DSA – Formação Cientista de Dados

3. Big Data Analytics com R e Microsoft Azure Machine Learning.

3.1. Introdução

Breve História da Linguagem R:

- 1993 – Projeto de pesquisa em Auckland, na Nova Zelândia
- 1995 – R liberado como projeto open-source
- 1997 – Formado o grupo R-Core
- 2000 – Liberada a versão 1.0.0 do R
- 2003 – Criação da R Foundation
- 2004 – Primeira conferência internacional de usuários em Vienna
- 2015 – Formado o R Consortium (com participação da IBM e Microsoft)

Por que cientistas de dados usam R?

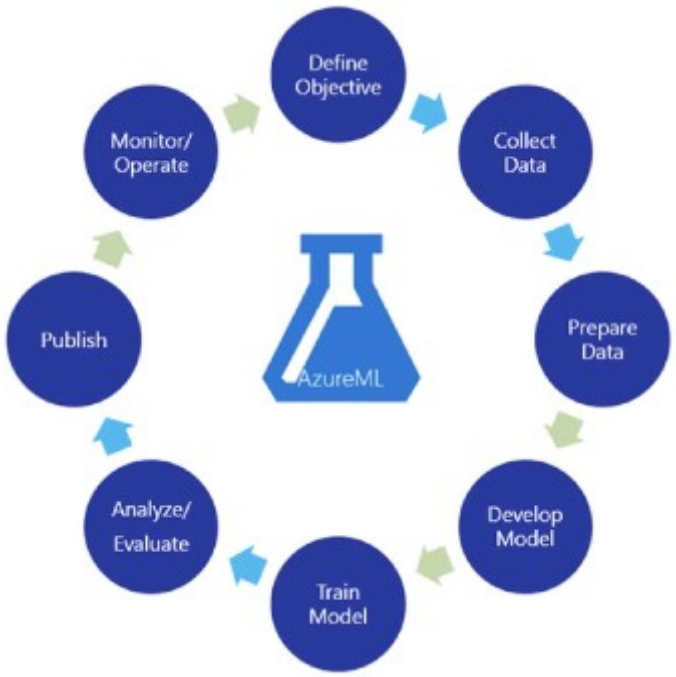
- Linguagem Open Source
- Versatilidade
    - Extração de Dados
    - Limpeza de Dados
    - Carregamento e Transformação de Dados
    - Análise Estatística
    - Modelagem Preditiva
    - Machine Learning
    - Visualização de Dados

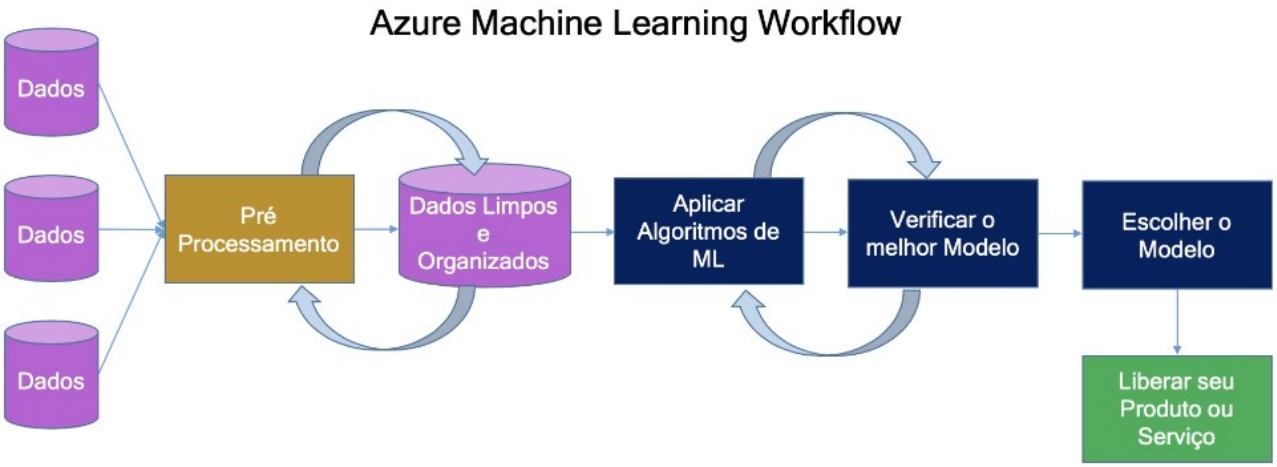
Vantagens e Desvantagens da linguagem R

Vantagens	Desvantagens
• Grande variedade de pacotes disponíveis • Flexibilidade e Rapidez	• Não há interface gráfica (tudo é feito por linha de comando) • Limitações no uso de memória principalmente com datasets muito

• Análise Estatística	grandes

Azure Machine Learning Workflow





Legenda: DADOS > PRÉ-PROCESSAMENTO <>DADOS LIMPOS E ORGANIZADOS > APLICAR ALGORITMOS DE ML <> VERIFICAR O MELHOR MODELO > ESCOLHER O MODELO > LIBERAR SEU PRODUTO OU SERVIÇO

Dados X Informação X Conhecimento X Inteligência

As pessoas trabalham com informações, mas a tecnologia armazena dados.

Inteligência – capacidade de resolver problemas, usando conhecimento, através das informações disponíveis.

Big Data Analytics

Extrair conhecimento a partir dos dados (muitos dados).

Machine Learning

Ensinar algoritmos a usar inteligência, ou seja, usar o conhecimento para resolver problemas.

Isso é feito coletando-se dados, pré processando esses dados e transformando-os em informação, alimentando-se algoritmos (conjunto de operações estatísticas e matemáticas) que aprendem a relação dessa informação, adquirindo conhecimento. Dessa forma, a partir do conhecimento adquirido, é possível resolver novos problemas de negócio a partir de novos dados.

A importância do Big Data Analytics

Tempestade Perfeita

- Crescimento exponencial do volume de dados
- Preço baixo do armazenamento de dados
- Alta capacidade de processamento dos computadores

Os dados são combustível para a Análise Preditiva (essência do que é feito em Data Science).

E podemos ver a Análise Preditiva em ação todos os dias:

- Filtros de Spam dos e-mails
- Programas de concessão de financiamento para casa própria
- Padrões de reconhecimento (voz, imagem)
- Seguro de Vida
- Detecção de Fraudes em Cartões de Crédito
- Controle de Voos
- Engines de Busca na Internet

Tudo isso é alimentado por dados, muitos dados (Big Data).

Quando aplicamos Data Science para análise de Big Data temos o que é chamado de Big Data Analytics e sua importância é cada vez maior no mundo atual.

Data Science e a Evolução dos Sistemas Analíticos

Legenda: Analysis, Structure, Algorithm, Process, Programming, Solving, Knowledge.

Estamos vivendo a era da explosão dos dados

bit byte > kilobyte > megabyte gigabyte terabyte > petabyte exabyte zettabyte yottabyte

Em 2014 a humanidade acumulou em dados o equivalente a 1.8 Zettabyte.

Tipos de Análise Realizados em Ciência de Dados:

- Descritiva: O que aconteceu? (BI Tradicional)
- Diagnóstica: Por que isso aconteceu? (BI Tradicional)
- Preditiva: O que acontecerá?
- Prescritiva: O que deve ser feito?

Data Science e Inteligência Artificial – Há Diferença?

Inteligência Artificial é um conjunto de técnicas para reproduzir um comportamento inteligente. Dentre essas técnicas, se encontra o Machine Learning que, dentre várias, possui a técnica de Deep Learning.

Data Science é uma área multidisciplinar que envolve Estatística, Matemática, Machine Learning, Programação, Banco de Dados, Conhecimento de Negócios, Técnicas de Apresentação e Visualização de Dados.

Aplicamos Data Science para analisar dados, construir modelos preditivos e resolver problemas de negócio. Ou seja, aplicamos Data Science quando estamos trabalhando com Machine Learning e Inteligência Artificial.

Instalando o R, RStudio e RTools no Windows 10

Dicas:

- Não utilize nome de usuário, no seu Sistema Operacional, com espaço ou acento.
- Instale os softwares em inglês (previna possíveis bugs)

Para fazer o download do R (interpretador):

- https://cran.r-project.org
- Clique em “Download R for Windows”
- Clique em “base”
- Obs.: no curso foi utilizado a versão “Download R 3.5.2 for Windows”
- Instale num diretório onde não haja espaços no nome:
  - Ex.: C:\R\R-3.5.2
- Selecione todos os itens/componentes
  - Core Files, 32-bit Files, 64-bit Files e Message translations
- Em “Startup options”, selecione “No (accpet defaults)
- Manter “R” no atalho que será colocado no menu iniciar
- Em “Select Additional Tasks”, selecione “Save version number in registry” e “Associate R with .Rdata files”
- Prossiga com a instalação

Para fazer o download do RTools (exclusivo para usuários windows):

- https://cran.r-project.org
- Clique em “Download R for Windows”
- Clique em “Rtools”
- Obs.: no curso foi utilizado a versão “Rtools35.exe”
- Instale num diretório onde não haja espaços no nome:
  - Ex.: C:\Rtools
- Em “Select Components”, mantenha as opções recomendadas
- Em “Select Additional Tasks”, selecione “Add rtools to system PATH” e “Save version information to registry”
- Prossiga com a instalação
- Após instalação, caso necessite adicionar o PATH em seguida, crie um arquivo de nome .Renviron, na pasta Documents:
  - Abra o R e digite a linha:
    - writeLines('PATH="${RTOOLS40_HOME}\\usr\\bin;${D:/Desenvolvedor/CienciaDeDados/Dev/Rtools/rtools40}"', con = "~/.Renviron")
  - Reinicie o R e confirme se deu certo, digitando a linha:
    - Sys.which("make")
      - Deverá aparecer o caminho da instalação do Rtools

Para fazer o download do RStudio:

- https://www.rstudio.com
- Na parte superior, clique em “Products”, depois em “RStudio”
- Clique na versão “Desktop” (pois há também a “Server”)
- Será mostrado a versão Free e a paga. Clique no botão para fazer o download da versão Free
- Outra vez, procure o botão para opção Free
- Em “Installers for Supported Platforms”, selecione a versão para instalação
- Obs.: no curso foi utilizado a versão “RStudio 1.1.463 Windows Vista/7/8/10”
- Instale num diretório onde não haja espaços no nome:
  - Ex.: C:\RStudio
- Prossiga com a instalação

3.2. Fundamentos da Linguagem R

Introdução

Site Oficial: https://www.r-project.org

A linguagem R é um ambiente de software gratuito para computação estatística e gráficos.

RStudio

O RStudio é uma IDE para linguagem R.

- Abra um painel para escrever seus scripts em R:
  - New File > R Script
- Altere a organização dos painéis:
  - Tools > Global Options > Pane Layout
  - No primeiro quadrante, selecione Console
  - No segundo, selecione Source (onde se escreve os scripts)
  - Mantenha o 3 e o 4, ou altere como quiser
  - Apply > OK
- Mude o interpretador dos scripts (testar outras versões), sem a necessidade de excluir a versão anterior. Basta:
  - Tools > Global Options > General
  - em R version, selecione a pasta da nova versão e pronto
  - Apply > OK

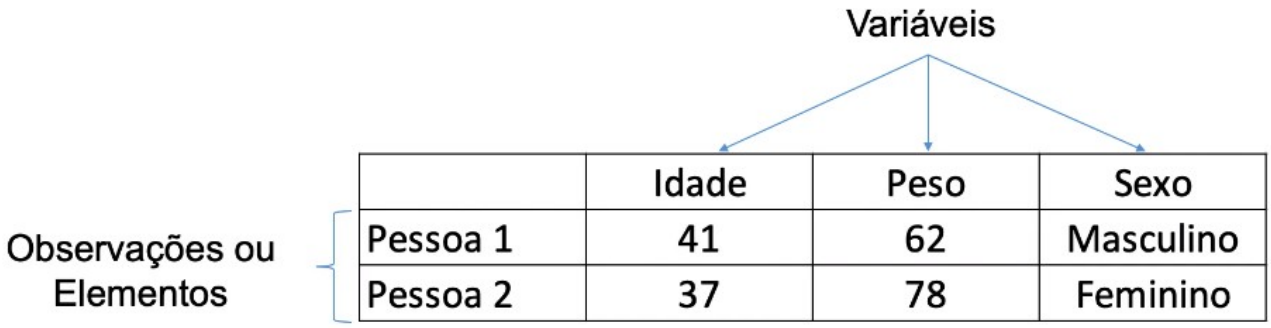
Definindo Diretório de Trabalho

- Antes de começar, defina um diretório para seus projetos em R. Lembre-se de criar as pastas sem espaços em branco ou acentos.
- No RStudio, aponte para a nova pasta criada (sempre que for realizar um novo projeto):
  - Aba Session > Set Working Directory > Choose Directory
  - Selecione a pasta e clique em Open
  - No console, aparecerá o comando setwd (“caminho da pasta do projeto”)
- Abra um novo script R em New File e salve-o (já no novo diretório)

3.3. Linguagem R – Fatores, Estruturas de Controle e Funções

Variáveis Qualitativas e Quantitativas

Variável é a característica de interesse que é medida em cada elemento da amostra ou população. Como o nome diz, seus valores variam de elemento para elemento. As variáveis podem ter valores numéricos ou não numéricos.



	Idade	Peso	Sexo
Pessoa 1	41	62	Masculino
Pessoa 2	37	78	Feminino

- Variáveis Qualitativas (Categóricas): representam uma classificação dos indivíduos
  - Nominais: profissão, sexo, religião
  - Ordinais: escolaridade, classe social, fila
- Variáveis Quantitativas (Numéricas):
  - Discretas (limite finito): número de filhos, número de carros, número de acessos
  - Contínuas (qualquer tipo de valor): altura, peso, salário

Uma das principais diferenças entre uma variável categórica e uma variável quantitativa é que uma variável categórica pode pertencer a um número limitado de categorias.

Entender o tipo de variável é importante porque determina o tipo de técnica de análise a ser utilizada na resolução de um problema.

As distinções são menos rígidas do que essas descrições.

Uma variável originalmente quantitativa pode ser coletada de forma qualitativa. Por exemplo, a variável idade, medida em anos completos, é quantitativa (contínua); mas, se for informada apenas a faixa etária (0 a 5 anos, 6 a 10 anos, etc.), é qualitativa (ordinal).

Outro ponto importante é que nem sempre uma variável representada por números é quantitativa. Por exemplo, o número do telefone de uma pessoa, da casa ou de sua identidade; e sexo, às vezes, é registrado na planilha de dados como 1 se masculino e 2 se feminino.

Lembre-se:

Você precisa conhecer os dados que tem em mãos, para poder trabalhar sua análise e selecionar as técnicas adequadas.

Fatores e Fatores Ordenados

Fatores

O termo fator se refere a tipos de dados estatísticos usados para armazenar variáveis categóricas. Fatores são variáveis categóricas que são muito úteis em sumarização de estatísticas, plots e regressões.

Fatores representam uma maneira muito eficiente para armazenar valores de caracteres, porque cada caractere único é armazenado apenas uma vez e os dados são armazenados como um vetor de inteiros.

É como se a linguagem R transformasse uma palavra em um vetor de números inteiros, para que se ganhe em performance na hora de fazer a execução de um processo de análise. É uma maneira de otimizar o armazenamento e o processamento de variáveis categóricas.

Para criar fatores usamos a função factor().

Fatores Ordenados

São fatores pelos quais preserva-se a ordenação natural dos níveis das variáveis. A ordenação segue a ordem alfabética, ao menos que outra ordenação diferente seja definida pelo usuário.

Para criar fatores ordenados usamos a função factor(…, ord=T) ou ordered().

Estruturas de Controle

Estruturas de controle permitem que se façam validações e se repitam um bloco de código, um número n de vezes, em que se realize mudanças no comportamento do script, de acordo com determinadas regras.

Condicionais If-Else

- if(condição){conjunto de tarefas}
  - else if(condição){conjunto de tarefas}
  - else {outro conjunto de tarefas}
- ifelse(condição, tarefa1, tarefa2)
- ifelse(condição, tarefa1,
  - ifelse(condição, tarefa1, tarefa2))

Loop For

- for(i in 1:N){conjunto de tarefas}

Loop While

- while(condição satisfeita){conjunto de tarefas}

Repetições

- rep(x, y) - “rep(repita x, y vezes)”
- repeat {}

Funções

Funções são objetos, dentro da linguagem R, que evitam repetição de código. Deixam o código mais legível, menos repetitivo e mais elegante.

Tudo o que você atribui com "<-" vira um objeto no R.

Sintaxe Padrão:

nome_da_função(parâmetros){código a ser executado}

Sintaxe Sem Número Fixo de Parâmetros:

nome_da_função(...){código a ser executado}

Sintaxe Sem Parâmetros:

nome_da_função(){código a ser executado}

Funções Anônimas (bom para quando o código for usado uma única vez):

teste_func <- sapply(c(1:10), *function(x) {x %% 2 == 0}*)

Precisamos ficar atentos ao escopo de uma Função!

Ao criar uma função, tudo que está dentro da função tem um escopo local, ou seja, só existe dentro da função.

Criando Funções:

function(argumentos){corpo da função}

nome_da_funcao <- function(argumentos){corpo da função}

Funções Built-in:

São funções internas da linguagem R.

- c() - vetor
- matrix() - matrizes
- seq() - sequencia de números
- help() - ajuda da linguagem
- contributors() - lista os contribuidores da linguagem R

Família Apply – Uma forma elegante de se fazer loops em R

apply() - arrays e matrizes

tapply() - os vetores podem ser divididos em diferentes subsets

lapply() - vetores e listas

sapply() - versão amigável da lapply

vapply() - similar a sapply, com valor de retorno modificado

rapply() - similar a lapply()

eapply() - gera uma lista

mapply() - similar a sapply, multivariada

by

Se você estiver trabalhando com os objetos:

list, numeric, character (list/vecor) => sapply ou lapply

matrix, data.frame (agregação por coluna) => by / tapply

Operações por linha ou operações específicas => apply

Expressões Regulares em R

São um conjunto de caracteres, que formam um padrão que permite fazer buscas em strings.

Tabela de Expressões Regulares:

- \\d - Dígitos, 0,1,2 ... 9
- \\D - Não dígito
- \\s - Espaço
- \\S - Não Espaço
- \\w - Palavra
- \\W - Não Palavra
- \\t - Tab
- \\n - Nova Linha
- ^ - Começo da String
- $ - Fim da String
- \ - Caracteres especiais, e.g. \\ is "\", \+ is "+"
- | - Alternation match. e.g. /(e|d)n/ matches "en" and "dn"
- • - Any character, except \n or line terminator
- [ab] - a or b
- [^ab] - Any character except a and b
- [0-9] - All Digit
- [A-Z] - All uppercase A to Z letters
- [a-z] - All lowercase a to z letters

- [A-z] - All Uppercase and lowercase a to z letters
- i+ - i at least one time
- i* - i zero or more times
- i? - i zero or 1 time
- i{n} - i occurs n times in sequence
- i{n1,n2} - i occurs n1 - n2 times in sequence
- i{n1,n2}? - non greedy match, see above example
- i{n,} - i occures >= n times
- [:alnum:] - Alphanumeric characters:
- [:alpha:] - and [:digit:] [:alpha:] Alphabetic characters: [:lower:] and [:upper:]
- [:blank:] - Espaços em branco: e.g. space, tab
- [:cntrl:] - Control characters
- [:digit:] - Digitos: 0 1 2 3 4 5 6 7 8 9
- [:graph:] - Graphical characters: [:alnum:] and [:punct:]
- [:lower:] - Lower-case letters in the current locale
- [:print:] - Printable characters: [:alnum:], [:punct:] and space
- [:punct:] - Punctuation character: ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ` { | } ~
- [:space:] - Space characters: tab, newline, vertical tab, form feed, carriage return, space
- [:upper:] - Upper-case letters in the current locale
- [:xdigit:] - Hexadecimal digits: 0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f

3.4. Linguagem R – Gráficos

Introdução

Os gráficos fazem parte de todo o processo de Data Science, desde a análise exploratória até a apresentação do resultado final. É um equívoco achar que os gráficos servem apenas como produto do resultado final. Utilizá-los durante todo o processo, pode permitir insights, visualizar padrões, etc.

