar, apresentamos o código em R que gerou as estimativas da tabela acima.

```
> # Estimação do modelo
> require(dynlm)
> # Procedimento em duas etapas
> reg1 <- lm(expinf_mensal$x ~ IPCA$x - 1)
> res <- ts(reg1$residuals, start = c(2005,09), freq = 12)
> reg2 <- dynlm(d(expinf_cons, 1) ~ d(ipca, 1) + L(res, 1) -1 )
> # Procedimento em única etapa
> reg <- dynlm(d(expinf_cons, 1) ~ d(ipca, 1) + L(expinf_cons, 1) + L(ipca, 1) -1)
```

Considerações finais

Neste capítulo exploramos as consequências do uso de modelos clássicos de regressão linear quando a variável dependente é observada ao longo do tempo. Em particular, nos aprofundamos na presença de correlação serial nos resíduos e discutimos duas frentes para solucionar o problema: *(i)* incluir uma estrutura autoregressiva nos erros e aplicar os procedimentos de Cochrane-Orcutt e Prais-Winsten; ou *(ii)* tratar a correlação serial como um indício de especificação incorreta do modelo e incluir defasagens tanto das variáveis independentes quanto da variável dependente pelo emprego dos modelos autoregressivos com defasagens distribuídas (ADL). Um exemplo com dados artificiais contribuiu para o entendimento dos métodos em *(i)* e expôs os códigos utilizados no software R.

Os procedimentos descritos até a Seção 8.4 pressupõem estacionariedade de todas as variáveis. Em contextos em que essa suposição não é razoável, podemos incorrer em conclusões equivocadas. Apesar das diferentes formas de não estacionariedade, nos restringimos a variáveis integradas e apre-

sentamos o modelo de correção de erro (ECM) como possível solução para problemas associados a regressões espúrias. Na Tabela 8.4 encontra-se um resumo dos modelos cabíveis a diferentes situações de estacionariedade e não estacionariedade.

Variável Independente	Variável Dependente	Erro	Método de Estimação
I(0)	I(0)	I(0)	ADL
I(1)	I(0)	I(1)	Modelo mal especificado
I(1)	I(1)	I(1)	Primeira diferença em todas as variáveis. Modelo ADL
I(1)	I(1)	I(0)	Variáveis Cointegradas. ECM

Tabela 8.4: Resumo dos modelos

Por fim, ilustramos o emprego dos modelos ADL e ECM na modelagem da série histórica das expectativas de inflação dos consumidores brasileiros usando o IPCA como variável explicativa.

Modelo Vetorial Autoregressivo

Pedro Costa Ferreira

Lucas Farias Lima

Introdução

É comum, principalmente para modelos econômicos, que existam fortes evidências de que uma variável seja definida dentro de um sistema; um exemplo básico são os modelos macroeconômicos de curto prazo onde PIB, consumo, investimento e gastos governamentais são determinados simultaneamente.

Nestes casos, é um erro de especificação modelar esses processos de maneira puramente autoregressiva, ou não controlando a dinâmica temporal, dado que podem existir relações entre os choques contemporâneos. Assim, surgiram os modelos de função de transferência, onde se assumia uma relação unidirecional entre séries de tempo. Todavia, essa hipótese não corresponde a muitos casos de interesse, como no exemplo do parágrafo acima.

Uma solução surge com os modelos VAR (do inglês, *Vector Autoregressive*) no trabalho seminal de Christopher Sims, *Macroeconomics and Reality* (Sims, 1980), que propõe tratar todas as variáveis do sistema simetricamente, sem fazer hipóteses sobre a estrutura de correlação entre elas.

A abordagem de Sims despertou o interesse teórico e aplicado na possibilidade de se verificar a relação entre as váriaveis do sistema, o que é comumente feito através das funções de impulso-resposta e decomposição de variância; relacionadas ao interessante resultado que nos permite representar o VAR como um processo de médias móveis infinitas.

Entretanto, como em muitos casos é interessante que o modelo possua sentido econômico, o que é de certa maneira sacrificado pela ausência de uma estrutura de causalidade na especificação reduzida, vários esforços surgiram na busca por meios de se identificar os parâmetros estruturais após a estimação do VAR, o que deu origem aos modelos SVAR (do inglês *structural VAR*).

A abordagem vetorial permite também uma análise muito mais detalhada das relações de cointegração, advindas da extensão do importante trabalho em Engle & Granger (1987), que dá origem aos modelos VECM (do inglês *vector error correction model*), dos quais o método de estimação proposto por Johansen (1988) é a principal referência.

O conteúdo deste texto está baseado principalmente nos livros de Pfaff (2008), Enders (2008) e Lütkepohl (2005). A não ser quando explicitamente mencionado, fazem parte do pacote **vars** (Pfaff et al., 2013). Ademais, todos os códigos R e dados utilizados para produção deste capítulo, alguns

antes dos resultados apresentados, estão todos disponíveis em:

<https://github.com/pedrocostaferreira/Analise-de-Series-Temporais-em-R>.

Este capítulo está estruturado da seguinte forma: após esta introdução, a seção 9.2 define o modelo VAR e apresenta suas principais propriedades, como estabilidade e representação em médias móveis. Na seção 9.3 são abordadas a estimação e previsão, assim como algumas ferramentas de análise do modelo, como funções de impulso-resposta e Granger-causalidade. As duas seções seguintes, 9.4 e 9.5 apresentam as versões mais robustas do modelo - estrutural (SVAR) e de correção de erro (VECM) - que permitem a identificação de relações estruturais e de longo prazo, respectivamente. Finalmente, a seção 9.6 apresenta um breve exemplo da metodologia, aplicada a dados de investimento e confiança da economia brasileira.

O Modelo VAR

Definição

Abordaremos os principais aspectos dos modelos VAR estudando uma especificação bivariada simples, com a média do processo sendo o único termo determinístico; o que não afeta a generalidade do que será exposto. Tal abordagem auxilia a abstração e facilita a compreensão de vários aspectos da metodologia.

Assim, partindo da perspectiva de modelos dinâmicos, e seguindo a apresentação feita por Enders (2008, p. 264), olhamos primeiramente para o que seria um modelo em sistema bivariado (equação (9.1)). A idéia é que se não sabemos se há *feedback*[1], partimos de um modelo em que as inovações contemporâneas de um processo afetam o outro.

$$y_{1,t} = \alpha_1 - \delta_2 y_{2,t} + \beta_1 y_{1,t-1} + \beta_2 y_{2,t-1} + e_{1,t}, \tag{9.1a}$$

$$y_{2,t} = \alpha_2 - \delta_1 y_{1,t} + \beta_2 y_{2,t-1} + \beta_1 y_{1,t-1} + e_{2,t}, \tag{9.1b}$$

[1] Relação onde há causalidade mútua.

No sistema 9.1, os termos $y_{1,t}$ e $y_{2,t}$ representam a dinâmica contemporânea, relaxando a hipótese de restrição de *feedback*. O modelo acima está no que chamamos forma estrutural, devido à presença do termo contemporâneo no lado direito de cada equação.

Para colocá-lo na forma reduzida, escrevemos o modelo na forma matricial a partir do rearranjo:

$$y_{1,t} + \delta_2 y_{2,t} = \alpha_1 + \beta_1 y_{1,t-1} + \beta_2 y_{2,t-1} + e_{1,t}, \tag{9.2a}$$

$$y_{2,t} + \delta_1 y_{1,t} = \alpha_2 + \beta_2 y_{2,t-1} + \beta_1 y_{1,t-1} + e_{2,t}, \tag{9.2b}$$

a partir do qual obtemos:

$$\begin{bmatrix} 1 & \delta_2 \\ \delta_1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_t = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} + \begin{bmatrix} \beta_1 & \beta_2 \\ \beta_2 & \beta_1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_{t-1} + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}_t \tag{9.3}$$

que podemos simplificar como:

$$Ay_t = \alpha + B^* y_{t-1} + e_t, \tag{9.4}$$

bastando apenas pré-multiplicar a equação (9.3) por[2]:

$$A^{-1} = \begin{bmatrix} 1 & \delta_2 \\ \delta_1 & 1 \end{bmatrix}^{-1} = \frac{1}{1-\delta_1\delta_2} \begin{bmatrix} 1 & -\delta_1 \\ -\delta_2 & 1 \end{bmatrix} \tag{9.5}$$

donde obtemos:

$$y_t = a + By_{t-1} + \varepsilon_t, \tag{9.6}$$

a chamada forma reduzida, em que $a = A^{-1}\alpha$, $B = A^{-1}B^*$ e $\varepsilon_t = A^{-1}e_t$.

Vale observar que a pré-multiplicação acima torna os termos de erro da equação (9.6) (i.e. ε_t) combinações lineares dos erros puros (e_t), da equação (9.3). Essa transformação tem consequências importantes, principalmente quando se deseja estudar características estruturais das variáveis contidas

[2]Lembrando que se pode obter a inversa de uma matriz como o produto do inverso de seu determinante e sua adjunta.

no modelo.

De fato, os erros estarem correlacionados de forma contemporânea cria a necessidade de retornarmos ao modelo na forma (9.4) a partir de um modelo reduzido estimado caso desejemos realizar análises estruturais. Obtendo a matriz A, por exemplo. Para essa identificação (i.e. obter os parâmetros da versão estrutural a partir da reduzida) recorremos à restrições, que dão origem aos modelos estruturais (SVAR), que estudaremos melhor na seção 9.4.

Estabilidade e Estacionariedade

Um dos primeiros aspectos de modelos VAR a serem tratados em livros-texto é sua estabilidade. Da mesma forma que para modelos univariados, a definição das condições de estabilidade de modelos VAR tem direta relação com a possibilidade de estimação e realização de inferência do modelo.

Similarmente com a condição de que as raízes do polinômino definido nos operadores de defasagem de um modelo univariado estejam fora do círculo unitário (Enders, 2008, p. 28), para modelos multivariados verificamos uma condição a partir dos autovalores da matriz de coeficientes (Enders, 2008, p. 267).

Para ver o porquê disso mais claramente, sigamos o desenvolvimento de Lütkepohl (2005) e olhemos para a evolução da equação (9.6), assumindo que este se inicia em $t = 1$:

$$\begin{aligned}
y_1 &= a + By_0 + \varepsilon_1, \\
y_2 &= a + By_1 + \varepsilon_2, \\
&= a + B(a + By_0 + \varepsilon_1) + \varepsilon_2, \\
&= (I + B)a + B^2 y_0 + B\varepsilon_1 + \varepsilon_2, \\
&\vdots \\
y_t &= (I + B + B^2 + \cdots + B^{t-1})a + B^t y_0 + \sum_{i=0}^{t-1} B^i \varepsilon_{t-i} \\
&\vdots
\end{aligned} \tag{9.7}$$

Agora, se pudermos assumir que este processo se iniciou no que alguns autores chamam de

"passado infinito", reescrevendo a última equação de 9.7 temos:

$$y_t = (I + B + B^2 + \cdots + B^k)a + B^{k+1}y_{t-k-1} + \sum_{i=0}^{k} B^i \varepsilon_{t-i} \tag{9.8}$$

Daí fica claro que o sistema só converge caso B^k desapareça a medida que $k \to \infty$ (por isso a hipótese do "passado infinito", que nos permite assumir estarmos em um presente onde k é suficientemente grande). Neste caso, teremos que $(I + B + B^2 + \cdots + B^k) \to (1 - B)^{-1}$, o que permite representar o VAR como um processo de médias móveis:

$$y_t = \bar{y} + \sum_{i=0}^{\infty} B^i \varepsilon_{t-i} \tag{9.9}$$

onde $\bar{y} = \frac{a}{(I-B)}$

Por outro lado, para que essa convergência ocorra é necessário que os autovalores da matriz B (que chamaremos λ) sejam menores que 1 em módulo. Dado que estes satisfazem $det(I\lambda - B) = 0$, definindo $z = 1/\lambda$ (o recíproco de λ) temos a condição para estabilidade de um modelo VAR em (9.10). Ou seja, se as raízes do determinante de $B(z) = I - Bz$ estão fora do círculo unitário, o VAR em questão é estável[3].