O que é visualização de Dados?

Visualização de dados (DataViz) é a representação de dados em formato gráfico.

Gráficos, Tabelas e Estatísticas tornam a compreensão dos dados muito mais fácil.

O que são gráficos?

O gráfico é uma representação com forma geométrica, construída de maneira exata e precisa, a partir de informações numéricas, obtidas através de pesquisas e organizadas em uma tabela.

Como a Linguagem R Trata as Visualizações de Dados?

Existe um consenso na comunidade de Data Science que a linguagem R fornece algumas das melhores ferramentas para geração de gráficos.

Os gráficos podem ser replicados, modificados e publicados com apenas algumas poucas linhas de código.

Pacote Básico de Plotagem (Base Plotting System)

Ele é formado por 2 pacotes:

- graphics – contém as funções gráficas básicas, incluindo plot, hist e boxplot.
- GrDevices – contém as implementações de dispositivos gráficos como X11, pdf, PostScript, png, etc.

Os plots (gráficos bem simples) são objetos construídos através de funções e com atributos.

Podemos criar gráficos bem mais completos e profissionais usando pacotes como ggplot2, lattice e outros pacotes R.

Usando o Base Plotting System é possível criar gráficos do tipo:

- Colunas
- Barra
- Linha
- Dispersão
- Área
- Bolhas
- Superfície

- Cone
- Pizza

É importante atentar para os tipos de pontos, o peso e tamanho desses pontos, tipos de linhas e cores utilizados nos gráficos, pois impacta em como as pessoas vêm e interpretam as informações dos gráficos.

Gramática dos Gráficos

Os gráficos são construídos em camadas e cada uma delas adiciona um elemento visual.

Essas camadas são:

- os dados – o conjunto de dados a ser analisado
- a estética – a escala em que mapeamos os dados
- a geometria – os elementos visuais usados para representar os dados
- *facets* – visualização do gráfico em porções menores (é quando se utiliza uma única área de visualização para apresentar vários gráficos de maneira simultânea)
- a estatística – representação e análise dos dados
- as coordenadas – a área na qual o gráfico será construído
- os temas – visual geral do gráfico

Quando os elementos se juntam, acontece o que chamamos visualização dos dados.

A gramática dos gráficos descreve como os elementos devem ser combinados para formar uma visualização.

Scatterplot – Gráfico de Dispersão (base plotting system)

Mostra a relação entre duas variáveis, uma independente e outra dependente. Ex.: preço imóvel/nº quartos.

Boxplot – Box-and-whiskey (base plotting system)

Esse gráfico fornece as seguintes informações (por padrão):

- Mediana (linha dentro do retângulo)
- Valores mínimos e máximos da variável (linhas foras do retângulo)
- Quartis (retângulo)

Histogram – Histograma (base plotting system)

Usado para visualizar a distribuição de frequência de uma variável.

O histograma NÃO é um gráfico de colunas; e suas caixinhas são chamadas de bins.

Barplot – Gráfico de Barras (base plotting system)

Um dos gráficos mais utilizados, porque é simples e objetivo, transmitindo a informação de maneira direta.

Pie chart – Gráfico de Pizza (base plotting system)

Amado por usuários comuns, odiado por cientistas de dados. Isso porque existem gráficos melhores para se utilizar, como o gráfico de barras.

Quanto menos fatias, mais simples de se interpretar. Quanto mais fatias, o gráfico acaba se tornando muito poluído e complexo de se interpretar.

Explorando ggplot2

É um sistema completo, alternativo ao sistema básico de gráficos do R. Oferece mais opções de modificação, legendas prontas e formatação mais sólida.

Acesse o endereço abaixo para fazer o download da folha de referência do ggplot2 em português:

https://rstudio.com/wp-content/uploads/2016/03/ggplot2-cheatsheet-portuguese.pdf

No link abaixo, encontra-se o guia completo:

https://cran.r-project.org/web/packages/ggplot2/ggplot2.pdf

Lattice

O pacote Lattice é um sistema de visualização de dados de alto nível, poderoso e elegante, com ênfase em dados multivalorados (por isso, é uma excelente ferramenta para pesquisadores que necessitam realizar análise multivalorada).

Não é tão elegante como o ggplot2 e nem tão simples como o base plotting system, mas é um intermediário entre os dois, sendo muito útil durante o processo de análise de dados.

Na criação de gráficos, condições e agrupamentos são 2 conceitos importantes, que permitem compreender mais facilmente os dados que se tem em mãos. O conceito por trás do Lattice é agrupar os dados e criar visualizações de forma que fique mais fácil a busca por padrões.

Mapas

Gráficos de mapas. Mais detalhes, no script.

3.5. Manipulação de Arquivos TXT, CSV e Planilhas Excel em R

Introdução

Qual o objetivo em usar a Linguagem R?

Analisar dados.

De onde importamos os dados para o R?

De diversas fontes, suportadas pela Linguagem R.

- Arquivos Texto – flat files (txt, csv)
- arquivos Excel (xls, xlsx)
- Bancos de Dados (Oracle, SQL Server, MySQL, PostgreSQL, SQLite)
- Softwares Estatísticos (SAS, SPSS, Stata)
- Dados da Internet (Web Crawling, Web Scraping)
- Data Lakes (Apache Hadoop)

Do mesmo jeito que é possível importar todos esses tipos de dados, o R pode gravá-los como destino (exportá-los).

Importação e Manipulação de Arquivos em R

A maior parte do seu tempo em tarefas de Ciência de Dados será usada na preparação dos dados.

A linguagem R apresenta 3 pacotes principais para carga de dados a partir de arquivos texto:

- utils
  - read.table()
  - read.csv()
  - read.delim()
- readr
  - read_delim()
  - read_csv()
  - read_tsv()
- data.table
  - fread()

Pacote utils	Pacote readr
read.table()	read_delim()
read.csv()	read_csv()
read.delim()	read_tsv()

Pacote utils

É automaticamente carregado na sua sessão R, pode importar arquivos simples em diferentes formas, através das funções:

read.csv	Para valores separados por vírgula e ponto como separador decimal
read.csv2	Para valores separados por ponto e vírgula e vírgula como separador decimal
read.delim	Para valores separados por tab e ponto como separador decimal
read.delim2	Para valores separados por tab e vírgula e vírgula como separador decimal
read.fwf	Para valores com número exato de bytes por coluna
read.table	Faz a leitura de um arquivo em formato de tabela e carrega como dataframe

read.table()

Muito útil quando se está fazendo a leitura de arquivos ASCII*, que contém dados em formato retangular.

**ASCII: American Standard for Computer Information Interchange – padrão internacional que define o formato de um arquivo, que contenha apenas caracteres que representam letras, números e alguns outros símbolos. É um padrão mundialmente aceito.*

read.table("arquivo.txt", header = TRUE, sep = ",", stringsAsFactors = FALSE)

read.table("arquivo.txt", header = TRUE, sep = "\t", stringsAsFactors = FALSE)

read.csv()

Muito útil para importar arquivos com separadores de coluna.

read.csv("arquivo.csv", stringsAsFactors = FALSE)

read.csv2("arquivo.csv", sep = ”;”, dec = ”,”, stringsAsFactors = FALSE)

read.delim()

Muito útil para importar arquivos com qualquer tipo de separador de colunas.

read.delim()

read.delim("arquivo.txt") read.delim2("arquivo.txt")

Lembre-se dos Parâmetros:

- header: vai ou não importar o cabeçalho do arquivo
- col.names: definir o título das colunas
- na.string: preenche valores sem dado com NA

- colClasses: mudar o tipo de dado de uma coluna
- sep: separador
- stringsAsFactors: modifica o comportamento padrão do R

Pacote readr

Lançado em Abril/2015 pelos desenvolvedores do RStudio.

Possui as mesmas funcionalidades do pacote utils, só que com uma performance muito superior.

É necessário instalar o pacote – install.packages(“readr”).

Pacote data.table

Usado para carga e manipulação de arquivos na ordem de 100 GB em memória RAM. Ou seja, é o pacote ideal para grandes conjuntos de dados.

Importação e Manipulação de Planilhas Excel em R

Atenção: É mais seguro e vantajoso converter as planilhas excel em arquivos csv.

Pacote XLConnect:

- loadWorkbook() - carrega o Workbook, que é o conjunto de planilhas do arquivo excel
- getSheets()
- readWorksheet()
- createsheet()
- writeWorksheet()

Obs.: Esse pacote requer a instalação do JDK.

Instalando e configurando o JDK:

1. Faça o download do JDK gratuitamente no site da Oracle:
   https://www.oracle.com/technetwork/java/javase/downloads/index.html
2. Instale no seu computador
   Instale o diretório java direto no drive C, retirando eventuais espaços
3. Configure a variável de ambiente JAVA_HOME
   Aponte para o diretório de instalação do JDK
4. Inclua o diretório JAVA_HOME/bin na variável de ambiente PATH
   Mova-o para cima (primeira posição da lista)

Pacote xlsx *(permite trabalhar com o formato mais atual das planilhas do excel)*:

- read.xlsx(file, sheetIndex, header=TRUE, colClasses=NA)
- read.xlsx2(file, sheetIndex, header=TRUE, colClasses="character")

Pacote readxl:

- read_excel()
- Excel_sheets()

Pacote gdata:

- read.xls()

Obs.: requer a instalação do Perl.

Dicas para Importação e Manipulação de Arquivos em Linguagem R:

- Em seus arquivos, evite espaços em branco e números como título para as colunas (use nomes representativos).
- Normalmente, a primeira linha de cada arquivo é o cabeçalho, a lista de nomes para cada coluna.
- Para concatenação de palavras, use "." ou "_".
- Use nomes curtos como título de coluna.
- Evite o uso de caracteres especiais.
- Dados NA (Not Available) podem existir no seu conjunto de dados e isso será tratado no processo de limpeza.

Atenção aos Detalhes:

- Encoding
- Linha de cabeçalho
- Separador de colunas
- Quoting (aspas)
- Missing values
- Linhas em branco
- Espaços em branco em campos do tipo caracter
- Comentários

Outros Pacotes para Importação de Arquivos:

- rjson – leitura de arquivos JSON para R
- XML – leitura de arquivos xml
- httr – leitura de páginas html para o R
- Rcurl – Web Crawling (percorrer site em busca de dados)
- foreign – leitura de arquivos do SPSS, SAS
- sas7bdat – leitura de arquivos SAS

3.6. Trabalhando com Bancos de Dados Relacionais e NoSQL em R

Introdução

Bancos de Dados como Fontes de Dados

Bancos de Dados Relacionais – Dados Estruturados (RDBMS – Relational Database Management System)

Bancos de Dados Não Relacionais – Dados Semi ou Não Estruturados (NoSQL)

Introdução aos Bancos de Dados Relacionais

Um banco de dados relacional é um conjunto de arquivos físicos que armazenam os dados em formato estruturado, organizados em tabelas que, por sua vez, são divididas em linhas e colunas.

Usamos um SGBD (Sistema Gerenciador de Bancos de Dados ou em inglês RDBMS – Relational Database Management System) para gerenciar os arquivos do banco de dados.

A criação de um banco de dados relacional é baseada na modelagem relacional, trabalho normalmente feito por analistas de sistemas ou administradores de bancos de dados.

Usamos linguagem SQL (Structured Query Language) para manipular os dados em um banco de dados relacional.

A linguagem R suporta diversos bancos de dados relacionais através de pacotes específicos para cada banco de dados e podemos ainda fazer conexão aos bancos de dados usando um driver padrão.

Principais Bancos de Dados Relacionais

www.db-engines.com/en/ranking

- Oracle
- MySQL (www.mysql.com – usado no curso o mysql community)
  - Componentes utilizados:
    - MySQL Server
    - MySQL Workbench
    - MySQL Shell
    - MySQL Documentation
    - Samples and Examples
- Microsoft SQL Server
- PostgreSQL
- MongoDB
- IBM Db2
- SQLite
- etc.

Introdução a Linguagem SQL

Trata-se de uma linguagem de consulta e não de programação.

Sistemas Gerenciadores de Bancos de Dados Relacionais

Tudo o que fazemos no banco de dados, passa pelo SGBD.

A linguagem SQL é implementada de forma diferente em diferentes SGBDs, mas a base da linguagem é a mesma.

- Oracle utiliza PL/SQL
- MS Access utiliza o JET SQL
- MS SQL Server utiliza a T-SQL

Quais os Benefícios da Linguagem SQL?

- Permite que os usuários acessem dados em sistemas de gerenciamento de bancos de dados relacionais.
- Permite a manipulação de dados armazenados em bancos de dados.
- Permite a criação e remoção de objetos no banco de dados (tabelas, índices, visões, procedimentos armazenados).
- Permite que os usuários possam definir restrições de acesso.

Como a linguagem SQL é processada pelo SGBD?

Na seguinte ordem:

1. SQL Query
2. Processador da Linguagem SQL
    - Parser/Optimizer
3. Motor RDBMS (ou SGBD)
    - Gestor de Transação
4. Banco de Dados

Tipos de Instrução SQL

- DDL – Data Definition Language (responsável pela definição dos dados – criação de objetos)
    - Create
    - Alter
    - Drop
- DML – Data Manipulation Language (responsável pela manipulação dos dados)

    - Select
    - Insert
    - Delete
    - Update
- DCL – Data Control Language (responsável pela definição de privilégios de acesso)
    - Revoke
    - Alter

Introdução aos Bancos de Dados NoSQL (Not Only SQL)

NoSQL é uma tecnologia de banco de dados projetada para suportar os requisitos de aplicações em nuvem e arquitetado para superar em escala e desempenho as limitações de bancos de dados relacionais (RDBMS).

Bancos de dados NoSQL são usados para armazenar dados semi ou não estruturados e ganharam notoriedade por conta do crescimento do Big Data e das aplicações modernas.

Principais bancos NoSQL:

- Graph:
    - Neo4j
    - FlockDB
    - ArangoDB
    - GraphDB
- Document:
    - mongoDB
    - CouchDB
    - RavenDB
    - Terrastore
- Key-Value:
    - Oracle NoSQL DB
    - Memcache
    - Redis
    - Voldemort
- Column:
    - Cassandra
    - Apache Hbase

- Hypertable
- Accumulo

Uma lista completa de bancos de dados NoSQL pode ser encontrada em:

www.nosql-database.org

E por que devo aprender a usar um banco de dados NoSQL?

Porque estamos na era do Big Data.

MongoDB	**RDBMS**
Database	Database
Collection	Tabela
Document	Linha/Tupla
Field	Coluna
Embedded Documents	Join de Tabelas
Primary Key	Primary Key

3.7. Manipulação de Dados com R

Introdução

Data Wrangling – também conhecido como (aka):

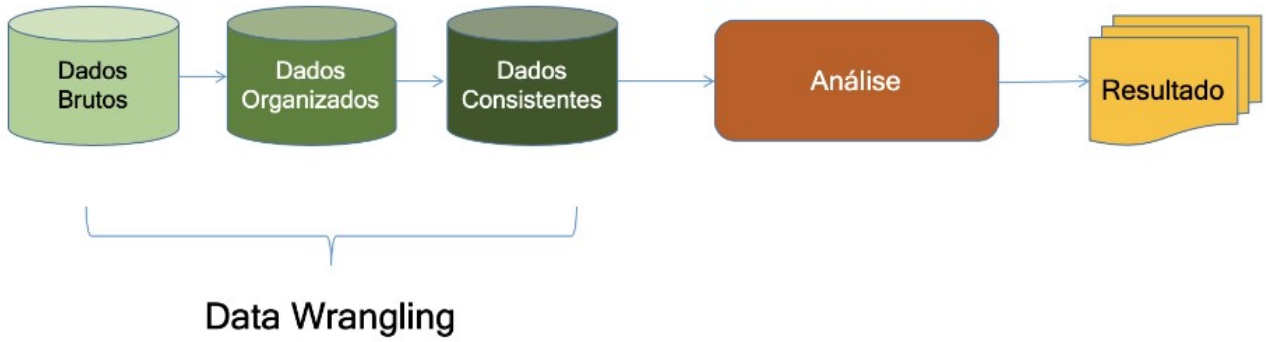
- Data Preprocessing
- Data Preparation
- Data Cleansing
- Data Scrubbing
- Data Munging
- Data Transformation
- Data Fold, Spindle, Mutilate…

A manipulação de dados pode ter muitas definições diferentes.

Mas o objetivo é um só: preparar os dados para o processo de análise, pois dificilmente os dados virão no formato ideal.

O que é Data Wrangling?

Preparação dos dados.



De maneira simples: Limpeza, Processamento, Organização e Manipulação.

Pacotes Para Manipulação de Dados em R

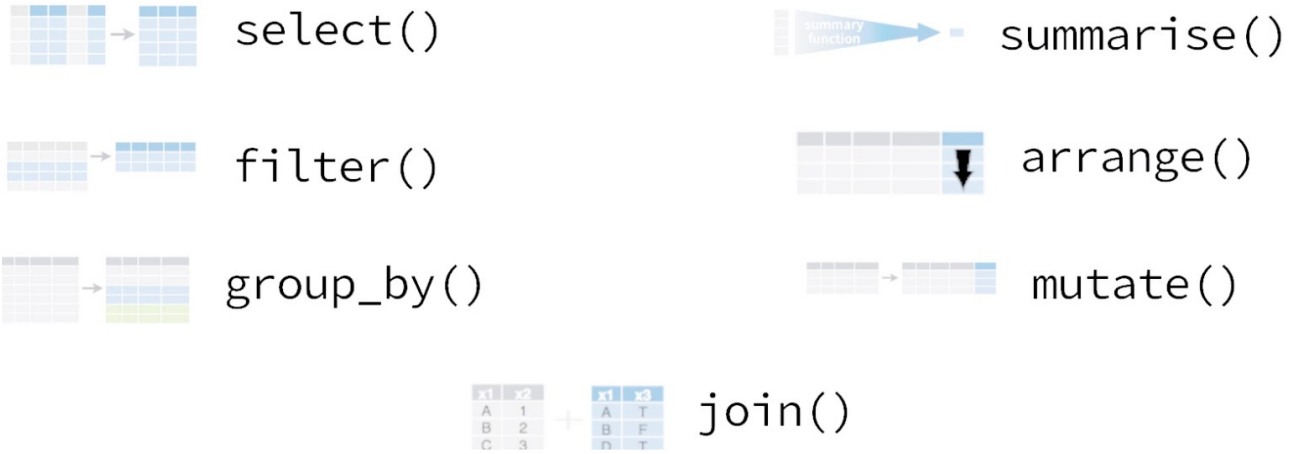
O que fazemos essencialmente durante a etapa de manipulação dos dados?

- Cada Variável em uma Coluna
- Cada Observação em uma Linha

A linguagem R pode ajudar o Cientista de Dados oferecendo alguns pacotes:

- dplyr – usado para Transformação de Dados:
  - select()
  - filter()

- group_by()
- summarise()
- arrange()
- join()
- mutate()



- tidyr – usado para Remodelagem de Dados:
  - gather()
  - spread()
  - separate()
  - unite()

Talvez você ainda não tenha percebido, mas com apenas uma função, somos capazes de mudar completamente o formato (shape) dos nossos dados e isso pode fazer muita diferença no processo de análise.