$$det(I - Bz) \neq 0, \quad \text{para} \quad |z| \leq 1 \tag{9.10}$$

Para exemplificar, tomemos o processo a seguir, um VAR(1) em três dimensões:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_2 \end{bmatrix}_t = \begin{bmatrix} a_1 \\ a_2 \\ a_2 \end{bmatrix} + \begin{bmatrix} .5 & 0 & 0 \\ .1 & .1 & .3 \\ .0 & .2 & .3 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_2 \end{bmatrix}_{t-1} + \begin{bmatrix} u_1 \\ u_2 \\ u_2 \end{bmatrix}_t \tag{9.11}$$

Vericando a condição em (9.10) temos que:

[3] O cálculo dos autovalores é implementado na função `roots()`, vista na próxima seção.

$$det\left(\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}-\begin{bmatrix}.5&0&0\\.1&.1&.3\\.0&.2&.3\end{bmatrix}z\right)=det\left(\begin{bmatrix}1-0.5z&0&0\\-0.1z&1-0.1z&-0.3z\\0&-0.2z&1-0.3z\end{bmatrix}\right) \tag{9.12a}$$

$$=(1-0.5z)[(1-0.1z)(1-0.3z)-(-0.3z)(-0.2z)] \tag{9.12b}$$

$$=(1-0.5z)\left(-0.03z^2-0.4z+1\right) \tag{9.12c}$$

$$=0.015z^3+0.17z^2-0.9z+1 \tag{9.12d}$$

As raízes desse polinômio são $z_1=-15.4858$, $z_2=2$ e $z_3=2.1525$. Como todos estão fora do círculo unitário, temos que seus recíprocos (os autovalores da matriz de coeficientes) estão dentro do círculo unitário, e, portanto, o modelo é estável.

Como dissemos, estacionariedade e estabilidade são duas propriedades fortemente relacionadas. No caso de modelos VAR, tal relação se dá pelo fato de que modelos VAR estáveis são sempre estacionários. A negativa dessa afirmação não é verdadeira (Lütkepohl, 2005, p. 25). Entretanto processos instáveis (e.g. que "explodem" em uma determinada direção) não têm apelo econômico, e por isso não são de interesse.

Simularremos agora um processo VAR e analisaremos suas propriedades de acordo com o que discutimos, seguindo Pfaff (2008, p. 26).

Consideremos assim o modelo da equação (9.13), no qual por motivos discutidos na seção seguinte, definimos que ε_2 é uma combinação linear de e_1 e e_2 - os termos de erro estruturais de y_1 e y_2, respectivamente. Esse processo está simulado na Figura 9.1:

$$\begin{bmatrix}y_1\\y_2\end{bmatrix}_t=\begin{bmatrix}0.5\\1.0\end{bmatrix}+\begin{bmatrix}0.7&0.4\\0.2&0.3\end{bmatrix}\begin{bmatrix}y_1\\y_2\end{bmatrix}_{t-1}+\begin{bmatrix}\varepsilon_1\\\varepsilon_2\end{bmatrix}_t \tag{9.13}$$

É fácil ver que os processos aparentam seguir uma dinâmica comum. Em contrapartida, podemos

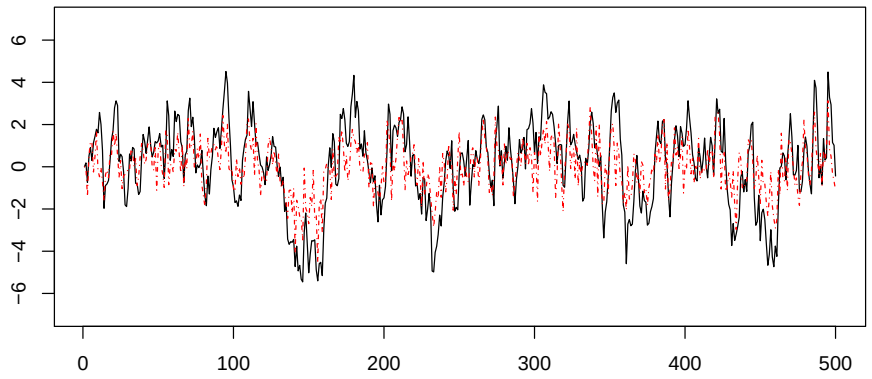


Figura 9.1: Simulação do sistema (9.13)

obter um processo não-estável simplesmente forçando que as raízes de seu polinômico sejam unitárias:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_{t-1} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix}_t \tag{9.14}$$

Na Figura 9.2 percebemos uma consequência imediata de processos instáveis: estes tendem a se "descolar", rompendo uma trajetória comum.

Esse processo, apesar da aparente instabilidade, apresenta uma característica interessante, exemplificada entre as observações de número 100 e 200: ainda que eles venham a romper uma trajetória comum, tendem recuperá-la nos períodos seguintes. Esse fenômeno, visto com mais detalhes na seção 9.5, é conhecido como cointegração; presente, grosso modo, quando duas séries não-estacionárias compartilham uma relação de longo prazo.

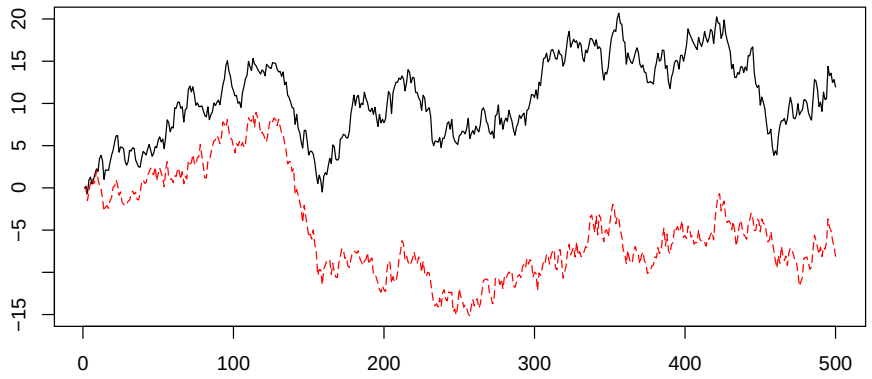


Figura 9.2: Simulação do sistema (9.14)

Estimação, Análise e Previsão

Estimação

Entre várias implicações relevantes da estabilidade, está a de que os coeficientes de um processo VAR estável de ordem infinita (representado como na equação (9.8)) convergem para zero rapidamente, o que implica ser possível aproximar satisfatoriamente o processo original a partir de um número finito de coeficientes.

Com base neste resultado, conforme em Pfaff (2008, p. 25), a determinação da quantidade de defasagens do VAR se faz através de critérios de informação ou pelo erro final de previsão.

No R a função `VARselect()` os valores de três critérios (AIC, SC e HQ) e o erro final de previsão(FPE).

VARselect(y, lag.max = 10, type = c("const", "trend", "both", "none"), season = NULL, exogen = NULL)

- `y`: objeto contendo as séries do modelo;
- `lag.max`: quantidade máxima de defasagens;
- `type`: parâmetros determinísticos;

- `season`: *dummies* sazonais;
- `exogen`: variáveis exógenas.

Para os dados simulados da equação (9.13), a função parametrizada com 6 defasagens máximas e sem componentes determinísticas retorna:

```
AIC(n)  HQ(n)  SC(n) FPE(n)
     1      1      1      1

criteria
                1          2          3          4          5          6
AIC(n) -0.5302180 -0.5186390 -0.5137012 -0.5055825 -0.4948419 -0.4949713
HQ(n)  -0.5168583 -0.4919197 -0.4736222 -0.4521439 -0.4280437 -0.4148134
SC(n)  -0.4961893 -0.4505818 -0.4116153 -0.3694680 -0.3246988 -0.2907995
FPE(n)  0.5884767  0.5953307  0.5982786  0.6031576  0.6096740  0.6096001
```

A primeira parte da saída indica o número de defasagens ótimo para cada critério, seguido pela tabela com o valor de cada um para cada defasagem. A defasagem ótima é sempre aquela para qual o critério apresenta o menor valor.

Na prática, dado que estamos suprimindo termos determinísticos, essa é toda informação que necessitamos para estimar um processo VAR. Como as funções que analisaremos a seguir necessitam de um processo, estimemos um VAR(1) para os dados que simulamos para a equação (9.13). Isso é possível com a função **VAR**:

VAR(y, p = 1, type = c("consplot(model1.irf, main="“t", "trend“, "both“, "none“), season = NULL, exogen = NULL, lag.max = NULL, ic = c("AIC“, “HQ", “SC", “FPE"))

Além dos mesmos argumentos da função `VARselect`, a função recebe:

- `x`: número de defasagens do modelo;
- `lag.max`: número máximo de defasagens a ser testado pelo argumento de `ic`;
- `ic`: critério de informação usado por `lag.max`.

Estimando o modelo para os simulados (equação (9.13)), obtemos:

```
VAR Estimation Results:
=======================

Estimated coefficients for equation y_1:
========================================
Call:
y_1 = y_1.l1 + y_2.l1

   y_1.l1    y_2.l1
0.6770019 0.3888598


Estimated coefficients for equation y_2:
========================================
Call:
y_2 = y_1.l1 + y_2.l1

   y_1.l1    y_2.l1
0.1983028 0.2227066
```

Observe que os coeficientes estimados estão bem próximos do original. Isso não ocorre simplesmente porque estamos trabalhando com dados simulados, mas sim pela eficiência do MQO para o caso onde os resíduos de cada equação são independentes e homocedásticos, dado que foram gerados a partir de uma distribuição normal.

De fato, a utilização do MQO é geralmente o motivo pelo qual a estacionariedade dos dados é propriedade de interesse na estimação de modelos VAR. Ainda assim, como visto em Enders (2008, p. 270), vários autores argumentam contra a diferenciação dos dados em caso de não estacionariedade, dada a possibilidade de existência de relações de longo prazo entre as variáveis em nível, o que estudaremos quando tratarmos de cointegração, na seção 9.5.

Diagnóstico

Após a estimação do modelo, seguimos naturalmente aos testes de diagnóstico, a fim de verificar se os resíduos satisfazem as hipóteses feitas para o método de estimação, que são em geral ideais para qualquer modelo. Nesse sentido, os testes mais comuns analisam a presença de correlação residual e heterocedasticidade, assim como a estabilidade dos parâmetros. Vamos apresentar algumas implementações desses testes, dos quais os detalhes podem ser vistos em Pfaff (2008, p. 44).

A função `serial.test()` retorna o teste de Portmanteu clássico e ajustado, além do de Breusch-

Godfrey, ambos com hipótese nula de ausência de correlação serial:

```
serial.test(x, type = c("PT.asymptotic", "PT.adjusted", "BG", "ES", ...))
```

- `x`: modelo VAR estimado (i.e. um objeto do tipo `varest`);
- `type`: tipo de teste a ser realizado;

Os testes de heterocedasticidade e normalidade são implementados nas funções `arch.test()` e `normality.test()`, respectivamente. Além disso, a não ser pela possibilidade de definir a quantidade de defasagens em ambos os casos (uni e multivariado) presente no primeiro teste, os argumentos das funções são iguais:

```
arch.test(x, multivariate.only = TRUE, ...)
```

- `x`: modelo VAR estimado;
- `multivariate.only`: codeTRUE retorna apenas a estatística multivariada, enquanto `FALSE` retorna também a de cada equação.

Finalmente, para verificar a estabilidade de um modelo estimado, podemos verificar a condição na equação (9.10), porém calculando os autovalores associados (o que nos permite facilmente avaliar a condição tendo em mente a definição de z) através da função `roots()`, assim como os processos de flutuação empírica disctutidos no Capítulo 4 - que utilizaremos na seção 9.6.

```
roots(x, modulus = TRUE)
```

- `x`: modelo VAR estimado;
- `modulus`: retorna o valor absoluto das raízes, caso contrário retorna tanto a parte real como complexa, caso haja[4].

Para nosso modelo estimado, como sabemos que este é estável por construção, esperamos autovalores inferiores a 1 em valor absoluto:

```
[1] 0.80861328 0.09109521
```

[4] Apesar do nome (raízes em inglês), a função retorna na verdade os autovalores associados à expressão em (9.10).

Uma maneira rápida de verificar essa condição no caso de um sistema com várias variáveis é plotar os valores da função `roots` no cículo unitário (Figura (9.3)).

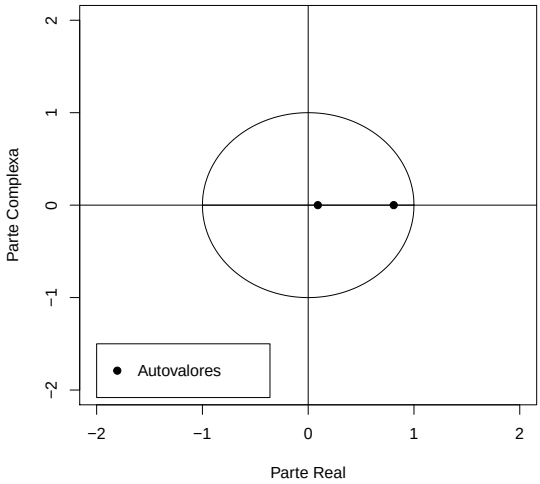


Figura 9.3: Autovalores de (9.13)

Dado que nosso VAR é bivariado, a saída nos apresenta dois autovalores, ambos corroborando a estabilidade do modelo.

Função Impulso-Resposta

Uma ferramenta útil na análise das interações das variáveis presentes em um modelo VAR pode ser obtida com algumas simples manipulações algébricas da representação em médias móveis. Se reescrevermos a equação (9.1) na forma da equação (9.9), obtemos a seguinte representação:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_t = \begin{bmatrix} \overline{y}_1 \\ \overline{y}_2 \end{bmatrix} + \sum_{i=0}^{\infty} B^i \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix}_{t-i} \tag{9.15}$$

Agora, se voltarmos à estrutura do erro puro em 9.1, ou seja:

$$\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix}_{t-i} = \frac{1}{1-\delta_1\delta_2} \begin{bmatrix} 1 & -\delta_1 \\ -\delta_2 & 1 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}_{t-i} \tag{9.16}$$

temos,

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_t = \begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \end{bmatrix} + \frac{1}{1-\delta_1\delta_2} \sum_{i=0}^{\infty} B^i \begin{bmatrix} 1 & -\delta_1 \\ -\delta_2 & 1 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}_{t-i} \tag{9.17}$$

que rearranjamos para encontrar:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_t = \begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \end{bmatrix} + \sum_{i=0}^{\infty} \begin{bmatrix} \phi_{1,1}(i) & \phi_{1,2}(i) \\ \phi_{2,1}(i) & \phi_{2,1}(i) \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}_{t-i} \tag{9.18}$$

onde cada $\phi_{i,j}$ é uma função de impulso-resposta e seu gráfico é uma maneira prática de visualizar o comportamento de choques entre as variáveis do modelo. Em modelos estáveis, esperamos que as respostas a esses choques, que chamamos impulso, convirjam para zero em tempo finito, preferencialmente em poucos lags. Vale observar que $\phi_{i,j}(0)$, a resposta instantânea, é chamado de multiplicador de choque.

O gráfico de $\phi_{i,j}(t)$ é formado para uma sequência de valores de tempo (i.e. $t = 0,1,2,\ldots,n,\ldots$) assumindo na equação (9.18) que $\varepsilon_t = 1$. Dessa maneira a interpretação se inicia assumindo um choque de uma unidade na variável j, com a respectiva resposta na variável i dada pelo comportamento gráfico.

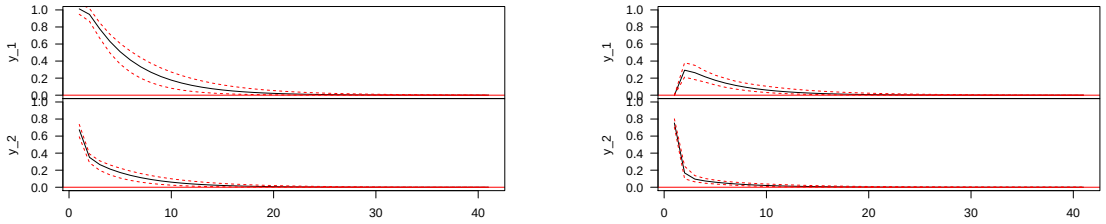
Essa função é implementada sob o nome **irf**:

```
irf(x, impulse = NULL, response = NULL, n.ahead = 10, ortho = TRUE, cumulative = FALSE,
...)
```

- `x`: objeto contendo o modelo VAR estimado;
- `impulse`: vetor contendo variáveis de impulso;
- `response`: vetor contendo variáveis resposta;
- `n.ahead`: define para quantos passos será calculada a resposta ao impulso;
- `ortho`: permite calcular respostas ortogonais - livre de ruídos de outras variáveis - caso os resíduos de diferentes equações do modelo sejam correlacionados.[5];
- `cumulative`: permite calcular o efeito cumulativo do choque.

[5]De fato, essa é a realidade em geral e a solução empregada por esse argumento, que é a mais simples, é a realização de uma decomposição de Cholesky na matriz de variância e covariância dos resíduos.

Como exemplo, geraremos a função para o modelo da equação (9.13) (Figura 9.4).



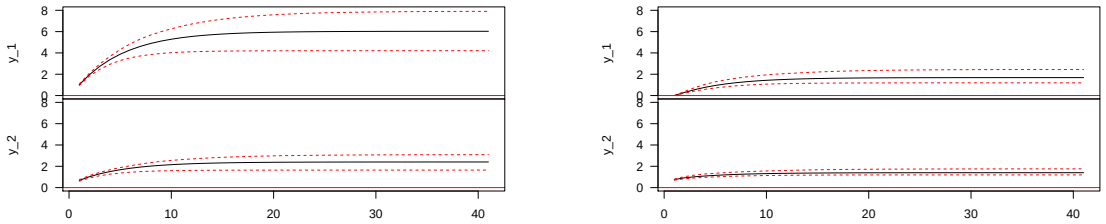
(a) Resposta do choque em y_1 (b) Resposta do choque em y_2

Figura 9.4: FIR da equação (9.13

)

A interpretação é bastante direta: um choque de uma unidade na variável y_1 gera uma perturbação exponencialmente descrescente na variável y_2 (Figura 9.4a), que desaparece completamente por volta do período 18. Já y_2 tem um efeito crecente em y_1 nos primeiros dois períodos (Figura 9.4b), a partir daí caindo e desaparecendo também em torno do período 18.

Outra perspectiva é oferecida quando calculamos o efeito acumulado (Figura 9.5), que nada mais é do que a soma dos choques em cada período. Como as funções convergem a zero[6], o efeito cumulativo também apresenta truncagem. Na literatura, essa função é conhecida como *multiplicador de longo-prazo*.



(a) Resposta Acumulada do choque em y_1 (b) Resposta Acumulada do choque em y_2

Figura 9.5: FIR Acumulada da equação (9.13), com 95% Bootstrap CI e 100 repetições

A interpretação neste caso, é a de que o choque de uma unidade em y_2 gera uma resposta acumulada (efeito de longo prazo) de 2 unidades em y_1.

[6] A convergência da função de impulso-resposta é uma consequência direta da estabilidade do modelo.

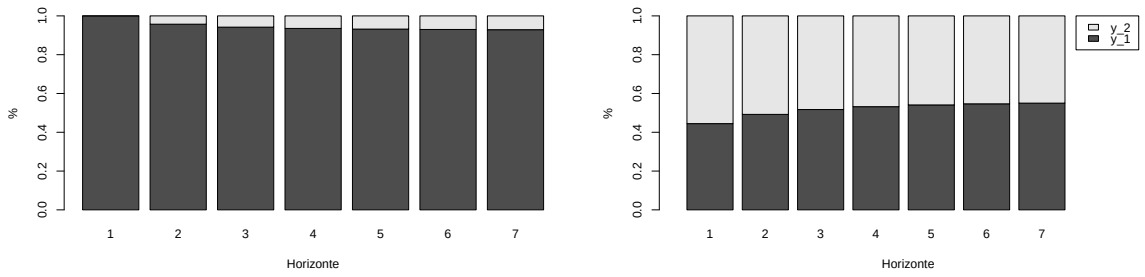
Decomposição de Variância

Outra técnica muito útil é a decomposição de variância dos erros de previsão, implementada na função `fevd()`[7]:

```
fevd(x, n.ahead=10, ...)
```

- `x`: modelo VAR estimado;
- `n.ahead`: quantidade de pontos de previsão.

Na Figura (9.6) temos o resultado da função para modelo da equação (9.13).



(a) FEVD de y_1 (b) FEVD de y_2

Figura 9.6: Decomposição de Variância da equação (9.13
)

Em linhas gerais, a função retorna quanto da variação (percentual) do erro de previsão é atribuído a cada variável para uma sequência de valores no tempo. Ou seja, a variância total dos erros de previsão é decomposta, no nosso caso, em duas componentes, uma relacionada à y_1 e a outra à y_2. Na prática, essa análise nos ajuda a verificar quais variáveis são realmente importantes quando o objetivo é realizar previsões. Quanto maior for a contribuição percentual de uma variável para a variação total de outra, mais importante ela é para realizarmos boas previsões da variável da qual realizamos a decomposição.

Assim, na decomposição da variância da equação (9.13), verificamos que por volta de 50% dos desvios da previsão de y_2 em relação aos valores observados se devem às oscilações de ε_1 (Figura 9.6b), enquanto que a mesma relação para ε_2 e y_1 é inferior a 10% (Figura 9.6a). Tais efeitos eram esperados, dado que a parte estocástica de y_2 depende fortemente dos choques estruturais em y_1, como ressaltamos

[7] Do inglês: *Forecast Error Variance Decomposition*.

quando apresentamos o processo.

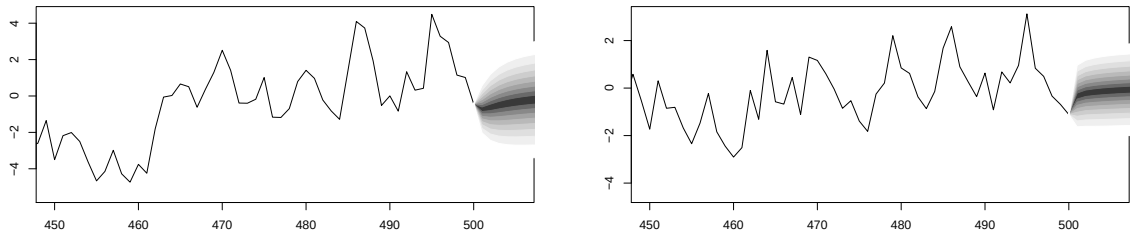
Previsões

Duas maneiras para gerar previsões estão implementadas em `predict()` e `fanchart()`, a primeira retorna os pontos médios dos intervalos de confiança para cada previsão, enquanto a segunda produz os intervalos como áreas sombreadas.

Várias parametrizações estão disponíveis (a maior parte similar às disponíveis em `plot()`). Abaixo segue a definição função `fanchart()`:

```
fanchart(x, colors = NULL, cis = NULL, names = NULL, main = NULL, ylab = NULL, xlab = NULL, col.y = NULL, nc, plot.type = c("multiple", "single"), mar = par("mar"), oma = par("oma"), ...)
```

- `x`: um objeto `predict`, com cada observação e seu intervalo de confiança (I.C.).



(a) Fanchart de y_1 (b) Fanchart de y_2

Figura 9.7: Previsões para os Dados Simulados da equação (9.13
)

As previsões (Figura 9.7) foram geradas para 10 períodos a frente, com intervalo de confiança de 95%, onde a tonalidade mais escura indica maior probabilidade do valor a ser observado se encontrar naquela região. Novamente, dada a estabilidade do modelo, a partir de alguns passos as previsões truncam em um mesmo valor.