Operador de Concatenação

Operador %>%

filter(data, variable == numeric_value)

ou

data %>% filter(variable == numeric_value)

Utilizando o concatenador, é possível selecionar o conjunto de dados e aplicar várias funções diferentes.

3.8. Introdução à Análise Estatística de Dados – Parte 1

O Papel da Estatística em Ciência de Dados:

Sozinhos, dados são apenas um apanhado de ruído e confusão.

Para dar-lhes sentido, interpretações e significados, necessita-se de um ramo poderoso da ciência: Estatística, que é fundamental no processo de descoberta científica ao fornecer modelos capazes de aprimorar pesquisas e nortear tomadas de decisão.

Ao observar um fenômeno sucessivamente é possível notar que, muito raramente, os resultados encontrados serão iguais. Isso porque, praticamente tudo está sujeito a variação.

No entanto, o uso de métodos estatísticos permite que se facilite a compreensão e descrição dessa “inconstância” e que ela seja usada de forma a ajudar no processo de tomada de decisão.

A Estatística possui diversas aplicações e cumpre os mais variados objetivos, sendo especialmente útil e, por vezes, indispensável quando se trata de Ciência de Dados.

Usamos Estatística para descrever, resumir e explorar os dados. Ao trabalhar com Machine Learning, usamos Estatística para interpretar e avaliar os resultados do modelo.

Em Ciência de Dados, a Estatística cumpre papel importante. Mas usamos ainda Matemática para criar um modelo de Machine Learning e Ciência da Computação quando precisamos criar programas de software para análise ou armazenar e processar os dados de forma distribuída, o que é necessário quando o volume de dados é muito grande. Isso sem falar no conhecimento das áreas de negócio.

Data Science envolve muitas áreas e a Estatística é uma delas.

As 3 Grandes Áreas da Estatística:

- Probabilidade:
  - Estudo da aleatoriedade e da incerteza.
  - Utiliza métodos de quantificação das chances associadas aos diversos resultados.
- Estatística Descritiva:

    - Utiliza métodos para coleta, organização, apresentação, análise e síntese de dados obtidos em uma população ou amostra.
- Estatística Inferencial:
    - É o processo de estimar informações sobre uma população a partir dos resultados observados em uma amostra.

Estatística é a ciência, parte da Matemática Aplicada, que fornece métodos para coletar, descrever, analisar, apresentar e interpretar dados, para utilização dos mesmos na tomada de decisões.

População e Amostra:

Sempre que você se deparar com um novo conjunto de dados, a primeira pergunta que deve ser feita é: qual é a minha população e qual é a minha amostra?

População	Amostra
São todos os elementos distintos – indivíduos, itens ou objetos – cujas características estejam sendo estudadas.	É uma parte da população, sendo coletada a partir da população que está sendo estudada.

Como Garantir que a Amostra Representa Fielmente a População?

A melhor estratégia para criar uma amostra fiel é através de randomização ou aleatorização.

Simplesmente, você coleta sua amostra de forma randomizada, sem escolher exatamente quem fará parte da amostra.

Não pense que é fácil como parece. Tão importante quanto a randomização é o tamanho da amostra e existem diversas técnicas para a coleta de amostra de dados.

Parâmetros x Estatísticas:

As **estatísticas** se baseiam nos dados da **amostra** e não em dados populacionais. Quando se coletam dados de toda uma população, temos o chamado censo.

Se você depois resume toda a informação do **censo** em um número, esse número é um **parâmetro**, não uma estatística.

Parâmetros – características sobre a população. Valores calculados usando dados da população são chamados de parâmetros.

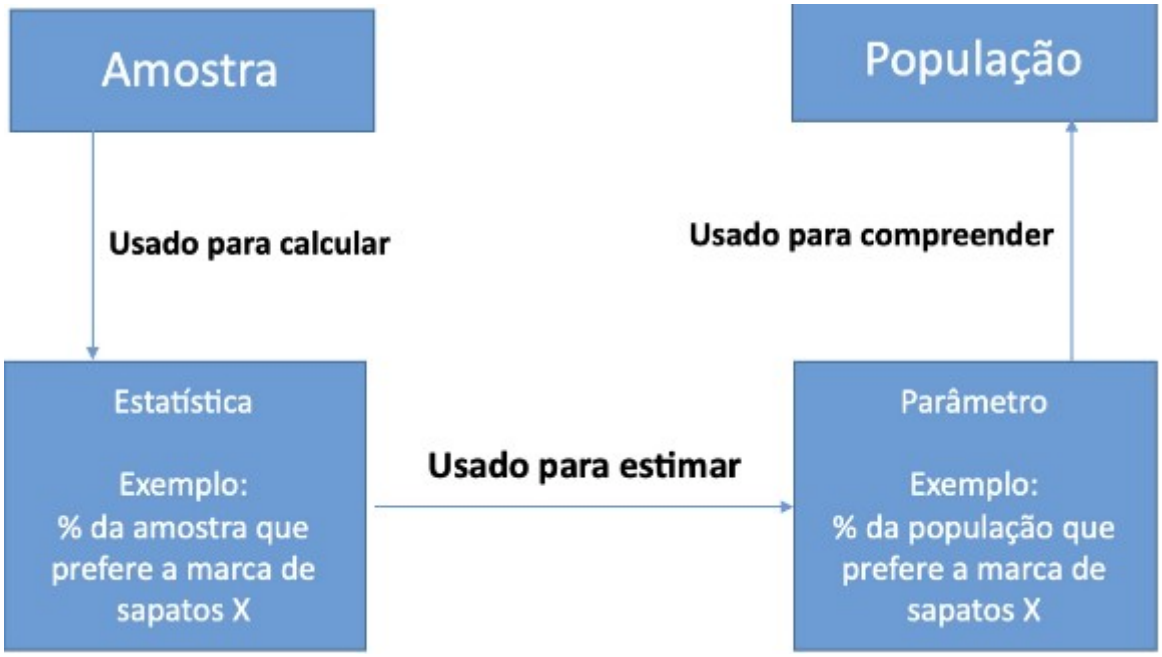
Estatísticas – características sobre a amostra. Valores calculados usando dados da amostra são chamados de estatísticas.

Estatística Inferencial realiza deduções e conclusões sobre a população, com base nos resultados obtidos da análise da amostra.

E por que não analisamos a população inteira? Por que precisamos de uma amostra?

Por diversas razões!

- Custo
- Tempo
- Necessidade



Observações x Variáveis:

Vamos relembrar um conceito fundamental: **dado é diferente de informação**

Dados – valores coletados através de observação ou medição.

Informação – dados que são transformados em fatos relevantes e usados para um propósito específico.

Dados não fazem sentido se não forem colocados em um contexto.

Os dados podem ser obtidos através de duas fontes principais:

Dados Primários	Dados Secundários
Coletados por quem faz a análise Confiáveis Possuem maior controle	Coletados por terceiros Não Confiáveis Não possuem muito controle

	Vantagens	Desvantagens
Dados Primários	Confiabilidade Qualidade Controle das informações Acertabilidade nos resultados	Alto custo Demanda tempo maior Equipe grande

	Dados atualizados	
Dados Secundários	Baixo custo Rapidez Existência de diversas fontes Diversidade de informações para quantificação de questões	Falta de controle Dados inadequados Diversidade na classificação dos dados Dados desatualizados Fontes não confiáveis Dificuldade de reproduzir um estudo obtendo os mesmos resultados

Observação – uma observação é uma ocorrência de um item de dados específico que é gravada sobre uma unidade de dados. Também chamada de registro.

Variável – uma variável é a característica de interesse que é medida em cada elemento da amostra ou população. Como o nome sugere, seus valores variam de elemento para elemento. As variáveis podem ter valores numéricos ou não numéricos.

Tipos de Variáveis:

Qualitativas – utilizam termos descritivos para descrever algo de interesse. Ex: cor dos olhos, estado civil, religião, sexo, grau de escolaridade, classe social, tipo sanguíneo, cor da pele, etc.

- Nominais – não representam nenhum tipo de hierarquia. Ex: profissão, sexo, religião.
- Ordinais – representam algum tipo de hierarquia. Ex: escolaridade, classe social, fila.

Quantitativas – representadas por valores numéricos que podem ser contados ou medidos. Ex: número de crianças em uma sala de aula, peso do corpo humano, idade, número de filhos, etc.

- Discretas – normalmente é possível contar esse tipo de variável. Ex: número de filhos, de carros ou de acessos.
- Contínuas – representam dados que podem assumir qualquer valor dentro de um range. Ex: altura, peso, salário, etc.

Atenção!

Um dado classificado como idade é quantitativo. Ex: 11, 15, 18, 25, 42 anos.

Entretanto, se esse dado for informado por faixa etária, ele é qualitativo (ordinal). Ex: 0 a 5 anos; 6 a 12 anos; 13 a 18 anos; etc.

É muito importante classificar os tipos de dados das variáveis, pois eles permitirão a você escolher o melhor teste estatístico a ser utilizado na análise dos dados.

A simples visualização dos dados, ainda que contenha toda a informação, muitas vezes não diz nada. Simplesmente, olhar para os dados não fornece um quadro claro do que pode estar acontecendo, especialmente quando a quantidade de dados for muito grande. Por isso, podemos ensinar algoritmos a fazer isso por nós. Exatamente onde começa o trabalho em Machine Learning.

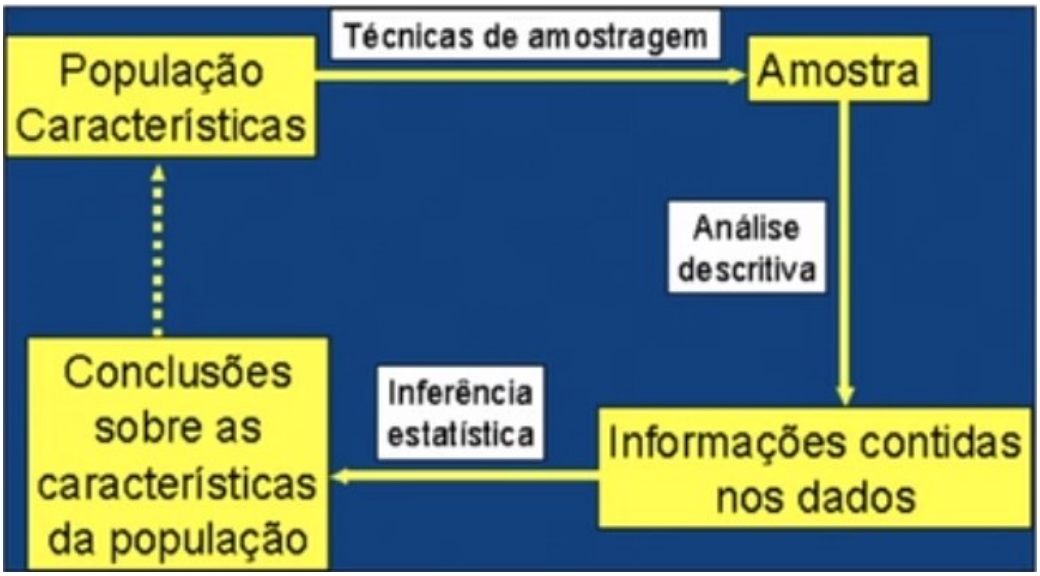
3.9. Introdução à Análise Estatística de Dados – Parte 2

Estatística Descritiva:

A **Estatística** é um conjunto de técnicas que permite, de forma sistemática, organizar, descrever, analisar e interpretar dados oriundos de estudos ou experimentos, realizados em qualquer área do conhecimento.

A **Estatística Descritiva** é a etapa inicial da análise utilizada para descrever e resumir os dados.

A disponibilidade de uma grande quantidade de dados e de métodos computacionais muito eficientes revigorou esta área de estatística.



Com a Estatística Descritiva podemos descrever os dados usando 2 tipos principais de medidas:

Medidas de Tendência Central	Medidas de Dispersão
Média Mediana Moda Valor Máximo e Valor Mínimo Amplitude	Desvio Padrão Variância Coeficiente de Variação

Representação Gráfica dos Dados:

Uma vez que as informações são coletadas dá-se início ao processo de análise dos dados. Um dos passos mais importantes dessa análise é, justamente, a construção e plotagem dos gráficos.

A principal função de uma representação gráfica é proporcionar a análise dos dados com maior clareza e fluidez, além de ajudar na solução de problemas e dificuldades à medida que estas aparecem.

Um gráfico não precisa, necessariamente, ser elegante nesta fase, o importante é que seja útil.

Introdução à Probabilidade:

Probabilidade é provavelmente um dos tópicos mais interessantes no campo da Estatística, pois permite medir a incerteza.

O mundo atual enfrenta muitos desafios sobre as incertezas, principalmente no mundo dos negócios. E a Probabilidade provê uma ferramenta valiosa para quantificar esta incerteza, de moram que os gestores possam tomar melhores decisões.

Probabilidade é um valor numérico que indica a chance, ou probabilidade, de um evento específico ocorrer. Esse valor numérico vai estar entre 0 e 1.

- Se um evento não possui chance de ocorrer, sua probabilidade é 0 (ou 0%)
- Se temos certeza sobre a ocorrência do evento, sua probabilidade é 1 (ou 100%)

Evento, Experimento e Espaço da Amostra:

Experimento – é o processo de medir ou observar uma atividade com o propósito de coletar dados. Ex: jogar um dado.

Espaço da Amostra – todos os possíveis resultados de um experimento. Ex: ao jogar um dado, os resultados possíveis são {1, 2, 3, 4, 5, 6}.

Evento – um ou mais resultados de um experimento. O resultado e/ou resultados são um subconjunto do espaço da amostra.

Evento Simples – um evento com um único resultado na sua forma mais básica, que não pode ser simplificado.

Probabilidade e Possibilidade São a Mesma Coisa?

Não!

Probabilidade é a medida da **possibilidade** de um evento ocorrer.

Em outras palavras, se a chance de chover amanhã é de 40%, há menos possibilidades que chova amanhã, do que não chova.

Probabilidade Clássica:

Probabilidade Clássica: é usada quando nós sabemos o número de possíveis resultados do evento de interesse e podemos calcular a probabilidade do evento com a seguinte fórmula:

P(A) = Número de possíveis resultados do Evento A (s) /

Número total de possíveis resultados dentro do espaço a amostra (n)

onde: P(A) é a probabilidade de um evento ocorrer.

Probabilidade Empírica:

Probabilidade Empírica: é usada quando nós NÃO sabemos o número de possíveis resultados do evento de interesse e podemos calcular a probabilidade do evento com a seguinte fórmula:

P(A) = Frequência em que o evento A ocorre / Número total de observações

onde: P(A) é a probabilidade de um evento ocorrer.

Exemplo:

Qual a probabilidade de que uma pessoa que entre na loja faça uma compra?

A probabilidade clássica não poderia nos ajudar aqui, pois não temos informação sobre porque as pessoas fazem uma compra e nem quando elas fazem uma compra. Ou seja, nesse caso, é necessário uma observação prévia, então usamos a probabilidade empírica, para contar quantas pessoas que entram na loja finalizam uma compra.

Probabilidade Subjetiva:

Usamos probabilidade subjetiva quando:

- Probabilidades clássicas ou empíricas não podem ser usadas
- Dados ou experimentos não estão disponíveis para calcular a probabilidade
- Nesses casos, confiamos na experiência ou julgamento para estimar as probabilidades

Exemplo:

Um experiente Diretor de Marketing estima que há 50% de probabilidade de que o maior concorrente da empresa reduza seus preços no mês seguinte.

Regras Básicas da Probabilidade:

São 5 regras básicas que regem a teoria da probabilidade:

1. Se P(A) = 1, então podemos garantir que o evento A **ocorrerá**.
2. Se P(A) = 0, então podemos garantir que o evento A **NÃO ocorrerá**.
3. A probabilidade de qualquer evento sempre será entre 0 e 1. Probabilidades nunca podem ser negativas ou maior que 1.
4. A soma de todas as probabilidades para um evento simples, em um espaço de amostra, será igual a 1.
5. O complemento do evento A é definido como todos os resultados em um espaço de amostra que NÃO fazem parte do evento A. Ou seja:

   P(A) = 1 – P(A’), onde P(A’) é o complemento do evento A.

Regras Básicas da Probabilidade Para Mais de Um Evento:

No mundo dos negócios os eventos raramente são simples e frequentemente envolvem dois ou mais eventos.

Exemplo:

O gerente de um banco pode estar interessado em saber a probabilidade de um cliente com histórico de crédito ruim não pagar um empréstimo de cheque especial. Neste caso, temos 2 eventos:

- Evento A: cliente não paga o cheque especial.
- Evento B: cliente tem um histórico de crédito ruim.

São utilizadas 3 regras principais para cálculo de probabilidade para mais de um evento:

1. **Intersecção de Eventos** – representa o número de vezes em que os eventos A e B ocorrem **ao mesmo tempo**.
2. **União de Eventos** – representa o número de vezes em que o evento A ou o evento B ocorrem **juntos**.
3. **Adição de Eventos** – usada para calcular a probabilidade de **união de eventos**, ou seja, a probabilidade do evento A mais evento B ocorrerem.
    - A Regra da Adição depende se 2 eventos são ou não mutuamente exclusivos.
        - Eventos Mutuamente Exclusivos: aqueles que não podem ocorrer ao mesmo tempo durante um experimento.

          P(A ou B) = P(A) + P(B)
        - Eventos Não Mutuamente Exclusivos: aqueles que podem ocorrer ao mesmo tempo durante um experimento.

          P(A ou B) = P(A) + P(B) – P(A e B)

Usamos a **tabela de contingência** para ajudar nos cálculos de probabilidade para mais de um evento.

Variáveis Aleatórias Discretas e Contínuas:

Variáveis Discretas são números inteiros, gerados a partir de resultados de experimentos.

Mas no mundo dos negócios, normalmente, nos deparamos com problemas que requerem medição e cujos valores podem assumir qualquer número em um intervalo. Para isso, usam-se **variáveis contínuas**.

Dados **contínuos**, normalmente, são **medidos** e não **contados**, como no caso dos valores **discretos**.

Variáveis Contínuas são, portanto, qualquer valor no conjunto de números reais, ou subconjunto deles.

Tipos de Distribuição de Probabilidade:

Em estatística, uma **Distribuição de Probabilidade** descreve a **chance** que uma variável pode assumir ao longo de um espaço de valores.

A Distribuição de Probabilidade tem por objetivo associar uma probabilidade a cada resultado numérico de um experimento.

Ela é uma função cujo domínio são os valores da variável e cuja imagem são as probabilidades da variável assumir cada valor do domínio. O conjunto imagem deste tipo de função está sempre restrito ao intervalo entre 0 e 1.

A soma de todos os valores de uma Distribuição de Probabilidades deve ser igual a 1.

A **Probabilidade de Ocorrência** de um evento deve ser maior que 0 e menor que 1.

Uma Distribuição de Probabilidade pode ser:

- **Discreta** – descreve quantidades aleatórias de dados que podem assumir valores **finitos**.
  - Binomial
  - Poisson
  - Hipergeométrica
  - Bernoulli
- **Contínua** – descreve quantidades aleatórias de dados que podem assumir valores **infinitos**.
  - Uniforme
  - Exponencial
  - Gama
  - Chi-Quadrado

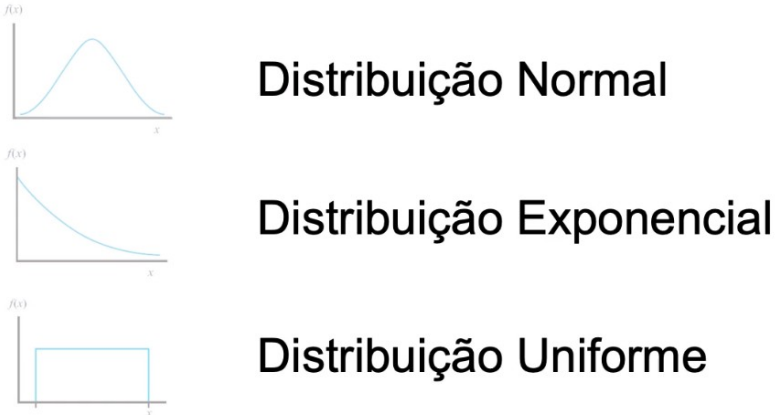
Distribuição Normal

Uma variável randômica contínua que seque uma Distribuição de Probabilidade Normal tem uma série de características distintas.