Causalidade de Granger

Entre os propósitos comuns na estimação de modelos VAR está a investigação de causalidade entre as variáveis. Uma metodologia muito utilizada para esse fim é a causalidade de Granger (Granger, 1969). Todavia, é importante salientar que essa metodologia se baseia em inferir se valores passados[8] de uma variável auxiliam na previsão de uma outra.

Dessa maneira, o teste não nos informa nada a respeito de causalidade em termos literais, mas, sim, oferece evidências estatísticas de que oscilações passadas de uma variável estão correlacionadas com as de uma outra.

Em um sistema bivariado, testar se y_2 Granger-causa y_1 equivale a estimar,por exemplo, para a equação (9.19), se $\beta_{2,t-i} = 0, \quad \forall i = 1,\ldots,k$.

$$y_{1,t} = \mu_1 + \beta_{1,t-1}y_{1,t-1} + \cdots + \beta_{1,t-p}y_{1,t-p} + \beta_{2,t-1}y_{2,t-1} + \cdots + \beta_{2,t-k}y_{2,t-k} + \varepsilon_{1,t} \tag{9.19}$$

Este teste está implementado na função `causality()`:

```
causality(x, cause = NULL, vcov.=NULL, ....)
```

- `x`: objeto contendo o modelo VAR estimado;
- `cause`: permite especificar a variável de "causa"; se não especificado, utiliza-se a primeira variável do `data frame`;
- `vcov`: permite especificar manualmente a matriz de covariância;

Para o modelo da equação (9.13), o teste para Granger-causalidade de y_1 em y_2 retorna:

```
$Granger

        Granger causality H0: y_1 do not Granger-cause y_2

data:  VAR object model1
F-Test = 34.861, df1 = 1, df2 = 994, p-value = 4.857e-09
```

Como o p-valor é praticamente 0, rejeitamos a hipótese nula, de que observações passadas de y_1

[8] Assim, não vale associar a existência de Granger-causalidade com exogeneidade, dado que esta necessita da ausência de efeitos contemporâneos.

não afetam de maneira estatisticamente significante o valor presente de y_2. Ou seja, y_1 Granger-causa y_2. O que, novamente, é esperado dado a maneira como construimos o processo.

A saída do teste apresenta também uma estatística acerca de relações contemporâneas, que pode ser visto com mais detalhe em Pfaff (2008, p. 36).

VAR Estrutural

Definição

Ainda que os modelos VAR resolvam as limitações da rigidez de *feedback* dos modelos de funções de transferência, tratando todas as variáveis simetricamente, tal abordagem , como enfatiza Enders (2008), é "desprovida de qualquer conteúdo econômico".

O problema se encontra quando se deseja extrair conclusões econômicas a partir do modelo estimado. Como vimos na definição teórica do modelo, os erros da forma reduzida são combinações lineares dos erros da forma irrestrita, o que na prática pode se traduzir em correlações significativas entres os termos de erro de diferentes equações, de maneira que não somos capazes de isolar os choques (e assim analisar relações causais).

Além do problema da interpretação econômica, como na maioria das vezes um modelo reduzido será sobreparametrizado, para fins de predição modelos estruturais são também mais atrativos, dado que as restrições permitem a redução do número de parâmetros do modelo, aumentando os graus de liberdade das estimativas.

Uma maneira simples de abordar essa limitação é estruturar a dinâmica dos resíduos de maneira recursiva, o que se traduz em trasformar a matriz da variância e covariância dos erros numa forma triangular. No sistema bivariado em (9.3), um exemplo desse tipo de restrição é impor $\delta_{2,1} = 0$, o que após pré-multiplicação do sistema por A^{-1}, resulta em uma dinâmica recursiva nos resíduos.

$$\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix}_{t-i} = \begin{bmatrix} 1 & -\delta_{1,1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix}_{t-i} \tag{9.20}$$

Ou seja, o termo de erro do processo $y_{1,t}$ é uma composição dos erros estruturais, o que na litera-

tura costuma-se chamar *choques puros*. Todavia, o ponto importante é que através dessa especificação somos capazes de recuperar o modelo estrutural.

Formalmente, o modelo VAR estrutural difere da forma reduzida se a matriz A na equação (9.21) difere da identidade[9].

$$Ay_t = B_1^* y_{t-1} + B_2^* y_{t-2} + \cdots + B_p^* y_{t-p} + \Phi\varepsilon_t \tag{9.21}$$

onde as matrizes B_i contêm os coeficientes estruturais.

De fato, é condição necessária, mas não suficiente, que em um modelo VAR de n variáveis sejam impostas $\frac{n(n-1)}{2}$ restrições no modelo estrutural para ser possível recuperá-lo a partir da estimação da forma reduzida. Quanto à insuficiência, a falta de uma estrutura triangular é um exemplo de limitação em sistemas com o número adequado de restrições mas que ainda não podem ser identificados.

Dado que a decomposição triangular (também conhecida como *decomposição de Cholesky*), é muito comum em trabalhos empíricos, um problema que surge é a ordenação dos choques. No sistema (9.20), por exemplo, estamos impondo que ε_2 precede ε_1 uma vez que o primeiro entra diretamente na equação do segundo, que apenas afeta y_2 no período seguinte. Esse ordenamento pode não ser tão fácil quando se trata de várias variáveis, o que acaba criando a necessidade de encontrar justificativas teóricas para se impor tais restrições.

Todavia, a importância da ordenação é mais relevante caso a correlação entre os resíduos seja alta e estatisticamente siginificante, uma vez que as variáveis seriam, portanto, mais sensíveis a choques em outras. Entretanto, no caso de sistemas de variáveis econômicas, é comum que muitas apresentem relações contemporâneas, o que impede assumir hipóteses de não-correlação.

[9] É simples ver a consequência disso considerando o modelo na equação (9.3).

Exemplo

Para apresentar o procedimento de estimação, reproduziremos o exemplo em Pfaff (2008, p. 45) e estimaremos um VAR estrutural a partir da simulação do seguinte modelo:

$$\begin{bmatrix} 1 & -0.7 \\ 0.8 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_t = \begin{bmatrix} 0.5 & 0.2 \\ -0.2 & -0.5 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_{t-1} + \begin{bmatrix} -0.3 & -0.7 \\ -0.1 & 0.3 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_{t-2} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix}_t \tag{9.22}$$

A estimação de um VAR estrutural parte da imposição de restrições na matriz de variância-covariância do modelo reduzido, dada por $\sum_{\varepsilon} = A^{-1}BB'A^{-1'}$. Assim, são possíveis restrições para estimar a matriz **A**, quando queremos obter os coeficientes contemporâneos das variáveis; na **B**, quando queremos obter a matriz de variância e covariância dos choques puros ou restrições para obter em ambas[10].

Vamos aqui simular o sistema da equação (9.22), na qual uma matriz do tipo **A**, que está pré-multiplicando seu lado direito define as relações contemporâneas entre y_1 e y_2. Em seguida vamos impor restrições na estrutura dos erros e então partir à estimação.

Para isso, carregamos no R a matriz de coeficientes e geramos a matriz **B**:

```
Apoly <- array(c(1.0,-0.5,0.3,0.8,
                 0.2,0.1,-0.7,-0.2,
                 0.7,1,0.5,-0.3),
               c(3,2,2))
B <- diag(2)
```

em seguida simulamos observações do modelo e as guardamos:

```
svarA <- ARMA(A=Apoly, B=B)
svarsim <- simulate(svarA, sampleT=500,
                    rng=list(seed=c(123456)))

svardat <- matrix(svarsim$output, nrow=500, ncol=2)
colnames(svardat) <- c("y1","y2")
```

Partimos agora ao procedimento para estimar a matriz **A**. Primeiro realizamos a estimação do modelo VAR com os dados simulados anteriormente, e em seguida impomos as restrições na matriz de covariância dos choques puros (argumento `Amat`), que já sabemos quais são dado que definimos essa para simular o processo:

[10] Mais detalhes podem ser vistos em Pfaff (2008, p. 45).

```
varest <- VAR(svardat, p=2, type="none")

Amat <- diag(2)
Amat[2,1] <- NA
Amat[1,2] <- NA
```

Finalmente, utilizamos a função `SVAR()`[11] para estimar o modelo de acordo com as restrições que estabelecemos:

```
SVAR(x, Amat = NULL, Bmat = NULL, ...)
```

- `x`: se refere ao modelo VAR originalmente estimado;
- `Amat` e `Bmat` : recebem as matrizes nas quais se quer realizar as restrições;

```
svar.A <- SVAR(varest, estmethod="direct",
             Amat=Amat, hessian=TRUE)
```

Abaixo, a saída resumida da estimação (sem os detalhes da otimização e resíduos), que esperamos corresponder a matriz $\mathbf{A}$ que definimos na equação (9.22):

```
SVAR Estimation Results:
========================

Estimated A matrix:
       y1      y2
y1 1.0000 -0.6975
y2 0.8571  1.0000
```

Não-estacionariedade e Cointegração

Como visto nas seções anteriores, a proposta original dos modelos VAR era apenas a obtenção de evidências estatísticas acerca da existência de relações entre variáveis de um sistema. Porém, dado o forte interesse de se poder mensurar essas relações preservando seu sentido econômico e os reflexos de choques de uma variável nas outras do sistemas, surgiu a proposta dos modelos estruturais, que permitiam recuperar as relações contemporâneas entre as variáveis.

Ainda que não sugerida pelo autor da metodologia, a estacionariedade das variáveis do sistema foi largamente imposta em diversos trabalhos empíricos, principalmente pela possibilidade de assim se utilizar MQO para estimação do modelo.

[11] A função permite uma série de parametrizações, principalmente porque a estimação é realizada através do método de log-verossimilhança, dos quais os detalhes sugerimos ao leitor consultar diretamente no manual do pacote `vars`.

O problema, como observa Pfaff (2008), é que a maioria das deduções de longo prazo em modelos macroeconômicos surge de modelos com variáveis em nível, o que impõe grande desafio em avaliar resultados empíricos a partir de modelo com variáveis diferenciadas, que é a solução mais comum quando se trabalha com variáveis não-estacionárias.

Por outro lado, modelos lineares estimados por métodos clássicos utilizando variáveis não estacionárias geram problemas ainda mais sérios, como as notáveis regressões espúrias reveladas em Yule (1926).

Engle-Granger

O problema apresentado anteriormente foi uma das motivações do trabalho seminal de Engle & Granger (1987), que possibilitou uma abordagem endógena da não estacionariedade, onde apontam que em alguns casos é possível representar a tendência estocástica presente nas variáveis como uma função linear delas mesmas, o que tem como principal consequência a possibilidade de estimar relações de longo prazo com variáveis não-estacionárias.

Mais precisamente, os autores provaram que é possível que a combinação linear de dois processos, como no sistema 9.23 com ordem de integração maior ou igual a 1 tenha ordem de integração inferior. A definição abaixo (9.23) é uma das contribuições dos autores:

$$\begin{aligned} y_t &= \alpha_1 + \sum_{i=1}^{k} \beta_{1,i} y_{t-1} + \sum_{i=1}^{l} \delta_{1,i} x_t + \varepsilon_{1,t}, \\ x_t &= \alpha_2 + \sum_{i=1}^{p} \beta_{2,i} x_{t-1} + \sum_{i=1}^{q} \delta_{2,i} y_t + \varepsilon_{2,t}, \end{aligned} \tag{9.23}$$

Definição 9.5.1. *Os componentes de um vetor x_t são ditos cointegrados de ordem (d,b) se:* ***i)*** *todos os componentes são integrados de ordem d e* ***ii)*** *existe um vetor não-nulo α tal que $\alpha' x_t$ é cointegrado de ordem $d-b$, com $b>0$. Nesse caso, α é chamado de vetor de cointegração.*

De maneira simplificada, a solução prática do teorema é a inclusão dos resíduos ($\hat{z}_t$) da Regressão (9.24) nas equações do sistema 9.23:

$$y_t = \sum_{i=1}^{k} \alpha_i x_{t,i} + z_t \tag{9.24}$$

Como $\hat{z}_t$ representa as oscilações em relação ao comportamento de equilíbrio de longo prazo das variáveis, em teoria espera-se que sejam estacionários; o que deve ser testado antes de imputá-los no sistema.

O resultado final (sistema 9.25), é chamado Modelo de Correção de Erros (ECM, do inglês *error-correction model*) uma vez que os desvios em $t-1$ são corrigidos em t dada a presença de z_t; do qual o coeficiente estimado, portanto, deve ser sempre negativo.

$$\begin{aligned}\Delta y_t &= \phi_1 + \lambda_1 z_t + \sum_{i=2}^{k} \beta_{1,i}\Delta y_{t-1} + \sum_{i=2}^{l} \delta_{1,i}\Delta x_t + \varepsilon_{1,t},\\ \Delta x_t &= \phi_2 + \lambda_2 z_t + \sum_{i=2}^{p} \beta_{2,i}\Delta x_{t-1} + \sum_{i=2}^{q} \delta_{2,i}\Delta y_t + \varepsilon_{2,t},\end{aligned} \tag{9.25}$$

VECM

Uma clara limitação da abordagem de Engle-Granger surge quando o sistema contém mais de duas variáveis. Se um sistema possui n variáveis, é fácil verificar que se pode obter até $n-1$ relações de cointegração. Assim, a medida que n cresce estimar essas relações fica mais trabalhoso.

Assumindo hipóteses análogas as de Engle-Granger, Campbell & Shiller (1987) definem cointegração de maneira mais generalizada.

Definição 9.5.2. *Um vetor y_t de dimensão n é dito cointegrado se existir pelo menos um vetor β de mesma dimensão tal que $\beta' y_t$ é tendência-estacionário. Se existirem r vetores cointegrantes (dois a dois L.I.[12]) dizemos que y_t é cointegrado com posto[13] de integração r e definimos a matriz $B = (\beta_1, \beta_2 \dots \beta_r)$ com dimensão $(n \times r)$ de vetores cointegrantes, de maneira que os r elementos do vetor $\beta' y_t$ são tendência-estacionários.*

Apresentamos agora o modelo de correção na forma vetorial[14]. Para isso, estabelecemos novamente o modelo em nível como na equação (9.26):

$$y_t = \mu + \sum_{i=1}^{p} \Pi_i y_{t-i} + \Phi D_t + \varepsilon_t, \tag{9.26}$$

[12] Linermente independentes (i.e. $\sum_i^r \alpha_i \beta_i = 0 \iff \alpha_i = 0, \forall i = 1 \dots r$).
[13] O posto se refere a quantidade de linhas linearmente independentes de uma matriz.
[14] Para mais detalhes, sugerimos ao leitor consultar Pfaff (2008)

onde Π_i e D_t são a matriz de coeficientes das variáveis defasadas - sobre a qual inferiremos a existência dos vetores de cointegração - e componentes determinísticos, respectivamente.

A partir dessa estrutura, e da definição 9.5.2, extraímos duas versões de um modelo de correção vetorial[15]. O primeiro (equação (9.27) com os impactos cumulativos de longo prazo, dados pelas matrizes Γ_i e a presença de y_t em nível com $t-p$ defasagens.

$$\Delta y_t = \mu + \sum_{i=1}^{p-1} \Gamma_i \Delta y_{t-i+2} + \Pi_i y_{t-p} + \Phi D_t + \varepsilon_t \tag{9.27a}$$

$$\Gamma_i = -(I - \Pi_1 - \cdots - \Pi_i) \tag{9.27b}$$

$$\Pi_i = -(I - \Pi_1 - \cdots - \Pi_p) \tag{9.27c}$$

A segunda especificação (equação (9.28) por sua vez mede os impactos transitórios, através de uma diferente especificação das matrizes Γ_i e a presença de y_t em nível com apenas uma defasagem.

$$\Delta y_t = \mu + \sum_{i=1}^{p-1} \Gamma_i \Delta y_{t-i+2} + \Pi_i y_{t-1} + \Phi D_t + \varepsilon_t \tag{9.28a}$$

$$\Gamma_i = -(\Pi_{i+1} - \cdots - \Pi_p) \tag{9.28b}$$

$$\Pi_i = -(I - \Pi_1 - \cdots - \Pi_p) \tag{9.28c}$$

Existem três casos possíveis para estrutura de Π, com implicações nas relações de cointegração do modelo. O primeiro é quando o posto de Π é igual a k. Nesses casos existem k combinações lineares possíveis, todas L.I. Isso ocorre se todas as variáveis no vetor y_t são estacionárias, de forma que um VAR em nível é a especificação adequada.

O caso em que o posto é nulo não é interessante, nesse caso somente a solução trivial é estacionária (i.e. não há relação de cointegração), de maneira que o VAR em diferenças é a alternativa natural.

Finalmente, quando o posto da matriz é maior que zero e menor que k existem relações de cointegração não triviais (i.e. Πy_t forma um sistema estacionário). De fato, dado que a matriz Π é

[15] Apesar das diferenças, a análise sobre a matriz Π será a mesma para ambas especificações.

singular[16], podemos escrevê-la como $\alpha\beta'$, ambas de dimensão $(r \times k)$. Na literatura chama-se α de matriz de ajuste, dado que seus elementos determinam a velocidade com a qual os processos voltam a seu comportamento de longo prazo. Como exposto anteriormente, as colunas de β são os vetores de cointegração do sistema.

Método de Johansen

A estimação dos modelos de cointegração vetorial tem como referência o trabalho em Johansen (1988) que desencadeou uma série de outros que consolidaram a metodologia atual.

Sucintamente, os resultados dos trabalhos nos fornecem duas estatísticas, baseadas em maximização de funções de verossimilhança baseadas em regressões auxiliares nas diferenças e valores defasados do vetor y_t.

A primera, acerca da quantidade de vetores de cointegração, proposta em Johansen (1988), chamada estatística do traço[17]:

$$-2\ln(Q) = -T \sum_{i=r+1}^{n} (1 - \hat{\lambda}_i) \tag{9.29}$$

A segunda, de comparação entre dois valores r e $r+1$, proposta em Johansen & Juselius (1990), chamada estatística do máximo autovalor:

$$-2\ln(Q; r|r+1) = -T\ln(1 - \hat{\lambda}_{r+1}) \tag{9.30}$$

Um detalhe importante é a utilização dos valores críticos adequados para cada especificação de componentes determinísticos, tarefa que orienta-se basicamente na comparação com as simulações realizadas pelos autores.

Para exemplificar, simularemos um sistema composto por três processos, no qual dois deles são AR(1) com uma componente linear do terceiro processo, que é um passeio aleatório. Ou seja, as três

[16] Diz-se que uma matriz é singular se não possui inversa, condição equivalente a ter posto menor que sua dimensão.
[17] Às vezes referida como estatística do posto.

variáveis são I(1). O gráfico do processo está na Figura 9.8.

$$\begin{aligned} x_t &= 0.75x_{t-1} + 0.8z_t + \varepsilon_{x,t}, \\ y_t &= -0.3y_{t-1} + -0.3z_t + \varepsilon_{y,t}, \\ z_t &= \sum_{i=0}^{t} \varepsilon_i + \varepsilon_{z,t}, \end{aligned} \tag{9.31}$$

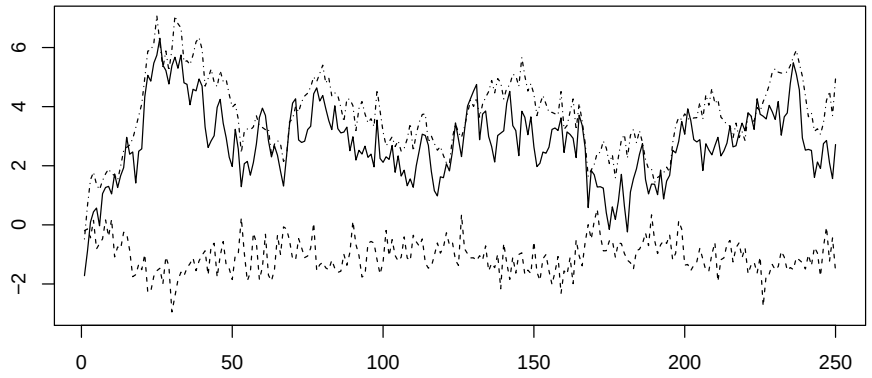


Figura 9.8: Gráfico do sistema (9.31)

A função `ca.jo()` calcula estatísticas de teste para os dois métodos.

ca.jo(x, type = c("eigen", "trace"), ecdet = c("none", "const", "trend"), K = 2, spec=c("longrun", "transitory"), season = NULL, dumvar = NULL)

- `x`: objeto contendo séries a serem testadas;
- `type`: o teste *default* `eigen` retorna resultados para a estatística do máximo autovalor, enquanto `trace` para a estatística do traço;
- `ecdet`: componentes determinísticos;
- `K`: ordem de defasagem do teste;
- `spec`: tipo de estrutura de representação (`longrun` ou `transitory`);
- `season`: frequência das *dummies* sazonais;
- `dumvar`: variáveis *dummy* arbitrárias.

Nos resultados abaixo, $r = 0$ equivale à hipótese nula de não existência de relações de cointegração, enquanto que $r \leq i$ equivale a de i ou menos relações de cointegração.

```
######################
# Johansen-Procedure #
######################

Test type: maximal eigenvalue statistic (lambda max) ,
without linear trend and constant in cointegration

Eigenvalues (lambda):
[1] 3.357938e-01 1.410541e-01 4.949758e-02 7.842278e-19

Values of teststatistic and critical values of test:

          test 10pct  5pct  1pct
r <= 2 |  12.59  7.52  9.24 12.97
r <= 1 |  37.71 13.75 15.67 20.20
r = 0  | 101.47 19.77 22.00 26.81
```

No resultado temos que as estatísticas de teste rejeitam as duas hipóteses nulas iniciais ($r = 0$ e $r \leq 1$) a 1% de significância, enquanto que a existência de duas relações apenas a 5%[18]. Como temos apenas 3 variáveis, haverá no máximo duas relações de cointegração.

Para o teste do traço, obtemos resultados semelhantes:

```
######################
# Johansen-Procedure #
######################

Test type: trace statistic ,
without linear trend and constant in cointegration

Eigenvalues (lambda):
[1] 3.357938e-01 1.410541e-01 4.949758e-02 7.842278e-19

Values of teststatistic and critical values of test:

          test 10pct  5pct  1pct
r <= 2 |  12.59  7.52  9.24 12.97
r <= 1 |  50.30 17.85 19.96 24.60
r = 0  | 151.77 32.00 34.91 41.07
```

Por padrão, a saída da função `ca.jo()` apresenta os vetores de cointegração, porém normalizados para a primeira variável, uma transformação proposta por Johansen, explicada com mais detalhe em

[18] A interpretação natural para cada linha da tabela (de baixo pra cima) é de que se a hipótese nula de i ou menos vetores de cointegração é rejeitada, exitem portanto mais do que i vetores; o que verificamos com a estatística da linha imediatamente superior.

Pfaff (2008, p. 83), que permite identificarmos esses vetores de acordo como aparecem na relação original. Podemos obtê-los através da função `cajorls()`.

```
cajorls(z, r = 1, reg.number = NULL)
```

- `z`: objeto estimado por `ca.jo()`;
- `r`: posto de cointegração ;
- `reg.number`: quantas relações queremos que ele retorne (a ordem é definida pela magnitude do autovalor associado a cada uma).