Distribuições Contínuas:

Quando transformadas em **gráficos**, as Distribuições de Probabilidade Contínua podem assumir uma variedade de formatos, dependendo dos valores dos dados. Os 3 formatos mais comuns são:

- Distribuição Normal
- Distribuição Exponencial
- Distribuição Uniforme



A **Distribuição Normal** é útil quando os dados tendem a estar próximos ao centro de distribuição (próximos da média) e quando valores extremos (outliers) são muito raros. Como é muito comum, ela é a ferramenta usada para calcular diversas estatísticas inferenciais, sendo muito importante em **Machine Learning**.

A **Distribuição Exponencial** é usada para descrever os dados quando valores mais baixos tendem a dominar a distribuição e quando valores muito altos não ocorrem com frequência.

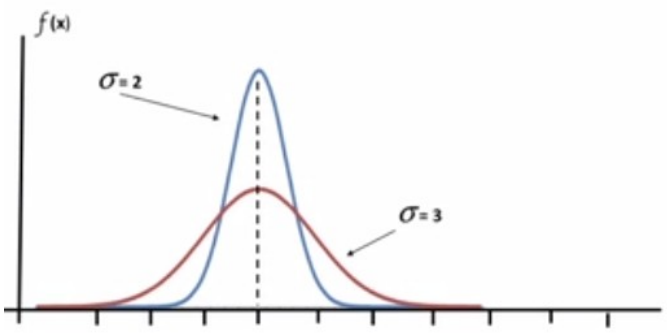
A **Distribuição Uniforme** é usada para descrever os dados quando todos os valores têm a mesma chance de ocorrer.

Distribuição Normal:

Uma variável randômica contínua que segue uma **Distribuição de Probabilidade Normal** tem uma série de características distintas. São elas:

- A distribuição tem um formato de sino e simétrico em torno da média.
- Como o formato da distribuição é simétrico, a média e a mediana possuem o mesmo valor.
- Variareis randômicas em torno da média, na parte mais alta da curva, tem maior probabilidade de ocorrer, que valores situados onde a curva é menor.
- A parte final da curva, tanto do lado direito, quanto do lado esquerdo, em uma distribuição normal, se estende indefinidamente, nunca tocando o eixo x do gráfico.

O **Desvio Padrão** tem uma função importante no formato da **curva** de uma **Distribuição Normal**.



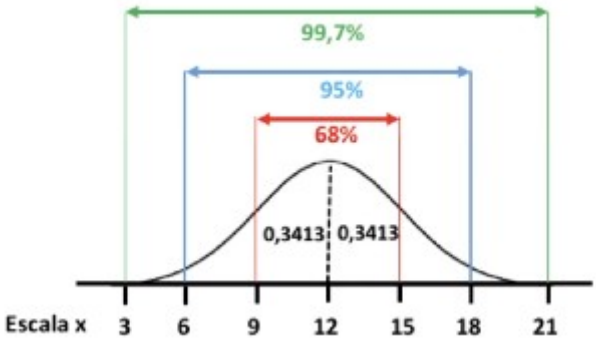
- A linha vermelha possui um desvio padrão de 3 (**?** = 3).

- A curva ficou mais aberta em relação à média.
- O tempo médio das ligações está entre 3 e 21 minutos e não mais entre 6 e 18 minutos, quando o desvio padrão é 2.
- Um desvio padrão menor resulta em uma curva mais estreita.
- Um desvio padrão maior, faz com que a curva seja mais baixa e mais aberta.

As probabilidades de distribuições normais podem ser calculadas através do uso de fórmulas, tabelas de probabilidade e softwares estatísticos como R, SAS e SPSS ou mesmo com pacotes estatísticos para a linguagem Python.

Distribuição Normal – Regra Empírica:

A **Regra Empírica** define o seguinte: se uma distribuição é simétrica e em formato de sino, aproximadamente 68%, 95% e 99% dos dados desta distribuição estarão em 1, 2 e 3 “desvios padrão” acima e abaixo da média, respectivamente:



3.10. Introdução à Análise Estatística de Dados – Parte 3

Amostragem:

Estatística Inferencial – trabalhando com dados representativos na amostra, podemos inferir o que está acontecendo na população como um todo.

Amostragem: é a técnica, processo ou pesquisa que podem ser realizadas para obter uma amostra.

Amostragem: usa a coleta, organização, apresentação e análise dos dados como meio de estudar os parâmetros de uma população.

Amostragem: é a técnica que seleciona apenas alguns elementos da população para se obter uma amostra.

Censo: é a técnica que seleciona e avalia todos os elementos da população quando se realiza uma pesquisa.

E mesmo que fosse possível medir a população inteira, seria um **desperdício** de **tempo** e **dinheiro**.

Se a **amostra** for selecionada **corretamente** e a análise sobre ela for feita seguindo as **metodologias estatísticas**, esta informação pode ser usada para fazer uma avaliação **precisa** sobre a **população** inteira.

Entretanto, existem **riscos** envolvidos em tomar decisões baseadas em **amostragem**.

A **amostragem** pode ser exposta a erros, que podem levar a decisões incorretas.

Nós podemos **quantificar a probabilidade destes erros ocorrerem**.

Tipos de Amostragem:

Existem muitas opções disponíveis para coletar uma amostra de uma população.

Amostragem Não-Probabilística

Amostragem Não-Probabilística é **subjetiva**, pois é influenciada pela pessoa que está conduzindo a pesquisa. Ela se baseia nas decisões pessoais do pesquisador.

Tipos de amostragem não-probabilística:

- Conveniência

Amostragem Probabilística

Amostragem Probabilística é **objetiva**, pois **não** é influenciada pela pessoa que está conduzindo a pesquisa.

Os elementos da amostra são selecionados aleatoriamente e todos eles possuem probabilidade conhecida de serem escolhidos.

Tal seleção ocorre através de uma forma de sorteio não viciado, como o sorteio em uma urna ou por números gerados por computador.

Tipos de amostragem probabilística:

- Aleatória Simples
- Sistemática
- Estratificada
- Conglomerados
- Reamostragem (Bootstrap)

Amostragem Probabilística (Bootstrapping):

Reamostragem é uma **técnica estatística** em que várias amostras são repetidamente extraídas da **população**.

Um tipo específico de técnica de Reamostragem é conhecido como método Bootstrap, desenvolvido por Bradley Efron, membro do Departamento de Estatística da Universidade de Stanford na Califórnia nos EUA.

O **Método Bootstrap** consiste em usar software de computador para extrair diversas amostras (com reposição), para estimar determinados parâmetros da população, tais como média e proporção.

Estatística se refere a **Amostra**.

Parâmetro se refere a **População**.

Erros de Amostragem:

Parâmetros: Valores que descrevem características da **população**, como média e mediana da população.

Estatísticas: Valores calculados a partir da **amostra**, como média e mediana da amostra.

Como as Estatísticas são calculadas a partir da amostra, que é uma parte da população, não seria razoável esperar que a média dos dados da amostra fosse igual à média dos dados da população?

A diferença entre estes 2 valores é chamada "**erro de amostragem da média da amostra**".

Fórmula do erro da amostragem:

$$\text{Erro de amostragem} = x - \mu$$

Onde:

x = média da amostra
μ = média da população

Sem dúvida amostragem é uma técnica fabulosa, que nos permite obter informações sobre uma população inteira, analisando apenas uma porção dos dados.

O **Erro Amostral** é a diferença entre um resultado amostral e o verdadeiro resultado populacional.

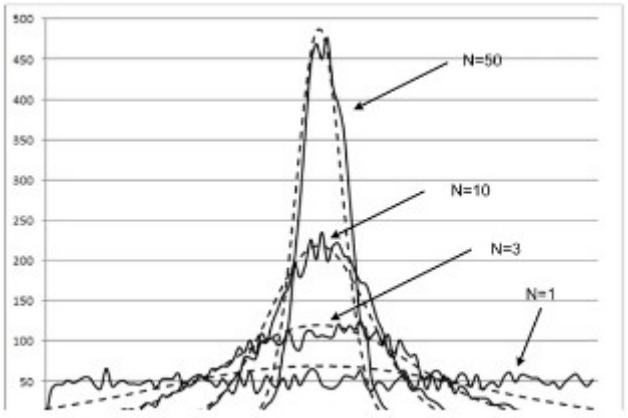
Erros de amostragem podem diferir de uma amostra para outra e podem ser **positivos** ou **negativos**.

Como regra, quanto **maior** o tamanho da **amostra**, **menor** será o **erro** de amostragem. Dessa forma, é necessário fazer um **trade-off** (uma escolha): quanto menor a amostragem, melhores serão a **eficiência**, **tempo** e **custo**, mas é maior a possibilidade de **erros**.

Teorema do Limite Central:

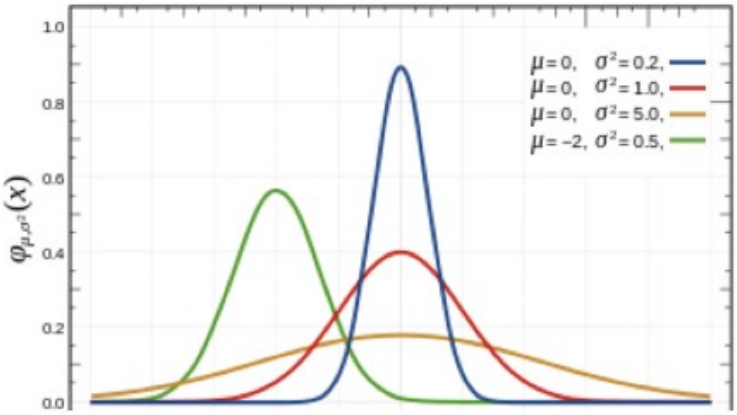
O **Teorema do Limite Central** é um importante conceito da estatística e a demonstração de muitos outros teoremas estatísticos dependem dele, tanto que é conhecido como a ”mãe de todos os teoremas” e realmente merece sua atenção.

Esse teorema afirma que quando o tamanho da amostra aumenta, a distribuição amostral da sua média aproxima-se cada vez mais de uma distribuição normal.

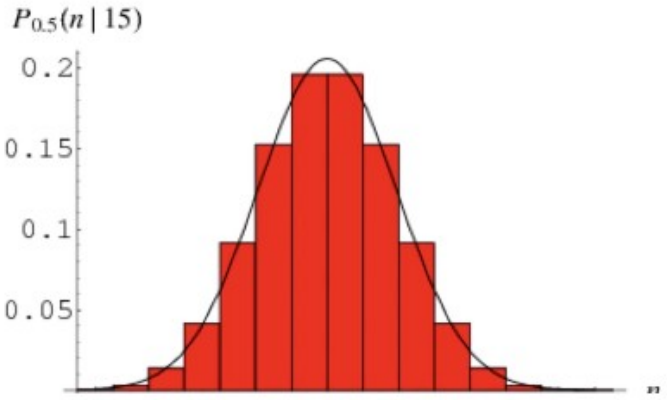


Este teorema é fundamental na teoria da inferência estatística e sem este teorema, provavelmente a estatística não teria avançado como a ciência que é hoje.

Como já vimos, uma variável aleatória pode ter uma distribuição, possuindo uma média **μ (Mu)** e um desvio padrão **σ (Sigma)**.



À medida que o tamanho ‘n’ da amostra aumenta, a distribuição das médias amostrais tende a uma distribuição normal.



Para **n >= 30**, a distribuição das médias amostrais pode ser aproximada satisfatoriamente por uma distribuição normal.

A **média das médias amostrais** será a média populacional.

Se a distribuição da variável ‘x’ for originalmente uma distribuição normal, então a distribuição das médias amostrais terá distribuição normal para qualquer tamanho amostral ‘n’.

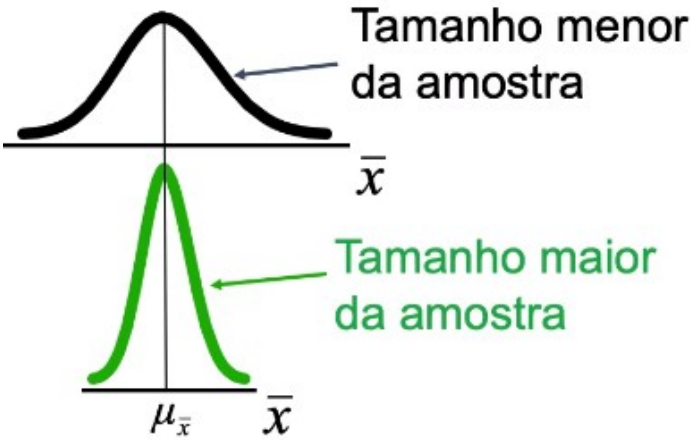
De acordo com o **Teorema do Limite Central**, médias amostrais de amostras suficientemente grandes, retiradas de qualquer população, terão uma distribuição normal.

Se a população seguir uma **distribuição normal** de probabilidade, as **médias amostrais** também terão **distribuição normal**, independente do tamanho das amostras.

A Importância do Tamanho da Amostra no Teorema do Limite Central:

O **tamanho da amostra** tem uma função importante no Teorema do Limite Central.

À medida que aumenta o tamanho das nossas amostras, o erro padrão da média se torna menor, o que reduz o erro de amostragem.



1ª Regra – Para qualquer população

O valor médio de todas as médias de amostras possíveis, a partir de um dado tamanho da população, é igual à média da população.

$$\mu_{\bar{x}} = \mu$$

2ª Regra – Para qualquer população

O desvio padrão das médias das amostras de tamanho n, é igual ao desvio padrão da população dividido pela raiz quadrada do tamanho da amostra.

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$$

Mas afinal, o que há de extraordinário no Teorema do Limite Central?

Ele nos diz que **qualquer** que seja a forma da distribuição original, suas **médias** resultam em uma **distribuição normal**.

Esse teorema possibilita medir o quanto sua média amostral irá variar, sem ter que pegar outra média amostral para fazer a comparação.

O formato da distribuição da população afetará o formato da distribuição da amostra, assim como o tamanho da amostra.

Escore z:

O **Escore z** identifica o número de desvios padrão que um determinado valor possui de distância em relação à média do conjunto de dados ao qual faz parte.

Pense no Escore z, como uma simples conversão de um valor para uma outra unidade específica, a fim de chegar a conclusões.

O Escore z em si, **não possui unidade**. É apenas um número.

Hamburguer	Restaurante	Calorias
Cheeseburguer	McDonalds's	300
Big Mac	McDonalds's	430
Whopper	Burger King	540
Double Cheeseburguer	Bob's	670
Chicken Burger	McDonalds's	780
Bacon Burger	Bob's	840
Quarteirão	McDonalds's	1.230
Mega Burger	Burger King	1.420
Média da amostra		776,30
Desvio padrão da amostra		385,10

Calculando o **Escore$_z$** para o Mega Burger: $z_{\bar{x}} = \frac{\bar{x} - \mu_{\bar{x}}}{\sigma_{\bar{x}}}$

$$\text{Escore}_z = (x - x1) / s = (1.420 - 776{,}30) / 385{,}10 = \mathbf{1.67}$$

O que 1.67 significa?

Significa que as calorias do Mega Burger, são 1.67 desvio padrão acima da média. Se o valor fosse negativo, indicaria que as calorias estariam abaixo da média.

O Escore z terá sempre um dos 3 atributos:

- **Positivo** – valores acima da média
- **Negativo** – valores abaixo da média
- **Zero** – valores iguais a média

O Escore z também possui uma característica importante: ele nos ajuda a identificar os **outliers** (valores extremos) dos nossos dados.

Valores de dados que possuem Escore z acima de **+3** ou abaixo de **-3** são classificados como **outliers**.

Intervalo de Confiança:

É o limite inferior e superior onde se busca a média da população.

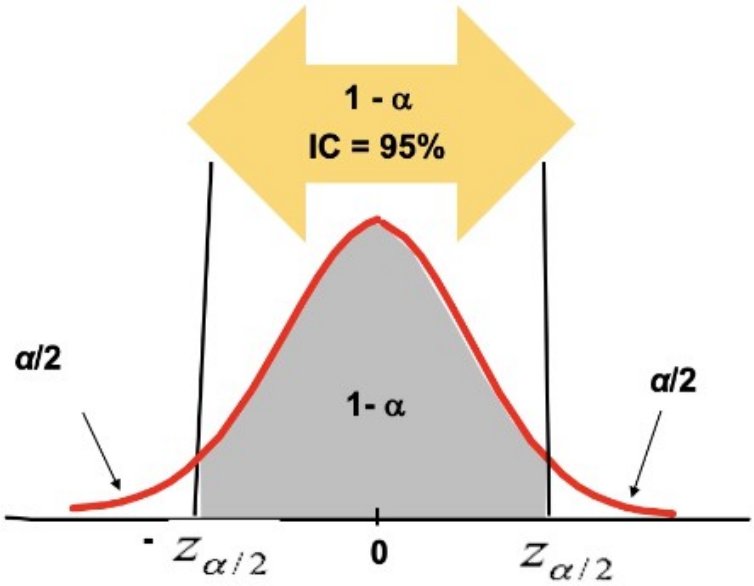
Nível de Confiança:

Nível de confiança é a probabilidade!

$$1 - \alpha$$

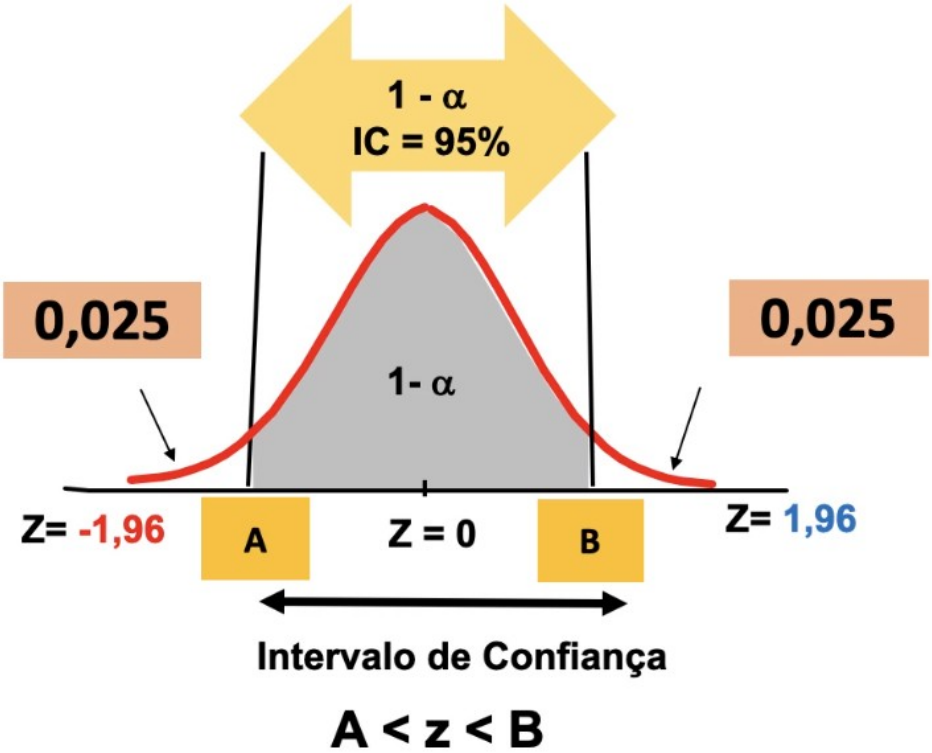
One alfa é o que chamamos de Nível de Significância.

O Nível de Confiança é expresso percentualmente e por isso usamos: **100 (1 - α)**



O **Intervalo de Confiança** consiste em um intervalo na **escala z (Escore z)** e está associado a um **NC**.

Ou seja, se coletarmos várias amostras e construirmos um intervalo de confiança para cada uma, a longo prazo, **95%** destes intervalos conteriam efetivamente a **média da população μ**.



O **valor crítico Z (Escore z)** corresponde ao valor da fronteira da área α/2 nas caudas direita ou esquerda da distribuição normal.

E por que isso é importante?

Pelo **Teorema do Limite Central**, sabemos que as médias amostrais **x** tendem a distribuir-se por uma **distribuição normal**.

Sendo **α/2** a área de cada extremo, há uma possibilidade de **α** da média amostral estar em um dos **2 extremos**.

Pela regra do complemento, há uma probabilidade da **média** estar na região **não sombreada**.

Como já sabemos, o **Intervalo de Confiança da Média** é um intervalo de **estimativa** em torno da **média da amostra**, que provê um range de valores no qual está a média da população.

De fato, a média da população raramente é conhecida, assim o **Intervalo de Confiança** é a **única evidência** que nós temos sobre a **média da população**.

Teste de Hipótese:

No mundo da **Estatística**, uma **hipótese** é uma **suposição** sobre um **parâmetro** específico de uma população, tal como média, proporção ou desvio padrão.

Podemos fazer uma suposição sobre o valor de um parâmetro da população, coletar uma amostra desta população, medir a amostra e atestar se a amostra suporta ou não a nossa suposição.

O mundo dos negócios é cheio de exemplos que lidam com testes de hipóteses!

Teste de Hipótese é um dos procedimentos estatísticos mais usados atualmente, com muitas aplicações em Data Science, como em Testes A/B aplicados por equipes de Marketing por exemplo.

O **objetivo** de um teste de hipótese é **decidir** se determinada afirmação sobre um parâmetro populacional **é ou não apoiada pela evidência** obtida de **dados amostrais**.

Cada teste de hipótese tem uma **hipótese nula** e uma **hipótese alternativa**, representados por:

- H0 – Hipótese nula
    - Representa o *status quo*, ou seja, **comprovar** uma suposição ou afirmação;
    - Valida que o parâmetro da população seja **<= ou =>** a um específico valor;
    - **É para ser verdadeira**, a menos que seja comprovada por uma evidência contrária.
- HA – Hipótese alternativa
    - Representa o oposto da hipótese nula.

Você precisa ser cuidadoso ao **definir a hipótese** nula e a hipótese alternativa. As decisões sobre as hipóteses vão depender da **natureza do teste e da pessoa** que o está conduzindo.

Uma declaração de hipótese pode somente ser usada com **parâmetro da população** (tal como μ), **não** com uma estatística de **amostra** (tal como a média da amostra).

O propósito do **teste de hipótese** é **atestar** uma **conclusão** sobre os parâmetros da população, sobre os quais **não** temos conhecimento completo.

A hipótese nula **H0 representa o status quo**, ou seja, a circunstância que está sendo testada e o objetivo dos testes de hipótese é sempre tentar rejeitar a hipótese nula.

A hipótese alternativa **HA representa o que se deseja provar ou estabelecer**, sendo formulada para contradizer a hipótese nula.

Podemos fazer apenas **2 afirmações** sobre a **hipótese nula**: **rejeitar ou não rejeitar**. A razão pela qual estamos limitados a apenas 2 conclusões possíveis é que o teste de hipótese se baseia em "**provar contradições**".

Com isso, nós podemos apenas concluir que **a hipótese pode ser verdadeira**, mas não temos evidências suficientes para afirmar que a hipótese nula é realmente verdadeira. Por conta desta limitação, **NUNCA podemos aceitar a hipótese nula**.

Podemos apenas dizer que: **Não há evidências suficientes para rejeitar a hipótese nula**.

Para iniciar um teste de hipótese é importante que as hipóteses nula e alternativa sejam escolhidas corretamente.

Se você deseja testar uma situação pré-estabelecida ou uma determinada afirmação, esta afirmação deverá ser a Hipótese nula, ou seja, H0.

Se você deseja obter uma evidência para suportar uma afirmação feita por você, então, você deve escolher a Hipótese alternativa, ou seja, HA.

O espaço de hipótese precisa conter todas as variações dos parâmetros no algoritmo de ML.

Análise de Regressão:

Variável Dependente e Independente

Uma variável independente x explica a variação em outra variável, que é chamada variável dependente y. Este relacionamento existe em apenas uma direção:

variável independente (x) → variável dependente (y)

Por exemplo: A quantidade de quilômetros rodados de um carro, seria uma variável independente e o preço do carro seria uma variável dependente.

Este relacionamento não funciona em modo reverso, ou seja, se alterarmos o preço do carro, a quantidade de quilômetros rodados não será alterada.

Uma das preocupações estatísticas ao analisar dados é a de criar modelos que expliquem estruturas do fenômeno em observação. E o **modelo de regressão** é um dos métodos estatísticos mais usados para investigar a relação entre variáveis.

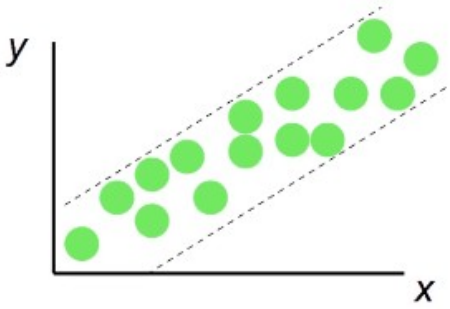
E temos algumas técnicas de análise estatística principais para estudar este relacionamento:

- **Teste de hipótese** para validar uma **suposição** no relacionamento entre variáveis.
- **Análise de correlação**, que determina a **força e direção** do relacionamento entre duas variáveis.
- **Regressão linear simples**, que descreve o relacionamento entre duas variáveis usando uma **equação**.

Correlação:

A análise de correlação nos permite medir a força e direção de um relacionamento linear entre duas variáveis. O coeficiente de correlação indica o grau de associação entre duas variáveis.

O relacionamento entre duas variáveis é linear, se o gráfico de dispersão entre elas tem o padrão de uma linha reta. Exemplo de relação linear:

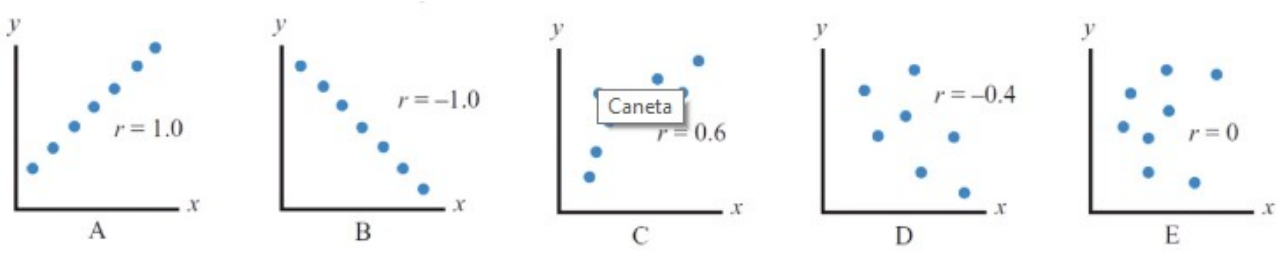


Relacionamento **positivo**, inclinação se move para **cima**.

Relacionamento **negativo**, inclinação se move para **baixo**.

O **coeficiente de correlação (r)** indica a força e direção de uma relação linear entre a variável independente e dependente.

Exemplos de valores de r:



- Gráfico A (r = 1.0): correlação positiva perfeita entre x e y
- Gráfico B (r = -1.0): correlação negativa perfeita entre x e y
- Gráfico C (r = 0.6): relação positiva moderada: y tende a aumentar se x aumenta, mas não necessariamente na mesma taxa observada no Gráfico A
- Gráfico D (r = -0.4): relação negativa fraca: o coeficiente de correlação é próximo de zero ou negativo: y tende a diminuir se x aumenta
- Gráfico E (r = 0): Sem relação entre x e y

Os valores de r variam entre -1.0 (uma forte relação negativa) até +1.0, uma forte relação positiva.

A **correlação**, isto é, a ligação entre dois eventos, **não implica** necessariamente uma relação de **causalidade**, ou seja, que um dos eventos tenha causado a ocorrência do outro.

A correlação pode, no entanto, indicar possíveis causas ou áreas para um estudo mais aprofundado, ou seja, a correlação pode ser uma pista. A ideia oposta, de que correlação prova automaticamente causalidade, é uma falácia lógica.

Obviamente, dois eventos que possuam de fato uma relação de causalidade deverão apresentar também uma correlação. **O que constitui a falácia é o salto imediato para a conclusão de causalidade**, sem que esta seja devidamente demonstrada.

Só porque (A) acontece juntamente com (B) não significa que (A) causa (B).

É necessário investigação adicional em função de diferentes cenários que podem ocorrer:

1. (A) causa realmente (B);
2. (B) pode ser a causa de (A);
3. Um terceiro fator (C) pode ser causa tanto de (A) como de (B);
4. Pode ser uma combinação das três situações anteriores: (A) causa (B) e ao mesmo tempo (B) causa também (A);
5. A correlação pode ser apenas uma coincidência, ou seja, os dois eventos não tem qualquer relação para além do fato de ocorrerem ao mesmo tempo. (Se estivermos falando de um estudo científico, utilizar uma amostra grande ajuda a reduzir a probabilidade de coincidência).

Então como se determina a causalidade?

Depende sobretudo da complexidade do problema, mas a verdade é que a causalidade dificilmente poderá ser determinada com certeza absoluta.

O **conjunto de técnicas de regressão** é provavelmente um dos mais utilizados em análise de dados.

Existem diversos modelos de regressão:

- Regressão Linear Simples e Múltipla
- Regressão Logística Binária
- Regressão Logística Multinomial
- Regressão Poisson
- Regressão Binomial
- Regressão Ridge
- Regressão Lasso
- Regressão ElasticNet

Os modelos de **regressão linear simples e múltipla** são os **mais utilizadas** em diversos campos do conhecimento.

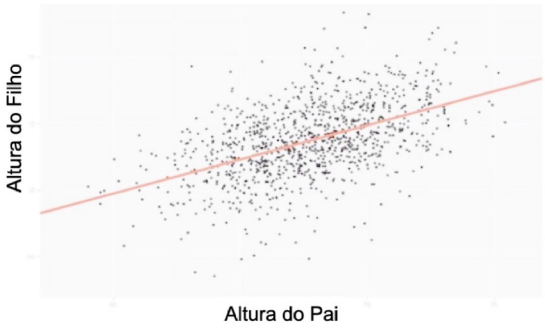
Análise de regressão é uma **metodologia estatística** que utiliza a relação entre duas ou mais **variáveis quantitativas** de tal forma que uma variável possa ser predita a partir de outra.

Análise de regressão é utilizada para se fazer a **previsão de resultados**. O caso **mais simples** de regressão é quando temos **duas variáveis** e a relação entre elas pode ser representada por uma **linha reta**.

A interpretação moderna da regressão é diferente – ocupa-se do estudo da dependência de uma variável (chamada variável **endógena**, resposta ou dependente), em relação a uma ou mais variáveis, (chamadas variáveis explicativas ou **exógenas**), com o objetivo de estimar e/ou prever a média (da população) ou valor médio de uma variável dependente em termos dos valores conhecidos ou fixos (em amostragem repetida) das variáveis independentes.

Fenômeno de Regressão:

Regressão Linear:



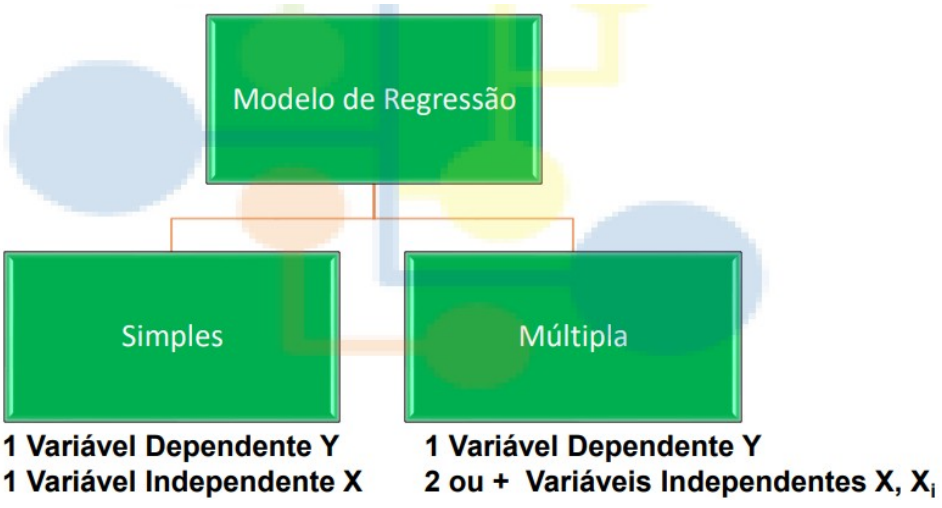
REGRESSÃO e CORRELAÇÃO são a mesma coisa? Não!

Análise de Regressão – prevê o **valor médio** de uma variável com base nos valores estabelecidos de uma ou mais variáveis.

Análise de Correlação – tem como objetivo medir o **grau de associação linear** entre duas variáveis.

Ou seja, usamos a correlação para medir o grau de relação entre duas variáveis e depois usamos regressão para estudar o relacionamento entre elas.

Tipos de Modelos de Regressão Linear:



Regressão Linear Simples:

Nós já sabemos que o coeficiente de correlação **r** nos provê uma medida que descreve a força e direção do relacionamento entre duas variáveis.

Nosso próximo passo é realizar uma **análise de regressão linear simples**, que nos habilite descrever uma **linha reta** que melhor representa uma **série de pares ordenados (x, y)**.

Como veremos mais adiante, ter uma linha reta que descreve o relacionamento entre a variável independente (x) e a variável dependente (y) nos oferece uma série de vantagens sobre o coeficiente de correlação.

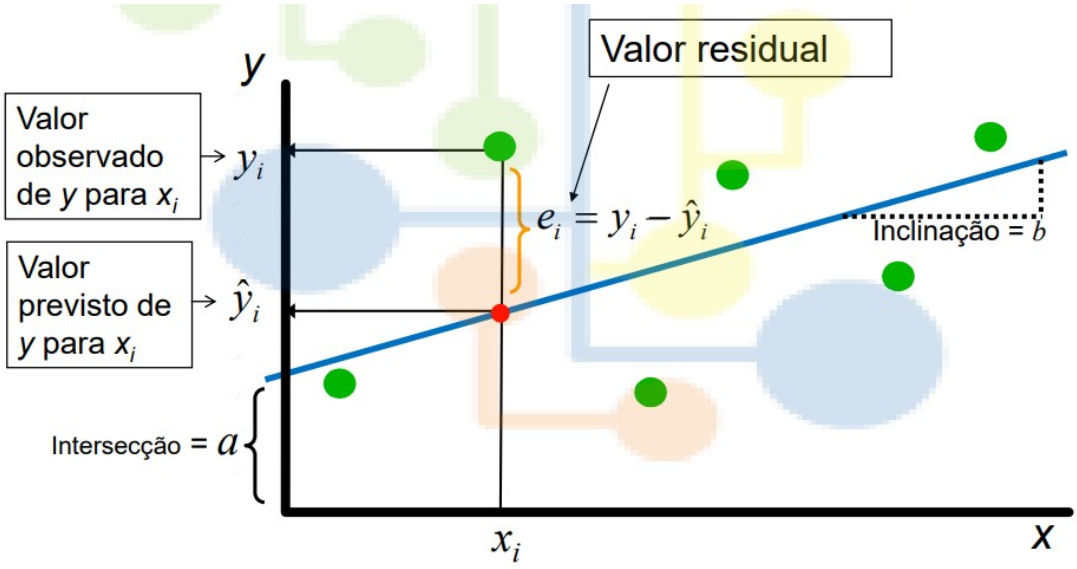
Fórmula para a equação que descreve uma linha reta através de um par ordenado:

$$\hat{y} = a + bx$$

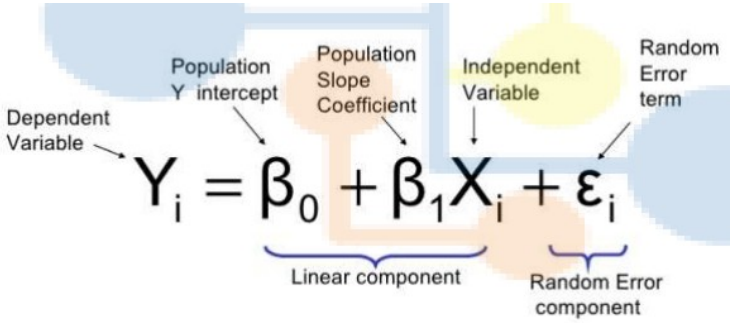
Onde:

$\hat{y}$ = valor previsto de *y* dado um valor para *x*
x = variável independente
a = ponto onde a linha intercepta o eixo y
b = inclinação da linha reta

Exemplo:



Fórmula de Regressão Linear Simples (2 variáveis):



Fórmula de Regressão Linear Múltipla (Mais de 2 variáveis):

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta_{p-1} X_{i,p-1} + \varepsilon_i$$

Para determinarmos a equação que melhor se ajusta a nuvem de pontos, devemos estabelecer duas condições fundamentais aos resíduos:

1. A somatória dos resíduos deve ser 0.
2. A somatória dos resíduos ao quadrado é a mínima possível.

Deve-se determinar α e β de modo que a somatória dos quadrados dos resíduos seja a menor possível (método de Mínimos Quadrados Ordinários – MQO, ou, em inglês, Ordinary Least Squares – OLS)

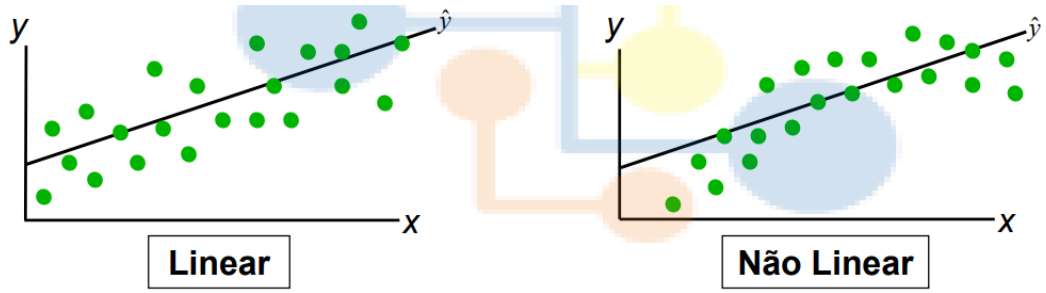
Premissas da Análise de Regressão:

Como estamos trabalhando com dados da amostra para fazer previsões sobre a população, como podemos saber qual é o nível de precisão nas predições que fazemos usando regressão linear?

Para responder a esta pergunta, precisamos construir um **intervalo de confiança**. Os intervalos de confiança oferecem uma estimativa do parâmetro da população, baseado na estatística da amostra.