A saída da função retorna não só os vetores (`beta`), assim como a matriz de coeficientes do modelo (`rlm`):

```
Coefficients:
        y1.d      y2.d      y3.d
ect1    -0.28617  -0.04983   0.03755
ect2    -0.05629  -0.88156  -0.07667
y1.dl1  -0.22437  -0.10328   0.07510
y2.dl1   0.08227  -0.80306   0.01229
y3.dl1   0.07289  -0.05364  -0.16650


$beta
                ect1         ect2
y1.l2      1.0000000 6.938894e-18
y2.l2      0.0000000 1.000000e+00
y3.l2     -0.8333508 2.676756e-0
constant   0.3157532 6.741008e-02
```

Na seção correspondente aos vetores de cointegração (`beta`), temos duas colunas, cada uma com um vetor de cointegração. Como estão normalizados, podemos facilmente encontrar a contrapartida com os coeficientes do processo gerador (equação (9.31).

Na primera coluna, encontramos a correspondência da estrutura AR(1) dada a relação de 1 para 1 entre y_1 e si mesmo. Como y_2 não está na equação de y_1, o coeficiente estimado foi nulo. Já a relação com y_3 foi estimada em -0.8333, bem próxima da verdadeira que é -0.8[19].

[19] Os coeficientes estão com sinais trocados pois na identificação proposta por Johansen estes termos aparecem do lado esquerdo da equação.

Investimento e Confiança Industrial

Realizaremos um pequeno estudo das interrelações de *proxies* para investimento e confiança industrial, a partir de um modelo VAR contendo também *proxies* para o hiato do produto, capacidade instalada e incerteza econômica[20]. Os dados utilizados vão de setembro de 2005 a março de 2016.

Para o investimento, utilizamos como *proxy* a Produção de Bens de Capital, dessazonalizada, divulgada pelo Instituto Brasileiro de Geografia e Estatística (IBGE). O hiato do produto é estimado utilizando o Índice de Atividade Econômica do Banco Central do Brasil (IBC-Br), ao qual aplicamos o filtro HP (Hodrick & Prescott, 1997) para obter as flutuações em torno da tendência de longo prazo.

O índice de confiança utilizado é o Índice de Confiança Industrial (ICI). A capacidade instalada é mensurada a partir do Nível de Utilização da Capacidade Instalada (NUCI). Finalmente, para a incerteza econômica utiliza-se o Indicador de Incerteza da Economia - Brasil (IIE-Br.[21].). Todas essas séries são produzidas e divulgadas pelo Instituto Brasileiro de Economia da FGV (FGV|IBRE).

A priori, seguindo a proposta inicial, não diferenciaremos as variáveis. O próximo passo é verificar o número de defasagens adequadas:

```
$selection
AIC(n)  HQ(n)  SC(n) FPE(n)
     3      2      1      3

$criteria
                1          2          3          4          5          6
AIC(n)   6.208073   5.955130   5.932581   6.163804   6.278435   6.412128
HQ(n)    6.498722   6.487987   6.707645   7.181075   7.537914   7.913815
SC(n)    6.924142   7.267924   7.842099   8.670046   9.381402  10.111820
FPE(n) 496.978658 386.860247 380.567621 485.178840 554.564507 652.248330
                7          8          9         10         11         12
AIC(n)   6.391821   6.465630   6.550572   6.614371   6.575081   6.570238
HQ(n)    8.135714   8.451731   8.778880   9.084887   9.287804   9.525168
SC(n)   10.688236  11.358770  12.040436  12.700960  13.258394  13.850275
FPE(n) 665.504804 757.002843 886.304953 1038.063235 1125.684215 1304.257888
```

Como os dados são mensais, seguimos a sugestão comum na literatura de verificar o número de defasagens a partir de um máximo de 12, contemplando assim o período de um ano. No resultado,

[20] Vale observar que não utilizamos uma *proxy* para custo de capital, pois não consta no estudo original do qual este exemplo se deriva.

[21] Desenvolvido pelo Núcleo de Métodos Estatísticos e Computacionais (FGV|IBRE)

apenas o AIC e FPE convergem, o que pode causar confusão na escolha. Todavia, há de se ter em mente que um dos principais propósitos da inclusão de defasagens em modelos autoregressivos é reduzir a correlação residual do modelo.

Parametrizamos então o modelo com 3 lags e a presença de constante, uma vez que nenhuma variável aparenta apresentar tendências determinísticas, porém possuem períodos condizentes com a presença de *drifts* (Figura 9.9).

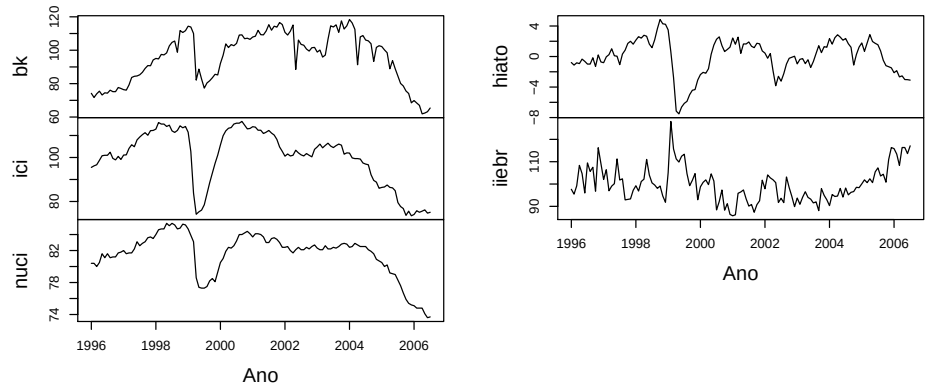


Figura 9.9: Variáveis contidas no modelo

Dado o que o modelo envolve 5 variáveis e três defasagens, a saída da função `summary()` é extensa, de maneira que apresentaremos apenas os resultados para a equação da variável de Produção de Bens de Capital (`bk`), com a omissão de algumas linhas da saída, incluindo as de variáveis não estatisticamente significantes (i.e. p-valor acima de 10%). Vale ressaltar ainda que mesmo que algumas variáveis e defasagens não sejam significantes para essa equação, o são para outras, de maneira que decidimos mantê-las[22]:

[22]De fato, para podermos definir especificações distintas para cada equação, o ideal seria estimar o modelo através do método SUR (Zellner, 1962) (do inglês *seemingly unrelared regressions*), não disponível nos pacotes discutídos nesse capítulo.

```
VAR Estimation Results:

Estimation results for equation bk:
===================================

         Estimate Std. Error t value Pr(>|t|)
bk.l1    -0.51224    0.09661  -5.302 6.21e-07 ***
ici.l1    0.62996    0.22337   2.820  0.00572 **
bk.l2    -0.17931    0.10524  -1.704  0.09132 .
nuci.l2   2.43609    1.05497   2.309  0.02286 *
nuci.l3   2.52674    0.92767   2.724  0.00754 **

---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.817 on 107 degrees of freedom
Multiple R-Squared: 0.3883,        Adjusted R-squared: 0.3025
F-statistic: 4.528 on 15 and 107 DF,  p-value: 1.407e-06
```

Após estimar o modelo verificamos sua estabilidade e realizamos os testes de correlação residual e heterocedasticidade. Quanto à estabilidade, vemos que os autovalores do modelo se encontram todos dentro do círculo unitário (Figura 9.10), de maneira que o processo é estável, o que é corroborado nos testes de flutuação empírica (Figura 9.11).

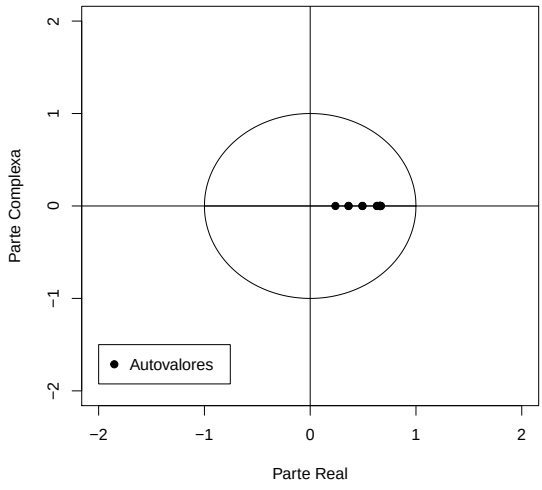


Figura 9.10: Autovalores do modelo estimado

no teste mais adequado ao modelo dado a tamanho da amostra (Portmanteau ajustado), sem ganhos de robustez para valores maiores de defasagem:

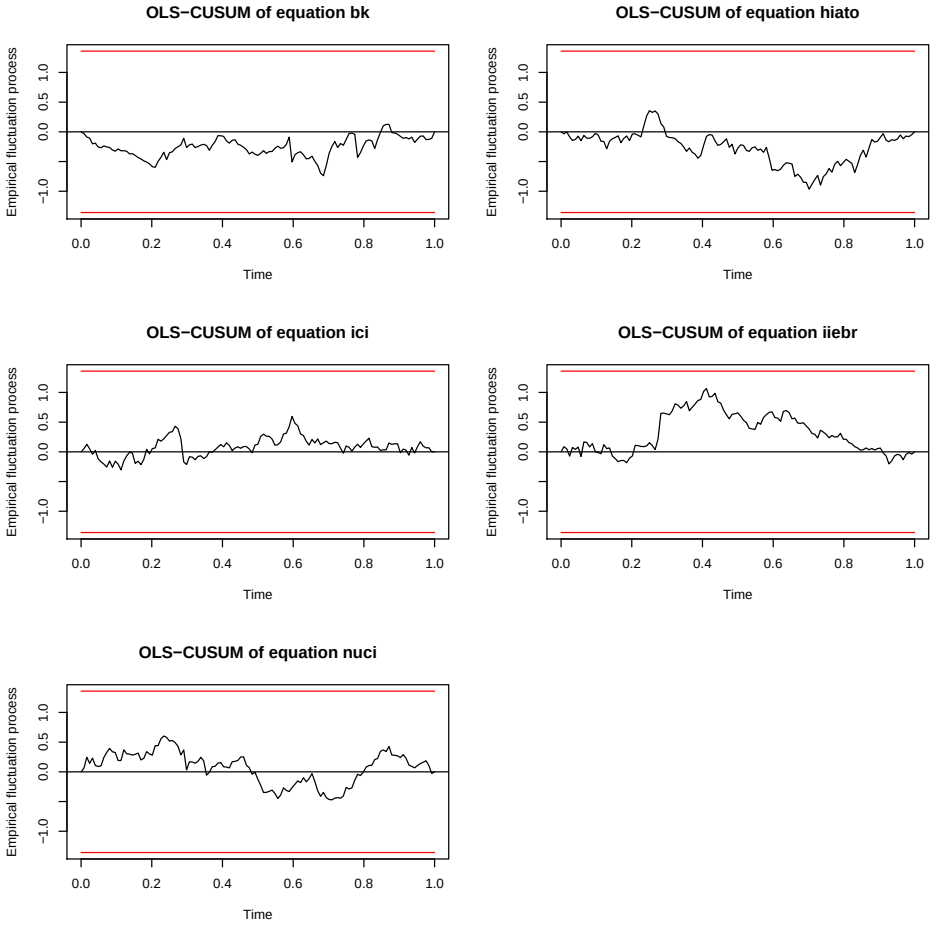


Figura 9.11: Funções de Flutuação Empírica

```
        Portmanteau Test (adjusted)

data:  Residuals of VAR object vetor2
Chi-squared = 356.43, df = 325, p-value = 0.111
```

Realizamos também o teste de heterocedasticidade:

```
        ARCH (multivariate)

data:  Residuals of VAR object var_inv
Chi-squared = 1177.8, df = 1125, p-value = 0.1334
```

Notamos que aambos os testes apresentam p-valores bem mais altos que os adequados para podermos prosseguir, o que não se altera muito quando estimamos o modelo com os dados diferenciados. Todavia, como o propósito deste exemplo é tão apenas apresentar aplicações das metodologias, de

maneira que não realizamos um estudo exaustivo da especificação ideal do modelo. Todavia, consideraremos e estabilidade do modelo e conduziremos o estudo mesmo sem poder considerar ausência de correlação residual e homocedasticidade dos resíduos. Entretanto, alertamos ao leitor que a rejeição dessas hipóteses impede quaiquer inferência quando realizamos trabalhos empíricos.