Para que os resultados de uma análise de regressão sejam confiáveis, algumas **premissas** precisam ser satisfeitas:

- Premissa 1: O relacionamento entre a variável independente e a variável dependente deve ser linear.



- Premissa 2: O valor residual não deve exibir um padrão através da variável independente.

$$e_i = y_i - \hat{y}_i$$

Armadilhas da Análise de Regressão:

- Não faça previsões para a variável dependente (y), além do range de valores da variável independente (x).
- Não há garantia que o relacionamento estimado é apropriado além do range que foi observado.
- Não confunda correlação com causalidade.

Apesar do relacionamento entre as variáveis ser estatisticamente significante, não prova que a variável independente realmente causou a mudança na variável dependente.

Considerações finais:

A Ética deve estar sempre associada a estatística e os dados devem ser tratados com muito cuidado, para que não se leve a conclusões que não correspondem a realidade.

O papel do Gestor na análise de dados é tomar as decisões baseadas em dados estatísticos.

3.11. Machine Learning em Linguagem R

Introdução ao Aprendizado de Máquina (Machine Learning):

O termo Machine Learning (ou aprendizado de máquina em português) possui atualmente as mais variadas definições, especialmente depois de tantos filmes sobre robôs e Inteligência Artificial, que transformaram Machine Learning em algo que realmente não é.

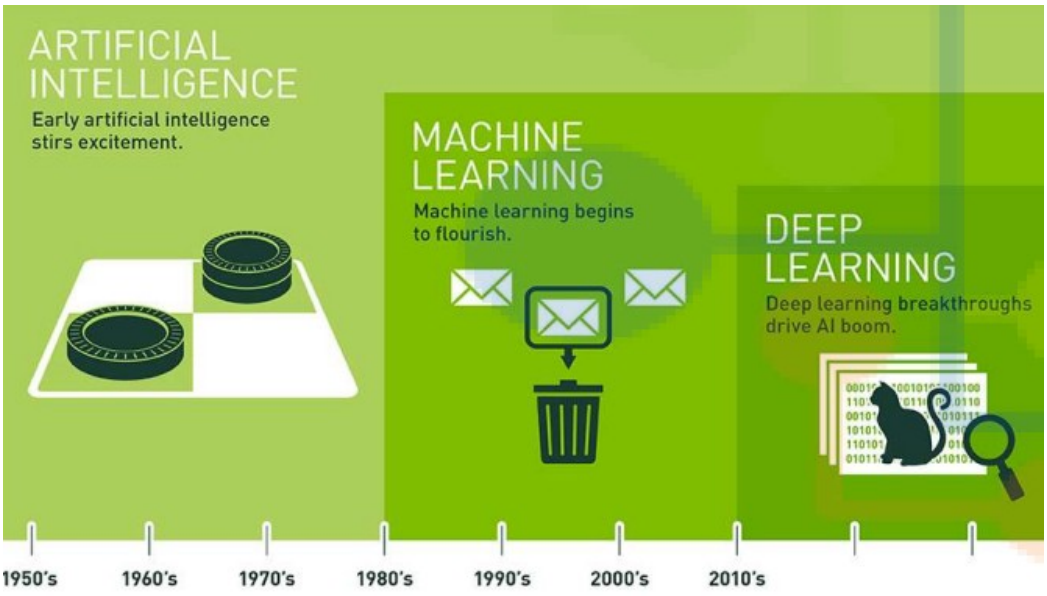
O que é Machine Learning?

Machine Learning é o método de análise de dados que automatiza a construção de modelos analíticos.

Machine Learning pode realizar análises preditivas mais rápido que qualquer humano seria capaz de fazer!

Então Machine Learning e IA são conceitos diferentes?

Machine Learning é um subconjunto da Inteligência Artificial.



Inteligência Artificial inclui Machine Learning, mas Machine Learning por si só não define Inteligência Artificial.

Inteligência Artificial é baseada em Machine Learning e Machine Learning é essencialmente diferente de Estatística.

Técnica	**Estatística**	**Machine Learning**
Entrada de Dados	Os parâmetros interpretam fenômenos da vida real e trabalham a magnitude.	Os dados são randomizados e transformados para aumentar a acurácia de análises preditivas.
Tratamento de Dados	Modelos são usados para previsões em amostras pequenas.	Trabalha com Big Data na forma de redes e grafos. Os dados são

		divididos em dados de treino e dados de teste.
Resultado	Captura a variabilidade e a incerteza dos parâmetros.	Probabilidade é usada para comparações e para buscar as melhores decisões.
Distribuição dos Dados	Assumimos uma distribuição bem definida dos dados.	A distribuição dos dados é desconhecida ou ignorada antes do processo de aprendizagem
Objetivos	Assumimos um determinado resultado e então tentamos prová-lo.	Os algoritmos aprendem a partir dos dados.

Machine Learning se baseia em alguns importantes conceitos da Matemática, Estatística e Ciência da Computação:

- Matemática:
  - Manipulação de Matrizes
- Estatística:
  - Teoria da Probabilidade e Inferência Estatística
- Ciência da Computação:
  - Programação
  - Armazenamento e Processamento de Dados

O Que São Algoritmos?

Machine Learning usa algoritmos para analisar grandes conjuntos de dados!

Algoritmos são procedimentos ou fórmulas usados para resolver problemas. Procedimentos podem ser entendidos como uma lista ou sequência de ações a serem seguidas.

O tipo de problema a ser resolvido, determina o tipo de algoritmo a ser utilizado.

Falhas são mais comuns que sucesso em processos de Machine Learning.

Para que um algoritmo possa aprender, você precisa treiná-lo.

Assim como você não precisa ver todas as árvores do mundo para aprender a reconhecer uma, os algoritmos de aprendizado de máquina podem usar o poder computacional dos computadores e a ampla disponibilidade de dados sobre os mais diversos tipos de fenômenos, para resolver um grande número de problemas, sem a necessidade de ter acesso a todos os dados.

Machine Learning Frameworks

Para criar modelos de Machine Learning você tem duas opções:

- Desenvolver os algoritmos a partir do zero
- Utilizar Frameworks prontos

Um framework é um conjunto de softwares que produzem um resultado específico. Um framework nos permite focar mais no problema de negócio e menos na parte de codificação.

Frameworks de Machine Learning permitem que você trabalhe em um problema, sem ter que saber muito sobre programação (embora seja altamente recomendável que você conheça bem sobre programação).

O framework cuida da gestão de infraestrutura, enquanto você pode focar mais na parte inteligente da sua aplicação.

E por que usar Machine Learning Frameworks?

Com o crescimento do volume de dados e a complexidade do Big Data, os cientistas de dados estão consumindo mais tempo apoiando a infraestrutura em vez de construir modelos para resolver seus problemas de dados.

Principais Machine Learning Frameworks

- Linguagem R (Pacote caret)
- Microsoft Azure Machine Learning
- Scikit-Learn (Linguagem Python)
- Apache Spark MLlib
- Google Tensor Flow
- Keras (Tensor Flow)
- Caffe
- CNTK Microsoft
- Mxnet Amazon
- KNIME
- rapidminer

Tipos de Aprendizagem em Machine Learning

O Processo de Aprendizagem ocorre de diferentes formas e podemos dividir os algoritmos de Machine Learning em 3 grupos principais:

- Aprendizagem Supervisionada:
    - É o termo usado sempre que o algoritmo é “treinado” sobre um conjunto de dados históricos contendo entradas e saídas.

    - Baseado no treinamento com os dados históricos, o modelo pode tomar decisões precisas quando recebe novos dados.
- Aprendizagem Não Supervisionada:
    - A aprendizagem não supervisionada ocorre quando um algoritmo aprende com exemplos simples, sem qualquer resposta associada, deixando a cargo do algoritmo determinar os padrões de dados por conta própria.
- Aprendizagem Por Reforço:
    - O conceito de Aprendizagem Por Reforço (Reinforcement Learning) é como aprender por tentativa e erro. Os erros ajudam a aprender, porque eles têm uma grande penalidade associada a eles (custo, perda de tempo e assim por diante), ensinando que um determinado curso de ação tem menor probabilidade de êxito do que outros.

Regressão X Classificação

Aprendizagem Supervisionada

A aprendizagem supervisionada ocorre quando um algoritmo aprende a partir de dados históricos de exemplo, com entradas (inputs) e possíveis saídas (outputs), que podem consistir em valores quantitativos ou qualitativos, a fim de prever a resposta correta quando recebe novos dados.

Categorias de Aprendizagem Supervisionada:

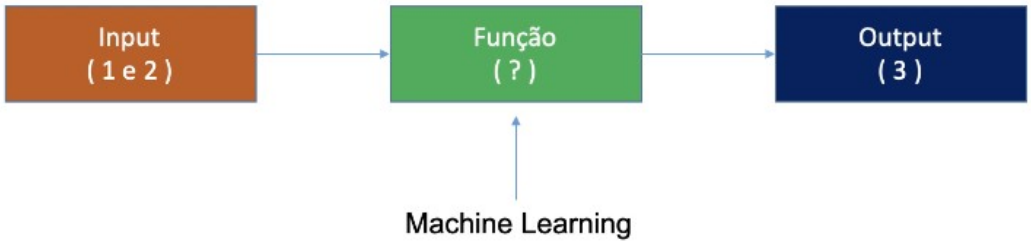
- Regressão: previsão de valores numéricos.
- Classificação: atribuição de rótulo à saída.

O Processo de Aprendizagem em Machine Learning

Um algoritmo de ML, como um algoritmo de classificação por exemplo, funciona da mesma forma. Ele constrói suas capacidades cognitivas através da criação de uma formulação matemática que inclui todas as características dadas sobre um determinado fenômeno.

O Processo de Aprendizagem ocorre de diferentes formas e podemos dividir os algoritmos de Machine Learning em 3 grupos principais: Aprendizagem Supervisionada, Aprendizagem Não Supervisionada e Aprendizagem Por Reforço.

Do ponto de vista matemático, você pode expressar o processo de representação no aprendizado de máquina utilizando o mapeamento equivalente.



O Processo de Aprendizagem em Detalhes – Parte 1

Um componente chave do processo de aprendizagem é a generalização!

E para poder generalizar a função que melhor resolve o problema, os algoritmos de Machine Learning se baseiam em 3 componentes:

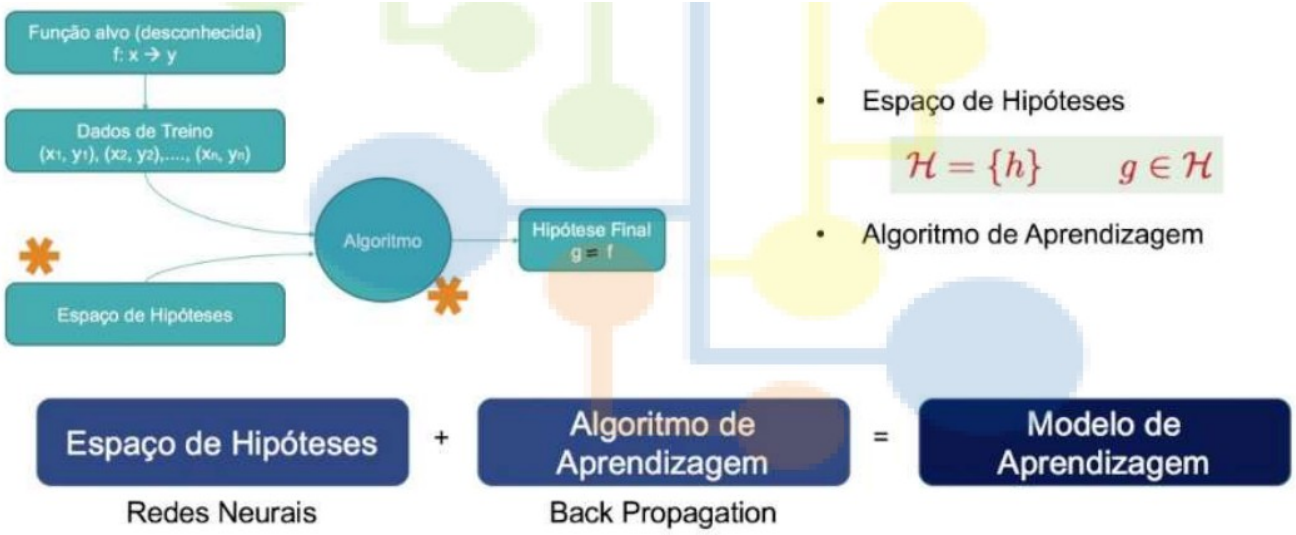
- Representação
- Avaliação
- Otimização

Os algoritmos de aprendizagem possuem diversos parâmetros internos (valores separados em vetores e matrizes).

Esses parâmetros funcionam como uma espécie de memória para o algoritmo, permitindo que o mapeamento ocorra e as características analisadas sejam conectadas.

As dimensões e tipos de parâmetros internos delimitam o tipo de funções-alvo que um algoritmo pode aprender. O engine de otimização no algoritmo muda os valores iniciais dos parâmetros durante a aprendizagem para representar função-alvo.

O Processo de Aprendizagem em Detalhes – Parte 2



Falso positivo é quando um algoritmo parece ter encontrado a melhor solução, mas ela não resolve o problema. Isso ocorre, normalmente, quando a quantidade de dados foi insuficiente ou houve algum erro na preparação dos dados.

Big Data é uma grande mistura de dados. Um bom algoritmo de Machine Learning deve ser capaz de distinguir os sinais e mapear as funções alvo de forma eficiente.

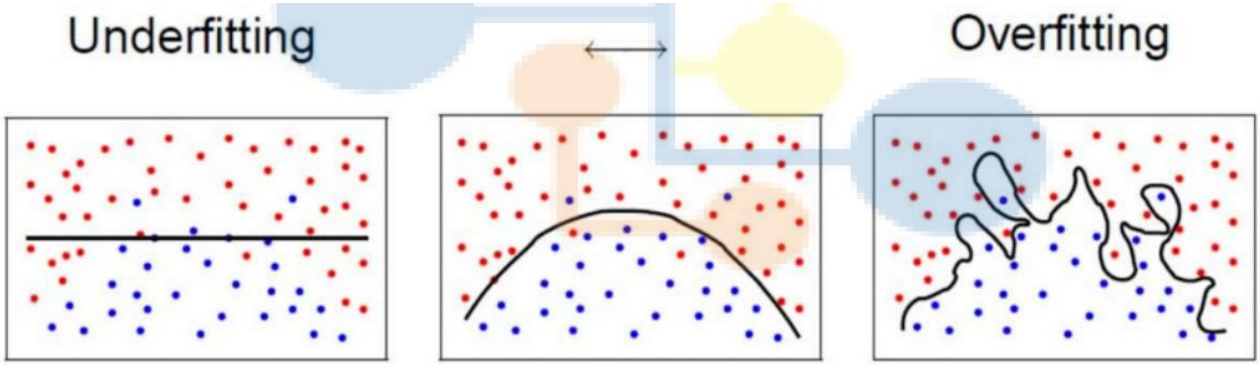
Cost Function é uma função de avaliação que mede quão bem o algoritmo mapeia a função alvo, aquela que ele está tentando deduzir, a partir de dados que você mesmo forneceu.

As técnicas de aprendizagem de máquina, baseadas em algoritmos, estatísticos utilizam Cálculo e Álgebra Linear e os dados precisam estar carregados em memória.

O Processo de Aprendizagem em Detalhes – Parte 3

O processo de ”fitting” um modelo a um dataset é chamado de treinamento do modelo.

O modelo pode aprender demais (overfitting) ou aprender de menos (underfitting).



Com o overfiting o modelo deixa de fazer generalização.

Para atingir o equilíbrio e criar grandes soluções de Machine Learning, você terá que fazer escolhas, achar o ponto ideal entre um modelo mais simples ou outro mais complexo.

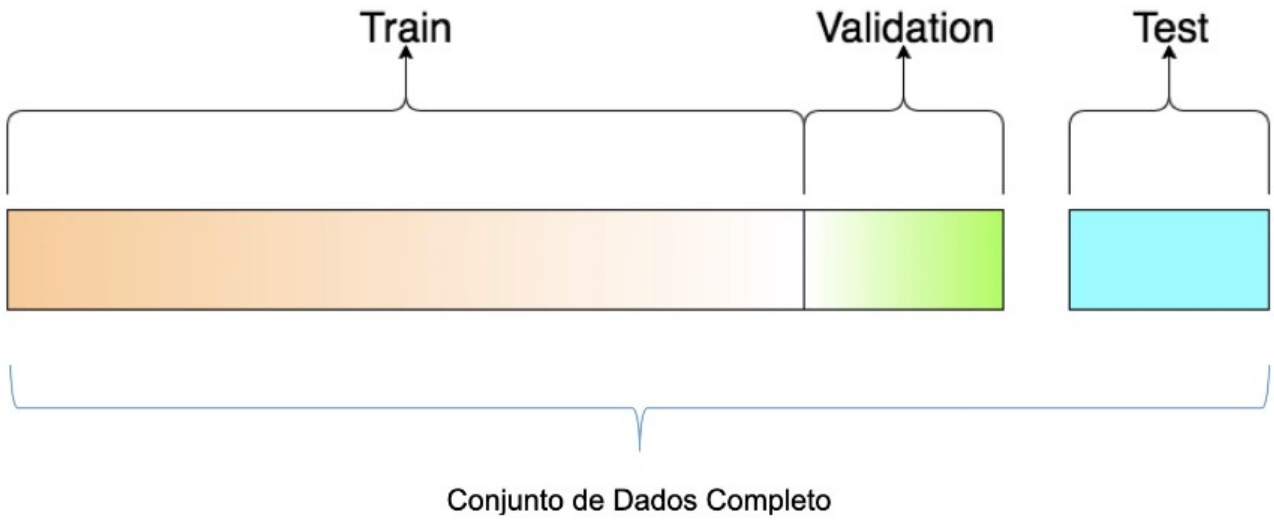
Para visualizar se os seus algoritmos de Machine Learning estão sofrendo algum tipo de força tendenciosa, você pode usar um gráfico chamado Curva de Aprendizagem.

Para usar uma curva de aprendizagem, você precisa:

1. Dividir seus dados em amostras, chamadas dados de treino e dados de teste (uma divisão 70/30 funciona bem). Dados de validação podem ser usados durante o treinamento.
2. Criar porções dos seus dados de treino, com tamanhos diferentes a cada passagem de treino.
3. Treinar seus modelos com os diferentes subsets. Registrar a performance.
4. Gerar um gráfico com os resultados. Atenção aos intervalos de confiança e ao desvio padrão.

Podemos criar curvas de aprendizagem em R de diversas formas, usando os pacotes mlr, caret ou mesmo o ggplot2.