Assim, partimos à análise das funções de impulso resposta, onde se aplica a identificação recursiva a partir da decomposição de Cholesky para obtenção dos choques puros.

Para a produção de bens de capital (`bk`) (Figura 9.12), obtemos respostas estatisticamente siginificantes para a confiança (`ici`), hiato (`hiato`) e incerteza (`iiebr`), além dele próprio. A função sugere que choques na confiança aumentam a produção de bens de capital, enquanto que choques no hiato e na incerteza reduzem.

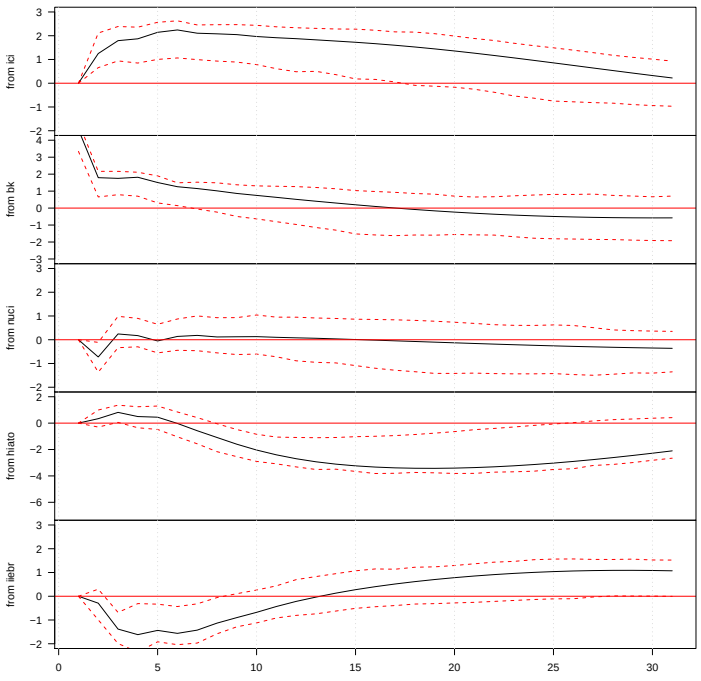


Figura 9.12: Respostas da Produção de Bens de Capital

Para a confiança e a incerteza, a resposta atinge maior valor entre o 5º e 7º mês do choque, a partir de onde convergem a zero. Para o hiato a resposta só parece ser significativa a partir do 8º mês, durando até o 26º, o que pode ocorrer tanto pelo tempo de ajuste necessário entre os pedidos e a produção quanto por uma possível espera dos agentes associados à demanda em verificar se o desvio

do produto potencial é de fato um sinal de aquecimento fora de equilíbrio da economia, o que requer revisões para redução de demanda.

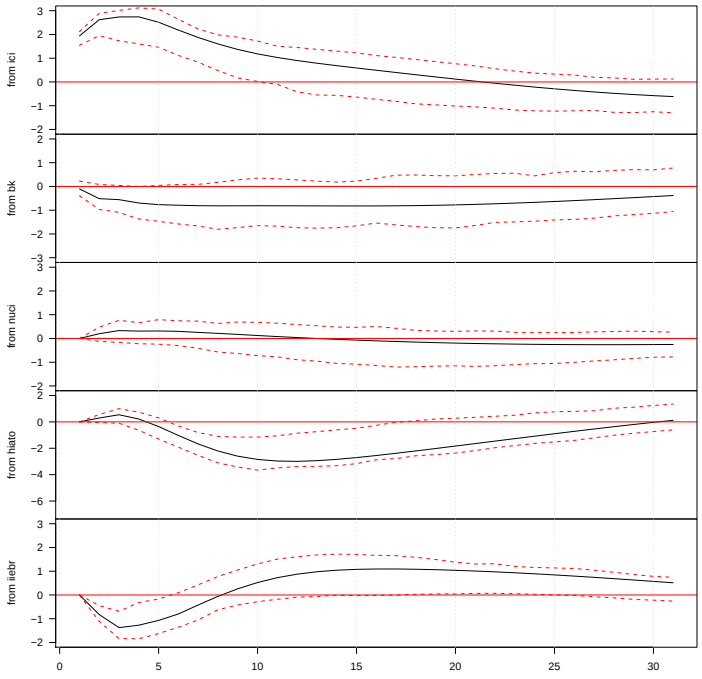


Figura 9.13: Respostas do Índice de Confiança Industrial

Para confiança industrial, Figura 9.13, as respostas siginificativas ocorrem para o hiato e a incerteza. As funções sugerem que choques em ambas as variáveis diminuem a confiança do setor industrial. Porém, enquanto a resposta máxima da incerteza ocorre em torno do 5º mês, é a partir daí que se faz sentir a resposta de desvios do hiato, o que deve possuir explicação similar à da produção de bens de capital.

Com base nesses resultados, vemos que confiança e incerteza são relevantes no estudo do comportamento da formação bruta de capital fixo. Ainda mais, vemos como os efeitos se concentram nos períodos imediatamente subsequentes aos choques, com picos antes dos 6 meses, se dissipando em menos de um ano. Ou seja, essas variáveis não aparentam explicar a dinâmica de longo prazo da produção de bens de capitais.

Considerações Finais

Neste capítulo fomos apresentados ao modelos VAR e suas principais propriedades, como estabilidade e representação em médias móveis, que sustentam resultados importantes tanto para estimação quanto análise do modelo. Entre as ferramentas para análise, discutimos as funções de impulso-resposta, a decomposição de erros de previsão e a causalidade de Granger, que podem oferecer importantes informações quanto ao comportamento estrutural do modelo. Dado que todas essas ferramentas são melhor aproveitadas quando somos capazes de identificar os parâmetros estruturais do modelo, apresentamos o que é e como obter os parâmetros de um VAR estrutural (SVAR). Considerando as relações de longo prazo presentes em muitos modelos envolvendo séries não-estacionárias, discutimos os modelos vetoriais de correção de erro (VECM), aos quais se aplicam as mesmas ferramentas de análise de um VAR comum. Finalmente, fizemos um breve estudo envolvendo *proxies* para investimento e confiança industrial no Brasil, exemplificando as principais técnicas apresentadas.

Referências Bibliográficas

Akaike, H. (1973). Information Theory and an Extension of the Maximum Likelihood Principle. *2nd International Symposium on Information Theory*, (pp. 267–281). pages 132, 164

Bai, J., & Perron, P. (2003). Computation and Analysis of Multiple Structural Change Models. *Journal of Applied Econometrics*, *18*(1), 1–22. pages 110, 112

Box, G. E. P., & Jenkins, G. M. (1970). Time Series Analysis: Forecasting and Control. pages 116, 117, 129, 160, 161, 166

Breusch, T. S. (1978). Testing for Autocorrelation in Dynamic Linear Models*. *Australian Economic Papers*, *17*(31), 334–355. pages 174

Brown, R. L., Durbin, J., & Evans, J. M. (1975). Techniques for Testing the Constancy of Regression Relationships over Time. *Journal of the Royal Statistical Society. Series B (Methodological)*, *37*(2), 149–192. pages 108

Calcagno, V. (2013). **glmulti***: Model Selection and Multimodel Inference Made Easy*. R package version 1.0.7.
URL https://CRAN.R-project.org/package=glmulti pages 184

Campbell, J. Y., & Shiller, R. J. (1987). Cointegration and Tests of Present Value Models. *Journal of Political Economy*, *95*(5), 1062–1088. pages 218

Chan, K.-S., & Ripley, B. (2012). **TSA***: Time Series Analysis*. R package version 1.01.
URL https://CRAN.R-project.org/package=TSA pages 118

Chang, I., Tiao, G. C., & Chen, C. (1988). Estimation of Time Series Parameters in the Presence of Outliers. *Technometrics*, *30*(2), 193–204. pages 133

Chow, G. C. (1960). Tests of Equality between Sets of Coefficients in Two Linear Regressions. *Econometrica: Journal of the Econometric Society*, (pp. 591–605). pages 108

Cochrane, D., & Orcutt, G. H. (1949). Application of Least Squares Regression to Relationships Containing Auto-correlated Error Terms. *Journal of the American Statistical Association*, *44*(245), 32–61. pages 177

Cowpertwait, P. S., & Metcalfe, A. V. (2009). *Introductory Time Series with R*. Springer Science & Business Media. pages 95, 97, 98

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the Estimators for Autoregressive Time Series with a Unit Root. *Journal of the American Statistical Association*, *74*(366a), 427–431. pages 123

Dolado, J. J., Jenkinson, T., & Sosvilla-Rivero, S. (1990). Cointegration and Unit Roots. *Journal of Economic Surveys*, *4*(3), 249–273. pages 103

Dragulescu, A. A. (2014). **xlsx***: Read, Write, Format Excel 2007 and Excel 97/2000/XP/2003 Files*. R package version 0.5.7.
URL https://CRAN.R-project.org/package=xlsx pages 19, 20, 118

Durbin, J. (1970). Testing for Serial Correlation in Least-squares Regression when some of the Regressors are Lagged Dependent Variables. *Econometrica: Journal of the Econometric Society*, (pp. 410–421). pages 175

Durbin, J., & Watson, G. S. (1950). Testing for Serial Correlation in Least Squares Regression. I. *Biometrika*, *37*(3-4), 409–428. pages 134, 174

Durbin, J., & Watson, G. S. (1951). Testing for Serial Correlation in Least Squares Regression. II. *Biometrika*, *38*(1/2), 159–177. pages 134, 174

Durbin, J., & Watson, G. S. (1971). Testing for Serial Correlation in Least Squares Regression. III. *Biometrika*, *58*(1), 1–19. pages 174

Elliott, G., Rothenberg, T., & Stock, J. (1996). Efficient Test for an Autoregressive Unit Root. *Econometrica*, *64*(4), 813–836. pages 123

Enders, W. (2008). *Applied Econometric Time Series.*, vol. 4. John Wiley & Sons Inc. pages 94, 95, 96, 102, 112, 113, 188, 196, 197, 199, 205, 213

Engle, R. F. (1984). Wald, Likelihood Ratio, and Lagrange Multiplier Tests in Econometrics. *Handbook of Econometrics*, *2*, 775–826. pages 135

Engle, R. F., & Granger, C. W. (1987). Co-Integration and Error Correction: Representation, Estimation, and Testing. *Econometrica*, *55*(2), 251–276. pages 196, 217

Ferreira, P. C., Gondin, J. L., & de Mattos, D. M. (2015). Métodos de Ajuste Sazonal para Séries de Business Tendency: um Estudo de Caso para a Sondagem da Indústria Utilizando o Método X13-ARIMASEATS. *FGV | IBRE*. pages 141

Findley, D. F., Monsell, B. C., Bell, W. R., Otto, M. C., & Chen, B.-C. (1998). New Capabilities and Methods of the X-12-ARIMA Seasonal-adjustment Program. *Journal of Business & Economic Statistics*, *16*(2), 127–152. pages 140

Fok, D., Franses, P. H., & Paap, R. (2005). Performance of Seasonal Adjustment Procedures: Simulation and Empirical Results. *Econometric Institute Report*. pages 140

Fox, J., & Weisberg, S. (2011). *An R Companion to Applied Regression*. Thousand Oaks CA: Sage, 2 ed.
URL http://socserv.socsci.mcmaster.ca/jfox/Books/Companion pages 176

French, M. W. (2001). Estimating Changes in Trend Growth of Total Factor Productivity: Kalman and H-P Filters versus a Markov-Switching Framework. FED - Finance and Economics Discussion Series - Working Paper No. 2001-44. pages 113

Gavrilov, I., & Pusev, R. (2014). **normtest***: Tests for Normality*. R package version 1.1.
URL https://CRAN.R-project.org/package=normtest pages 118