Treinamento, Validação e Teste



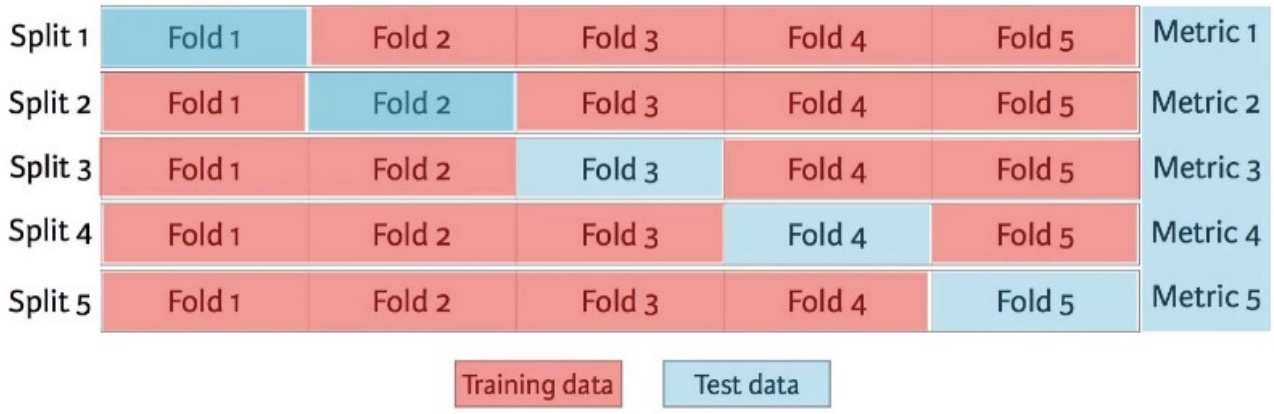
Normalmente é feita a seguinte divisão:

- 75 a 70% - dados de treino e 25 a 30% - dados de teste
- 70% - dados de treino, 20% - dados de validação e 10% - dados teste

É recomendado que essa separação seja feita de maneira aleatória, independentemente da ordenação dos dados, para evitar underfitting ou overfitting.

Cross-Validation

O Cross-Validation também pode ser usado para se obter um resultado melhor na generalização do seu modelo:

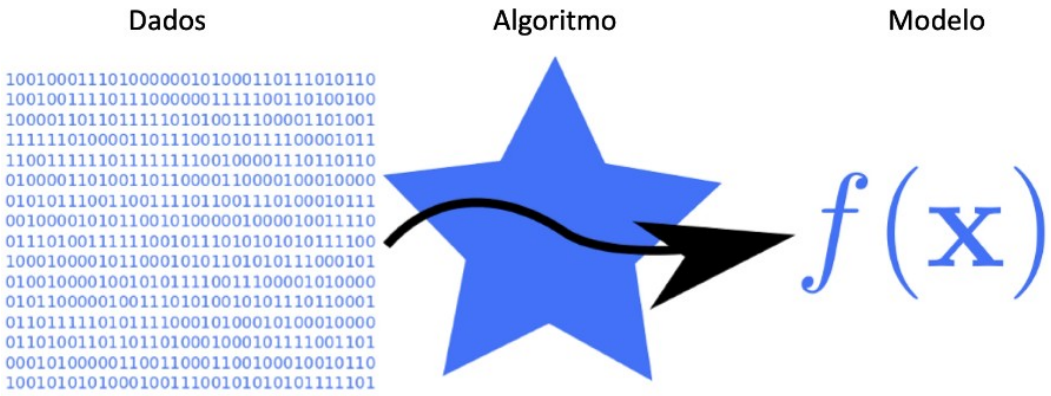


O conceito central das técnicas de validação cruzada é o particionamento do conjunto de dados em subconjuntos mutuamente exclusivos, e posteriormente, utiliza-se alguns destes subconjuntos para a estimação dos parâmetros do modelo (dados de treinamento) e o restante dos subconjuntos (dados de validação ou de teste) são empregados na validação do modelo.

O Que é um Modelo de Machine Learning?

Como já vimos, a aprendizagem de máquina é um subcampo da Inteligência Artificial que evoluiu a partir do estudo de reconhecimento de padrões e teoria da aprendizagem computacional.

Machine Learning é um campo de estudo que dá ao computador a capacidade de aprender, sem ser programado de forma explícita.



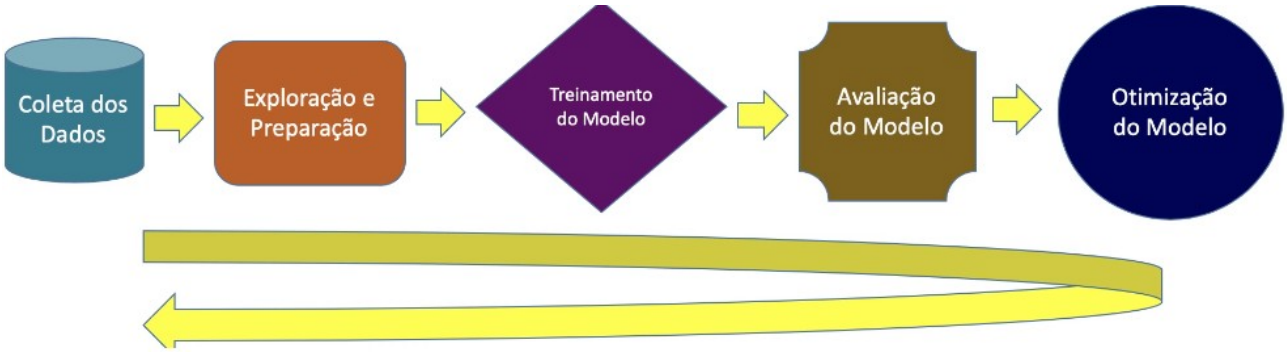
Modelo:

Existem muitos tipos diferentes de modelos. Você pode já estar familiarizado com alguns. Os exemplos incluem:

- Equações matemáticas
- Diagramas relacionais
- Agrupamentos de dados, conhecidos como clusters

Criação do Modelo:

(Definição do Problema de Negócio) > Coleta dos Dados > Exploração e Preparação > Treinamento do Modelo > Avaliação do Modelo > Otimização do Modelo

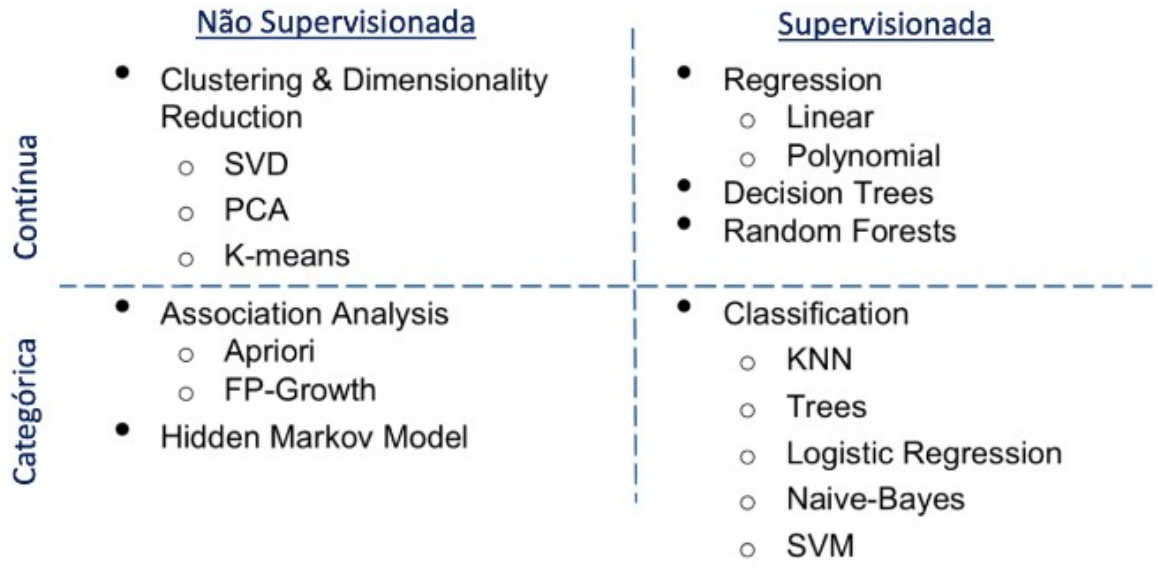


Este é um trabalho iterativo e assim como um surfista está sempre em busca da onda perfeita, seu trabalho como Cientista de Dados é buscar sempre o melhor modelo possível para suas previsões. Defina sua métrica com antecedência e trabalhe para alcançá-la (cuidado com perfeccionismo).

Lembre-se: um modelo de Machine Learning será usado para resolver um problema específico! Não caia na tentação de querer aplicar seu modelo a tudo que você vê pela frente. Cada problema de negócio, cada conjunto de dados, pode requerer um modelo diferente.

Algoritmos de Machine Learning – Parte 1

- Aprendizagem Supervisionada
  - Classificação
  - Regressão
- Aprendizagem Não Supervisionada
  - Clustering
  - Segmentação
  - Redução de Dimensionalidade
- Aprendizagem por Reforço
  - Sistemas de Recomendação
  - Sistemas de Recompensa
  - Processo de Decisão



Há tantos algoritmos disponíveis com tantos métodos diferentes, que somente o processo de escolha de qual deve ser usado, já vai consumir bastante do seu tempo como Cientista de Dados.

Podemos categorizar os algoritmos de Machine Learning em 2 grupos principais:

- Estilo de Aprendizagem
- Similaridade (Funcionalidade)

Algoritmos de Machine Learning – Parte 2

- Algoritmos de Regressão

    - Ordinary Least Squares Regression (OLSR)
    - Linear Regression
    - Logistic Regression
    - Stepwise Regression
    - Multivariate Adaptive Regression Splines (MARS)
    - Locally Estimated Scatterplot Smoothing (LOESS)
- Algoritmos Regulatórios
    - Ridge Regression
    - Least Absolute Shrinkage and Selection Operator (LASSO)
    - Elastic Net
    - Least-Angle Regression (LARS)
- Algoritmos Baseados em Instância (Instance-based)
    - k-Nearest Neighbour (kNN)
    - Learning Vector Quantization (LVQ)
    - Self-Organizing Map (SOM)
    - Locally Weighted Learning (LWL)
- Algoritmos de Árvore de Decisão
    - Classification and Regression Tree (CART)
    - Conditional Decision Trees
    - Iterative Dichotomiser 3 (ID3)
    - C4.5 and C5.0 (different versions of a powerful approach)
    - Chi-squared Automatic Interaction Detection (CHAID)
    - Decision Stump
    - M5
- Algoritmos Bayesianos
    - Naive Bayes
    - Gaussian Naive Bayes
    - Multinomial Naive Bayes
    - Averaged One-Dependence Estimators (AODE)
    - Bayesian Belief Network (BBN)
    - Bayesian Network (BN)
- Algoritmos de Clustering

    - k-Means
    - k-Medians
    - Expectation Maximisation (EM)
    - Hierarchical Clustering
- Algoritmos Baseados em Regras de Associação
    - Apriori algorithm
    - Eclat algorithm
- Redes Neurais Artificiais
    - Perceptron
    - Back-Propagation
    - Hopfield Network
    - Radial Basis Function Network (RBFN)
- Deep Learning
    - Deep Boltzmann Machine (DBM)
    - Deep Belief Networks (DBN)
    - Convolutional Neural Network (CNN)
    - Stacked Auto-Encoders
- Algoritmos de Redução de Dimensionalidade
    - Principal Component Analysis (PCA)
    - Principal Component Regression (PCR)
    - Partial Least Squares Regression (PLSR)
    - Multidimensional Scaling (MDS)
    - Projection Pursuit
    - Linear Discriminant Analysis (LDA)
    - Mixture Discriminant Analysis (MDA)
    - Quadratic Discriminant Analysis (QDA)
    - Flexible Discriminant Analysis (FDA)
- Algoritmos Ensemble
    - Boosting
    - Bootstrapped Aggregation (Bagging)
    - AdaBoost
    - Stacked Generalization (blending)

    - Gradient Boosting Machines (GBM)
    - Gradient Boosted Regression Trees (GBRT)
    - Random Forest
- Outros Algoritmos
    - Support Vector Machines
    - Computer Vision (CV)
    - Natural Language Processing (NLP)
    - Recommender Systems
    - Graphical Models

A Importância da Análise Exploratória de Dados

Análise Exploratória é simplesmente conhecer os dados.

A qualidade de qualquer projeto de aprendizagem de máquina é baseada principalmente na qualidade dos seus dados de entrada.

A análise exploratória de dados tem como um dos objetivos identificar as variáveis que são melhores candidatas a variáveis preditoras.

3.12. Microsoft Azure Machine Learning

O que é Microsoft Azure Machine Learning?

Microsoft Azure Machine Learning é um framework de Machine Learning que pode ser usado para construir modelos de análise preditiva, através de conjuntos de dados de treinamento e uma variedade de fontes de dados.

Machine Learning em nuvem (cloud).

O Microsoft Azure Machine Learning oferece uma forma rápida de trabalhar com análise preditiva!

Análise Preditiva não é adivinhação. É ciência sendo usada para extrair previsões a partir dos dados!

O principal conceito por trás do Azure e o qual a Microsoft gosta de reforçar, é o fail fast, ou em português, cometa erros mais rapidamente.

O Azure Machine Learning não é gratuito!

Microsoft Azure Machine Learning Studio

Uma característica interessante do Microsoft Azure Machine Learning Studio é que você pode trabalhar com linguagens como R, Python e SQL.

Não confunda o Microsoft Azure Machine Learning com o Microsoft Azure. Azure Machine Learning e Azure não são a mesma coisa. O primeiro é o serviço de Machine Learning na nuvem Microsoft, chamada Azure. O segundo, é uma ferramenta do primeiro.

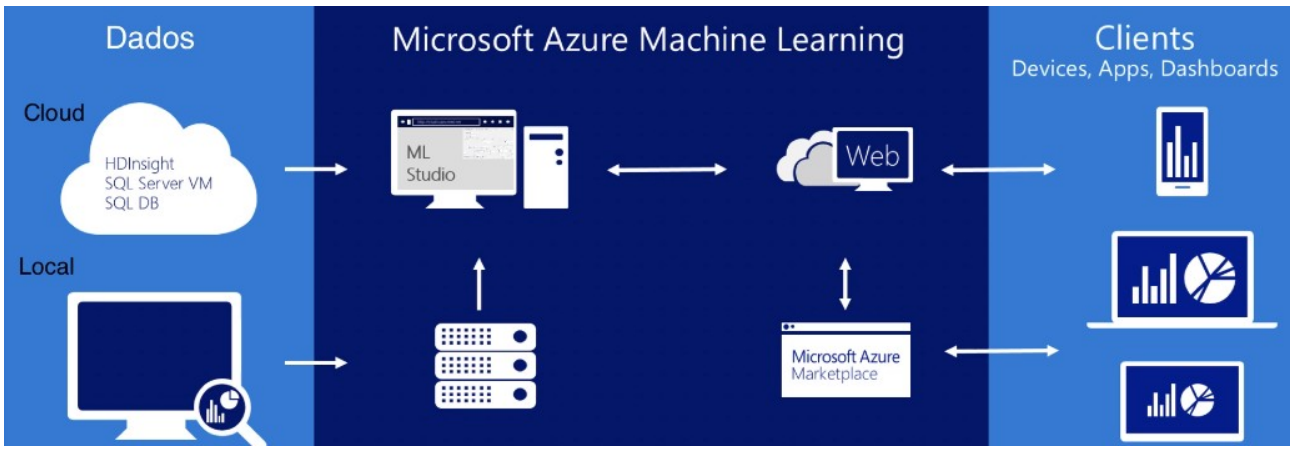
Mas não pense que as facilidades trazidas pelo Azure Machine Learning, significam que você não tenha que conhecer bem os algoritmos de Machine Learning e teorias necessárias para compreender claramente o que pode ser feito com o Azure ML.

Azure Machine Learning Workflow

Uma das vantagens de se utilizar o Microsoft Azure Machine Learning é a possibilidade de se criar um workflow, ou um fluxo de trabalho.

Azure Machine Learning Workflow:

Dados > Preparação > Exploração > Criação do Modelo > Avaliação e Otimização > Deploy



Azure Machine Learning Toolkit

Azure Machine Learning Toolkit é um conjunto de ferramentas oferecido no Azure Machine Learning Studio para todo o processo de modelagem preditiva!

Ferramentas de ML oferecidas pelo Toolkit:

- **Classificação** – algoritmos usados para classificar dados em diferentes categorias e então fazer previsões sobre dados com variáveis qualitativas.
- **Regressão** – algoritmos usados para pever uma ou mais variáveis numéricas.
- **Clustering** – algoritmos usados para identificar grupos ou padrões em conjuntos de dados e então fazer previsões de classificação em grupos, para uma determinada variável (aprendizagem não supervisionada).

Acesso ao Azure Machine Learning Studio

Para acessar o Azure ML sem a necessidade de cadastrar cartão de crédito, use a interface clássica:

https://studio.azureml.net/

Para criar uma conta, acesse:

https://azure.microsoft.com/pt-br/services/machine-learning

Clique na opção "Sign In" para um Free Workspace (a opção em verde). Você precisa ter uma conta Microsoft (msn, outlook).

Obs: Aversão Free Workspace não possui limitação de tempo de uso e sua conta nunca expira. As limitações são a capacidade de armazenamento (10GB), capacidade de processamento e limite a alguns serviços. Desta forma, você pode usar o Azure Machine Learning e testar seus experimentos sem se preocupar com tempo de utilização!

Documentação em português:

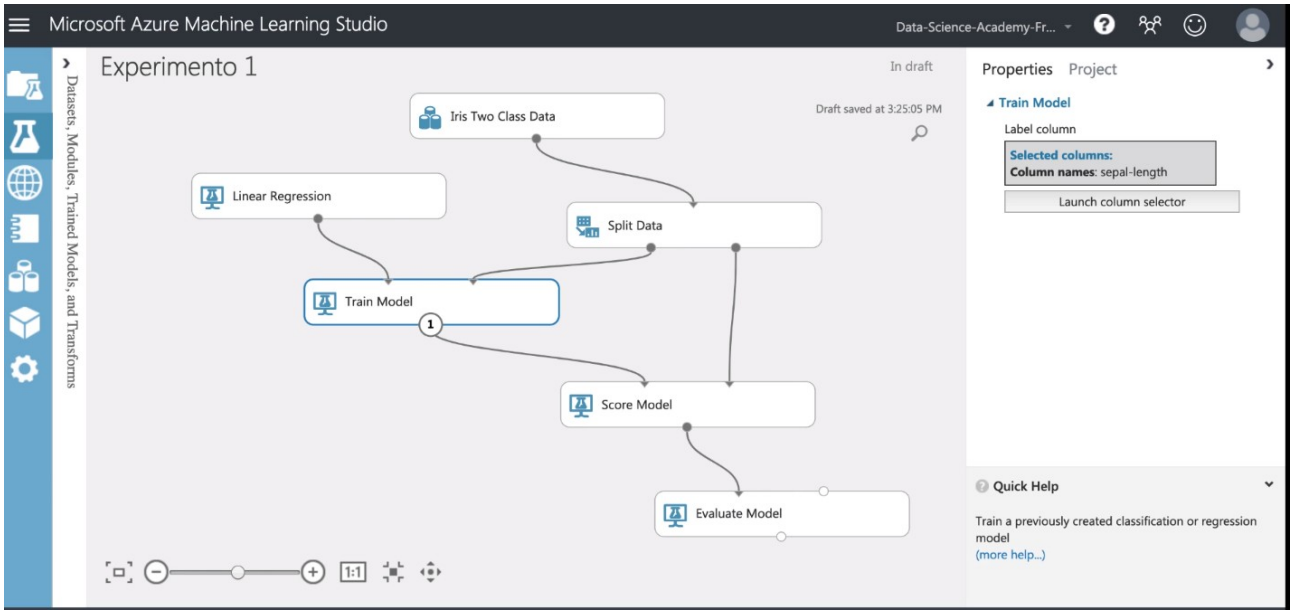
https://docs.microsoft.com/pt-br/azure/machine-learning

Menu do Azure ML Studio:

- experiments: fluxo de trabalho ou workflow
  - my experiments
  - samples – exemplos disponibilizados pela Microsoft
- web services: deploy do modelo em produção
- notebooks: sequencia de operações (python)
- datasets: arquivos ou dados
  - my datasets
  - samples – exemplos disponibilizados pela Microsoft
- trained models: modelos treinados
- settings: configurações
  - workspace name
  - workspace description
  - workspace type
  - workspace storage
  - delete workspace (na versão gratuita só permite 1)