Godfrey, L. G. (1978). Testing Against General Autoregressive and Moving Average Error Models when the Regressors Include Lagged Dependent Variables. *Econometrica: Journal of the Econometric Society*, (pp. 1293–1301). pages 174

Gómez, V., & Maravall, A. (1998). *Guide for Using the Programs TRAMO and SEATS*. Banco de España. pages 140

Granger, C. W. (1969). Investigating Causal Relations by Econometric Models and Cross-spectral Methods. *Econometrica: Journal of the Econometric Society*, *37*(3), 424–438. pages 212

Granger, C. W., & Newbold, P. (1974). Spurious Regressions in Econometrics. *Journal of econometrics*, *2*(2), 111–120. pages 102, 185

Granger, C. W., & Newbold, P. (1976). Forecasting Transformed Series. *Journal of the Royal Statistical Society. Series B (Methodological)*, (pp. 189–203). pages 128

Graves, S. (2014). **FinTS***: Companion to Tsay (2005) Analysis of Financial Time Series*. URL http://cran.r-project.org/package=FinTS pages

Greene, W. H. (2003). *Econometric Analysis*. Prentice Hall, Upper Saddle River, 5 ed. pages 183

Hamilton, J. D. (1994). *Time Series Analysis*. Princeton University Press. pages 94, 123, 127, 163

Harvey, A., & Shepard, N. (1993). Structural Time Series Models. *Elsevier Science Publishers B.V.*. pages 140

Hildreth, C., & Lu, J. (1960). Demand Relations with Auto-correlated Disturbances, Michigan State University, Agricultural Experimental Station. *Technical Bulletin*, *276*, 185. pages 178

Hodrick, R., & Prescott, E. (1997). Postwar Business Cycles. *Jornal of Money, Credit and Banking*. pages 112, 224

Hungarian Central Statistical Office (2007). Seasonal Adjustment Methods and Pratices. *European Commission Grant*. pages 140

Hylleberg, S., Engle, R. F., Granger, C. W., & Yoo, B. S. (1990). Seasonal Integration and Cointegration. *Journal of econometrics*, *44*(1), 215–238. pages 126

Hyndman, R. J. (2015). **forecast***: Forecasting Functions for Time Series and Linear Models*. R package version 6.2.
URL http://github.com/robjhyndman/forecast pages 70, 84

Hyndman, R. J., Koehler, A. B., Ord, J. K., & Snyder, R. D. (2008). *Forecasting with Exponential Smoothing: The State Space Approach*. Springer Science & Business Media. pages 70

Hyndman, R. J., Razbash, S., & Schmidt, D. (2012). Forecasting Functions for Time Series and Linear Models. *R package version (http://cran. r-project. org/web/packages/forec ast/)*. pages 70, 118

IBGE (2015a). Pesquisa de Orçamentos Familiares.

URL http://www.ibge.gov.br/home/xml/pof_2008_2009.shtm pages 21, 29

IBGE (2015b). Pesquisa Industrial Mensal - Produção Física.

URL http://www.ibge.gov.br/home/estatistica/indicadores/industria/pimpfbr/ pages 144

IBGE (2015c). Pesquisa Industrial Mensal Produção Física - Brasil - Notas Metodológicas.

URL http://www.ibge.gov.br/home/estatistica/indicadores/industria/pimpfbr/notas_metodologicas.shtm pages 151

IPEA (2015). Instituto de pesquisa econômica aplicada.

URL http://www.ipea.gov.br/portal/index.php?option=com_content&view=article&id=21971&catid=10&Itemid=9 pages 64

Jarque, C. M., & Bera, A. K. (1980). Efficient Tests for Normality, Homoscedasticity and Serial Independence of Regression Residuals. *Economics letters*, *6*(3), 255–259. pages 135

Johansen, S. (1988). Statistical Analysis of Cointegration Vectors. *Journal of Economic Dynamics and Control*, *12*(2), 231–254. pages 196, 220

Johansen, S., & Juselius, K. (1990). Maximum Likelihood Estimation and Inference on Cointegration—with Applications to the Demand for Money. *Oxford Bulletin of Economics and Statistics*, *52*(2), 169–210. pages 220

Keele, L., & De Boef, S. (2004). Not Just for Cointegration: Error Correction Models with Stationary Data. *Documento de Trabajo. Departamento de Política y Relaciones Internacionales, Nuffield College y Oxford University*. pages 186

Kleiber, C., & Zeileis, A. (2008). *Applied Econometrics with R*. Springer Science & Business Media. pages 112

Komsta, L., & Novomestky, F. (2015). Moments, Cumulants, Skewness, Kurtosis and Related Tests. pages 44

Koopman, S. J., Harvey, A., Doornik, J., & Shepard, N. (2009). Structural Time Series Analyser, Modeler and Predictor. *Timberlake Consultants*. pages 140

Kwiatkowski, D., Phillips, P. C., Schmidt, P., & Shin, Y. (1992). Testing the Null Hypothesis of Stationarity Against the Alternative of a Unit Root: How Sure Are We that Economic Time Series Have a Unit Root? *Journal of econometrics*, *54*(1), 159–178. pages 104, 123

Li, Z. (2010). **dcv***: Conventional Cross-validation Statistics for Climate-growth Model*. R package version 0.1.1.
URL https://CRAN.R-project.org/package=dcv pages 176

Livsey, J., Pang, O., & McElroy, T. (2014). Effect of Trading Day Regressors on Seasonal Adjustment of Growth Rates. *RESEARCH REPORT SERIES, US Census Bureau*.
URL https://www.census.gov/srd/papers/pdf/rrs2014-09.pdf pages 141

Ljung, G. M., & Box, G. E. (1978). On a Measure of Lack of Fit in Time Series Models. *Biometrika*, *65*(2), 297–303. pages 143

Lütkepohl, H. (2005). *New Introduction to Multiple Time Series Analysis*, vol. 3. Springer Science & Business Media. pages 196, 199, 201

MacKinnon, J. G. (1996). Numerical Distribution Functions for Unit Root and Cointegration Tests. *Journal of Applied Econometrics*, *11*, 601–618. pages 123

Mohr, F. (2015). **prais***: Prais-Winsten Estimation Procedure for AR(1) Serial Correlation*. R package version 0.1.1.
URL https://CRAN.R-project.org/package=prais pages 178

Morettin, P. A., & Toloi, C. (2006). *Análise de Séries Temporais*. 2. ed. São Paulo: Editora Blucher. pages 70, 81, 83

Newey, W. K., & West, K. D. (1987). A Simple, Positive Semi-Definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix. *Econometrica*, *55*(3), 703–708. pages 179

Pfaff, B. (2008). *Analysis of Integrated and Cointegrated Time Series with R*. Springer Science & Business Media. pages 196, 201, 203, 205, 213, 215, 217, 218, 223

Pfaff, B., Stigler, M., & Pfaff, M. B. (2013). Package 'vars'. pages 196

Pfaff, B., Zivot, E., & Stigler, M. (2016). **urca***: Unit Root and Cointegration Tests for Time Series Data*.
URL https://cran.r-project.org/web/packages/urca/index.html pages 118, 124, 188

Phillips, P. C., & Perron, P. (1988). Testing for a Unit Root in Time Series Regression. *Biometrika*, *75*(2), 335–346. pages 105, 123, 163

Pindyck, R. S., & Rubinfeld, D. L. (1998). *Econometric Models and Economic Forecasts*, vol. 4. Irwin/McGraw-Hill Boston. pages 174

Plosser, C. I. (1979). A time series analysis of seasonality in econometric models. *The National Bureau of Economic Research*. pages 140

Prais, S. J., & Winsten, C. B. (1954). Trend Estimators and Serial Correlation. Tech. rep., Cowles Commission Discussion Paper No. 383 (Chicago). pages 178

Rasmussen, R. (2004). On Time Series Data and Optimal Parameters. *The International Journal of Management Science*. pages 140

R Core Team (2015a). **graphics***: The R Graphics Package*. pages 38

R Core Team (2015b). **stats***: The R Stats Package*. pages

Sax, C. (2015a). Github christoph sax.
URL https://github.com/christophsax pages 141

Sax, C. (2015b). **seasonal***: R Interface to X-13-ARIMA-SEATS*.
URL http://cran.r-project.org/package=seasonal pages 146

Schwarz, G., et al. (1978). Estimating the Dimension of a Model. *The annals of statistics*, *6*(2), 461–464. pages 132, 164

Shapiro, S. S., & Wilk, M. B. (1965). An Analysis of Variance Test for Normality (Complete Samples). *Biometrika*, *52*(3/4), 591–611. pages 143

Shiskin, J., Young, A. H., & Musgrave, J. C. (1967). The X-11 Variant of the Census Method II Seasonal Adjustment Program. *Bureau of the Census*, *52*. pages 140

Sims, C. A. (1980). Macroeconomics and Reality. *Econometrica*, *48*(1), 1–48. pages 196

Spada, S., Quartagno, M., & Tamburini, M. (2012). **orcutt***: Estimate Procedure in Case of First Order Autocorrelation*. R package version 1.1.
URL https://CRAN.R-project.org/package=orcutt pages 178

Trapletti, A., & Hornik, K. (2015). **tseries***: Time Series Analysis and Computational Finance*. R package version 0.10-34.
URL http://CRAN.R-project.org/package=tseries pages

Tsay, R. S. (1988). Outliers, Level Shifts, and Variance Changes in Time Series. *Journal of forecasting*, *7*(1), 1–20. pages 133

Ulrich, J., Ulrich, M. J., & RUnit, S. (2013). Package 'ttr'. pages 70

U.S. Census Bureau (2015). X13-arima-seats reference manual acessible html output version.
URL https://www.census.gov/ts/x13as/docX13AS.pdf pages 140, 141, 143, 144, 146, 148

Verzani, J. (2015). **UsingR***: Data Sets, Etc. for the Text "Using R for Introductory Statistics", Second Edition*. R package version 2.0-5.
URL https://CRAN.R-project.org/package=UsingR pages 40

Walker, A. (2015). **openxlsx***: Read, Write and Edit XLSX Files*. R package version 3.0.0.
URL https://CRAN.R-project.org/package=openxlsx pages 19

Wickham, H. (2016). **readxl***: Read Excel Files*. R package version 0.1.1.
URL https://CRAN.R-project.org/package=readxl pages 19

Wickham, H., & Chang, W. (2015). **ggplot2***: An Implementation of the Grammar of Graphics*.
URL http://cran.r-project.org/package=ggplot2 pages 70

Yule, G. U. (1926). Why Do We Sometimes Get Nonsense-Correlations Between Time-Series?–A Study in Sampling and the Nature of Time-Series. *Journal of the Royal Statistical Society*, *89*(1), 1–63. pages 217

Zeileis, A. (2004). Econometric Computing with HC and HAC Covariance Matrix Estimators. *Journal of Statistical Software*, *11*(10), 1–17.
URL http://www.jstatsoft.org/v11/i10/ pages 179

Zeileis, A. (2016). **dynlm***: Dynamic Linear Regression*. R package version 0.3-5.
URL http://CRAN.R-project.org/package=dynlm pages 183

Zeileis, A., & Hothorn, T. (2002). Diagnostic Checking in Regression Relationships. *R News*, *2*(3), 7–10. pages 118, 175, 176

Zeileis, A., Leisch, F., Hornik, K., & Kleiber, C. (2001). strucchange. an R package for testing for structural change in linear regression models. pages 95, 108, 109

Zellner, A. (1962). An efficient method of estimating seemingly unrelated regressions and tests for aggregation bias. *Journal of the American Statistical Association*, *57*(298), 348–368. pages 225

Zellner, A. (1979). Front Matter to Seasonal Analysis of Economic Time Series. *The National Bureau of Economic Research*. pages 140

Zivot, E., & Andrews, D. W. K. (1999). Further Evidence on the Great Crash, the Oil-Price Shock, and the Unit-Root Hypothesis. *Journal of Business and Economic Statistics*. pages 106