Tela de novo experimento:

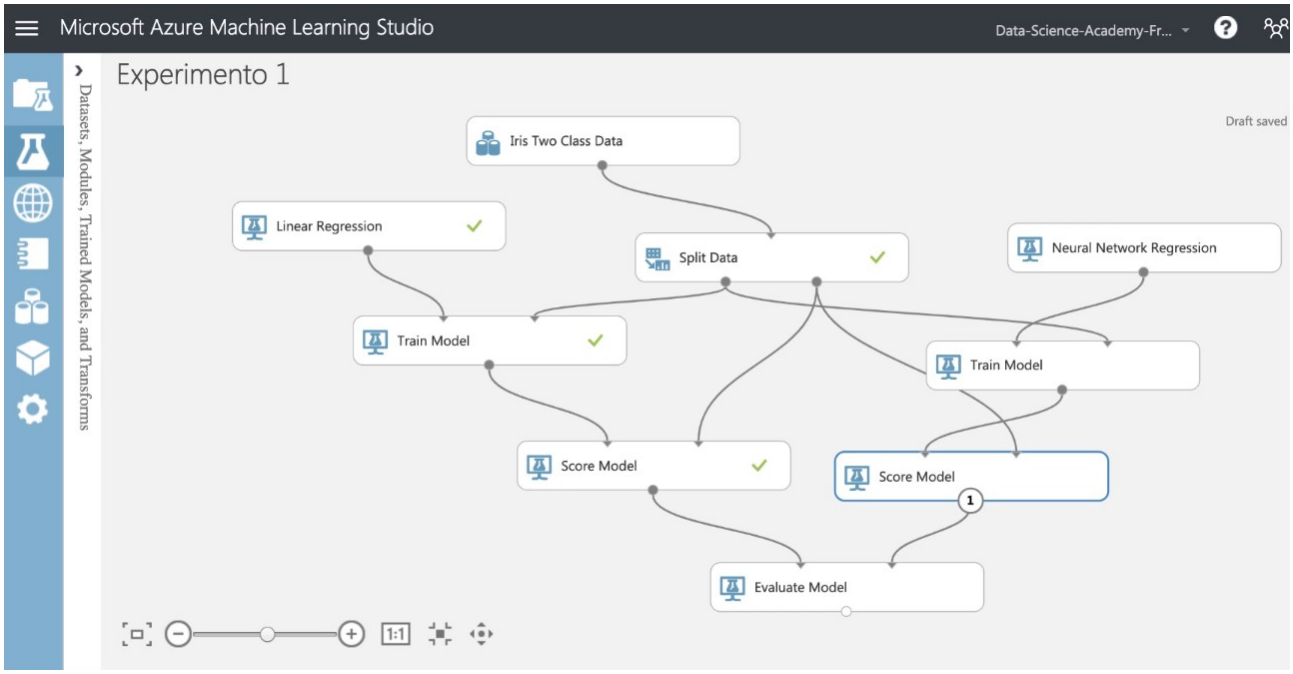
Exemplo de fluxo de trabalho:



Utilizando a busca (não precisa decorar o menu):

- Data set – fonte de dados
- Split data – separar entre dados de treino e de teste
- Linear Regression – nesse caso, foi utilizado algoritmo de regressão linear
- Train Model – treinamento do modelo
- Score Model – aplicar o modelo aos dados de teste
- Evaluate Model – avaliar o modelo para ver seu nível de performance

Exemplo para realizar comparação de modelos:



3.13. Data Munging no Azure Machine Learning

Introdução

Data Munging é uma de etapa manipulação dos dados, do processo de Data Science, realizado antes de treinar um modelo de Machine Learning.

Neste capítulo, será estudado:

- Execução de Scripts R no Azure ML
- Módulos de Manipulação
- Tratamento de Metadados
- Tratamento de Dados Duplicados e Missing
- Transformação e Padronização
- Erros e Outliers
- SQL no Azure ML
- Utilização de Joins para Combinar Datasets
- Pacotes dplyr e tidyr
- Criação de Gráficos com ggplot2 no Azure ML

Executando Script R no Azure ML

Você pode utilizar o módulo "Execute R Script" para aplicar procedimentos escritos em R ao seu experimento de análise de dados.

1. Novo experimento > Selecione um Dataset
2. Na busca digite exec > Selecione Execute R Script
3. O módulo Execute R Script possui 5 portas (3 de entrada e 2 de saída):
    1. E: Dataset1
    2. E: Dataset2
    3. E: Script Bundle (Zip) – arquivo zip com vários scripts em linguagem R
    4. S: Result dataset – resultado do processamento do script
    5. S: R Device (dataset) – permite geração de plot / visualização
4. Em Properties
    1. Execute R Script – com um padrão para funcionalidade do Execute R Script (pode ser alterado)
    2. Random Seed – para que a randomização traga resultados iguais aos do professor
    3. R Version – Versão do R > Selecione CRAN R n.n.n

5. Conecte o conjunto de dados na primeira porta do script R
6. Execute o experimento clicando em Run

Importando Script R no Azure ML

1. Converta o Script R feito no R Studio para Zip: Botão direito > Compressed Zip Folder
2. Suba o arquivo para o Azure: New > Dataset > From Local File > Selecione o arquivo Zip > Coloque uma descrição (se desejar)
3. Localize o arquivo usando a caixa de pesquisa, digitando o nome
4. Arraste o arquivo para dentro do experimento – ao fazer isso, o Azure descompacta o código e põe no diretório “src”
5. Conecte o script importado na porta Script Bundle do Execute R Script – isso é necessário quando o script faz referência a outro script (para buscar uma função, por exemplo)
6. Execute o experimento clicando em Run
7. Em caso de erro: clique com botão direito onde houver um ícone de erro > View log > Output log ou Error log (resumo do erro)

Executando Experimento de Análise de Dados no Azure ML

1. Antes de executar o código: Clique em Execute R Script > em R Script comente linhas de comandos setwd e getwd (são apenas para o R Studio), e Altere a variável flag que controla o local da execução do script
2. Para visualizar o resultado do experimento: Clique com botão direito na porta Result Dataset > Visualize
3. Para visualizar eventuais gráficos contidos no script do experimento: Clique com botão direito na porta R Device (dataset) > Visualize

Editando os Metadados no Azure ML

Metadados são dados sobre os dados. Com esse módulo é possível alterar o tipo das variáveis, o nome das colunas, etc.

1. Inclua um Dataset no experimento
2. Use a caixa de pesquisa e localize Edit Metadata > Arraste para o experimento
3. Conecte o Dataset ao Edit Metadata
4. Selecione Edit Metadata > Clique em Launch column selector > Escolha a variável
5. Data type – altere o tipo de dado
6. Categorical – transforma a variável em categórica ou não categórica

Obs.: Muito usado para tratamento de dados para modelos de classificação!

7. Fields – edita os campos
8. New column names – edita o nome das colunas do dataset.

Tratamento de Dados Duplicados e Missing no Azure ML

Para dados Missing:

1. Inclua um Dataset no experimento
2. Use a caixa de pesquisa e localize Clean Missing Data > Arraste para o experimento
3. Conecte a saída do Dataset ou do script à porta de entrada do módulo de limpeza
4. Clique no módulo > Launch column selector > With Rules > Retire da lista o que não interessa
5. Defina as regras de limpeza
    1. Minimum missing value ratio > 0
    2. Maximum missing value ratio > 1
    3. Cleaning mode – selecione o tipo de tratamento
6. Clique em Run
7. Visualize os resultados: Clique com botão direito na porta desejada

Para dados Duplicados:

8. Use a caixa de pesquisa e localize Remove Duplicate Rows > Arraste para o experimento
9. Conecte a saída do Clean Missing Data à porta de entrada do módulo Remove Duplicate Rows
10. Clique no módulo > Launch column selector > By Name > Inclua na lista o que interessa (normalmente, usa-se atributos chave primária para excluir regs duplicados)
11. Escolha se quiser manter a primeira linha: Clique em Retain first duplicate row
12. Clique em Run
13. Visualize os resultados: Clique com botão direito na porta desejada

Transformação e Padronização no Azure ML

Para Normalização (colocar variáveis numéricas na mesma escala):

14. Use a caixa de pesquisa e localize Normalize Data > Arraste para o experimento
15. Conecte a saída do Remove Duplicate Rows à porta de entrada do módulo Normalize Data
16. Defina as regras de normalização: Clique em Transformation method (clique no help para entender os tipos de normalização)
17. Selecione as colunas: Clique em Launch column selector
18. Clique em Run

19. Visualize os resultados: Clique com botão direito na porta desejada (compare resultado com o Dataset original)

Para Padronização (padronizar variáveis categóricas):

20. Use a caixa de pesquisa e localize Edit Metadata Data > Arraste para o experimento
21. Conecte a saída do Remove Duplicate Rows à porta de entrada do módulo Edit Metadata
22. Selecione Edit Metadata > Clique em Launch column selector > Escolha a variável
23. Em Data type > Selecione String
24. Em Categorical > Selecione Make categorical
25. Em New column names > Digite o novo nome (se desejar)

Treinamento de Erros e Outliers no Azure ML

Outlier é um valor extremo, muito distante da média.

1. Use a caixa de pesquisa e localize Clip Values > Arraste para o experimento
2. Conecte a saída do Dataset à porta de entrada do módulo Clip Values
3. Conecte a saída do Clip Values à porta de entrada do Script R

   (É recomendável limpar os dados antes da transformação dos mesmos)
4. Selecione o módulo > em Set of thresholds tem-se:
   1. ClipPeaks – limpar os picos ou valores extremos mínimo e máximo
   2. ClipSubpeaks – limpar os subvalores dos picos (um pouco acima dos picos)
   3. ClipPeaksAndSubpeaks – limpar picos de valores e subvalores

   (Essa é uma boa opção)
5. Em Threshold tem-se:
   1. Constant – valores constantes
   2. Percentile – valores percentuais
      1. Percentile number of upper – selecione 90 (tudo acima de 90% será removido)
      2. Percentile number of lower – selecione 10 (tudo abaixo de 10% será removido)

         (Comece com valores mais conservadores e aumente gradativamente se necessário)
6. Em Substitute value for peaks (regras usadas para tratar os valores extremos) tem-se:
      1. Threshold – substitui por valores logo acima ou abaixo dos extremos
      2. Mean – substitui por Média
      3. Median – substitui por Mediana

    4. Missing – substitui por valores missing (aqui é importante usar antes do Clean Missing Data)
7. Em List of columns clique em Launch column selector e selecione as colunas desejadas
8. Clique em Run

Manipulação de Dados com Linguagem SQL no Azure ML

Para utilizar a Linguagem SQL no Azure ML você não precisa instalar nenhum pacote antes. Ele já está disponível para uso, como padrão.

1. Inclua um Dataset no experimento
2. Use a caixa de pesquisa e localize Apply SQL Transformation > Arraste para o experimento
    1. O módulo possui 3 portas de entrada, sendo 3 tabelas, e uma porta de saída, sendo 1 dataset
3. Conecte a saída do Dataset ou do script à porta de entrada do módulo de SQL
4. Clique no módulo
5. Em SQL Query Script desenvolva seu SQL

    (o nome da tabela é o nome da porta, ou seja, t1, t2 ou t3)

Usando Join Data para Combinar Datasets no Azure ML

Você pode unir 2 datasets usando o módulo Join no Azure Machine Learning.

1. Inclua 2 Datasets no experimento
2. Use a caixa de pesquisa e localize Join Data > Arraste para o experimento
    1. O módulo possui 2 portas de entrada, sendo 2 datasets, e uma porta de saída, sendo 1 dataset
3. Conecte a saída dos Datasets ou scripts às portas de entrada do módulo Join
4. Clique no módulo
5. Em Join key columns for L (data set da porta esquerda) > Clique em Launch column selector (selecione a chave para join)
6. Em Join key columns for R (data set da porta direita) > Clique em Launch column selector (selecione a chave para join)
7. Marque a opção Match case
8. Em Join type selecione o tipo de join:
    1. Inner Join
    2. Left Outer Join
    3. Full Outer Join
    4. Left Semi-Join

9. Marque Keep right key columns in joined table
10. Salve e Execute o experimento

Instalando Pacotes R no Azure ML

O Azure Machine Learning possui alguns pacotes R pré-instalados, mas requer a instalação manual de pacotes que não estejam disponíveis.

1. Acesse o endereço https://cran.r-project.org
    1. Clique em Packages no menu esquerdo da página
    2. Clique em Table of available packages, sorted by name (pacotes disponíveis por nome)
    3. Localize o seu pacote na lista > Clique no link
    4. Em Downloads > Windows binaries (como o Azure é Microsoft, esse deve ser o arquivo correto) > Clique no link da versão desejada do pacote
    5. Alguns pacotes possuem dependências. Se esse for o caso, na tela do pacote:
        1. Em Depends (um pouco acima de downloads) > Clique nas dependências necessárias
        2. Faça os downloads usando o mesmo procedimento descrito no item 4
2. Após download dos arquivos dos pacotes e suas dependências, crie um novo arquivo para envio ao Azure:
    1. Selecione o(s) arquivo(s) de formato zip
    2. botão direito > Send to (Enviar para) > Compressed (zipped) folder
3. No Azure ML importe o novo arquivo:
    1. New > Dataset > From Local File > Selecione o arquivo
    2. Caso esteja atualizando o pacote, clique em This is the new version of an existing dataset
4. Use a caixa de pesquisa e localize o novo arquivo/dataset > Arraste para o experimento
5. Use a caixa de pesquisa e localize Execute R Script > Arraste para o experimento
6. Conecte a porta de saída do dataset na porta 3 (Script Bundle (zip) do Execute R Script
7. Salve e Execute o experimento

3.14. Análise de Regressão com Linguagem R e Microsoft Azure Machine Learning

Processo de Data Science Para Análise de Big Data (Big Data Analytics)

- Compreender o Problema a ser Resolvido
- Coletar os Dados
- Explorar, Limpar e Preparar os Dados
- Selecionar e Transformar as Variáveis
- Construir, Testar, Avaliar e Otimizar o Modelo
- Contar a História dos Dados

O Que é Feature Selection (seleção de variáveis)?

Em Machine Learning, Feature Selection é o processo de selecionar um subset de variáveis que sejam relevantes para construção do modelo preditivo.

Por Que Usamos Feature Selection?

- Simplificação do modelo, para facilitar sua interpretação.
- Redução do tempo de treinamento do modelo.
- Melhora da generalização do modelo, evitando overfitting.

Que variáveis (features) presentes em nosso conjunto de dados, devem ser usadas na criação do modelo?

As técnicas de feature selection automatizam a seleção das variáveis com maior potencial para variáveis preditoras.

Feature Selection é uma espécie de filtro, que remove do seu dataset as variáveis que não serão úteis para a criação do modelo preditivo.

O principal objetivo ao usar técnicas de Feature Selection é criar um modelo preditivo com a maior precisão possível e que seja **generalizável**.

Existem diversos métodos para Feature Selection:

- Teste do Qui-quadrado
- Coeficientes de Correlação
- Algoritmos de Eliminação Recursiva
- Algoritmos de Regularização (LASSO, Elastic Net, Ridge Regression)

Atenção!

- Feature Selection é um ponto chave do processo de Machine Learning, tão importante quanto a seleção do algoritmo de Machine Learning que será utilisado.
- Aplicar incorretamente o Feature Selection pode tornar o seu modelo tendencioso e causar overfitting.
- Feature Selection é diferente de Redução de Dimensionalidade.

Redução de Dimensionalidade

A Redução de Dimensionalidade, assim como Feature Selection, tem como objetivo a redução do número de variáveis do dataset. Porém, a Redução de Dimensionalidade faz isso criando novas combinações dos atributos, enquanto o Feature Selection inclui ou exclui variáveis no dataset sem modificá-las.

- Principal Component Analysis (PCA)
- Singular Value Decomposition (SVD)

Antes de aplicar Feature Selection, diversas perguntas devem ser respondidas:

- Suas variáveis são mensuráveis?
- Você encontrou interdependência entre as variáveis?
- Você tem conhecimento sobre a área de negócio que gerou os dados?
- Sabe identificar as variáveis mais relevantes dentro do seu conjunto de dados?
- A análise exploratória dos dados encontrou “sujeira” nos seus dados?

As técnicas de Feature Selection basicamente calculam o nível de significância de cada variável e eliminam aquelas com significância mais baixa.

Veremos duas formas de fazer isso:

1. Usando o módulo Filter Based Feature Selection do Azure ML
2. Criando um modelo randomForest para calcular a significância de cada variável, usando R

Filter Based Feature Selection no Azure ML

1. Após inclusão no modelo/experimento do Dataset, Normalize, Execute R Script, Select Columns, etc.
2. Na busca, procure por Filter Based Feature Selection e arraste para o experimento
3. Selecione os atributos necessários para criação do modelo em Select Columns in Dataset
4. Conecte a saída do Select Columns in Dataset à entrada do Filter Based Feature Selection
5. Clique em Filter Based Feature Selection
    1. Em Feature scoring method:
        - Pearson Correlation (preferencial)

        - Mutual Information
        - Kendall Correlation (preferencial)
        - Spearman Correlation (preferencial)
        - Chi Squared (preferencial para categóricas)
        - Fisher Score
        - Count Based
    2. Target column – variável alvo
    3. Number of desired features – número de variáveis preditoras relevantes
6. Clique com o botão direito na primeira porta > Selecion Visualize

   (visualize as variáveis mais importantes para o modelo)

FIM DO PRÉ-PROCESSAMENTO!!!

Construindo e Treinando o Modelo Preditivo no Azure ML

1. Na busca, procure por Split Data (para dividir entre dados de treino e de teste) e arraste para o experimento
2. Clique em Split Data:
    1. Splitting mode – critérios para divisão dos dados:
        - Split Rows
        - Recommender Split
        - Regular Expression
        - Relative Expression
    2. Fraction of rows in the first…
        - Ex: 0.7 (70% treino e 30% teste)
    3. Random seed – informe se quiser obter resultados semelhantes a anteriores (para estudos ou comparações)
    4. Stratified split
        - True/False (padrão)
3. Na busca, procure por Train Model (para treinamento do modelo) e arraste para o experimento
4. Conecte a primeira porta de saída do Split Data à segunda porta de entrada do Train Model (dados de treino)
5. Clique em Train Model > Selecione a variável target em Launch column selector
6. Na busca, procure pelo modelo de Machine Learning a ser utilizado. Ex: Linear Regression > Arraste para dentro do experimento

7. Conecte a porta de saída do Linear Regression à primeira porta de entrada do Train Model
8. Na busca, procure pelo módulo Score Model (para testar o modelo com os dados de teste) e arraste para o experimento
9. Conecte a porta de saída de Train Model à primeira porta de entrada do Score Model (Trained Model)
10. Conecte a segunda porta de saída do Split Data à segunda porta de entrada do Score Model (Dataset de teste)
11. Na busca, procure por Evaluate Model (para avaliar o modelo) e arraste para o experimento
12. Conecte a saída do Score Model à primeira porta de entrada do Evaluate Model
13. Salve e Execute o experimento
14. Clique com botão direito na porta de saída do Evaluate Model > Visualize
    - Error Histogram – histograma com os erros de previsão
    - Metrics (onde a métrica mais importante é a Coefficient of Determination, que é o R squared – ele vai de 0 a 1; quanto maior, melhor; abaixo de 0.80 não dá pra usar o modelo)

Criando um Módulo R no Azure ML

O objetivo é criar seu próprio script de machine learning.

1. Em R Language Modules ou procure na busca por Create R Model > Arraste para o experimento
2. Clique em Create R Model
    - Trainer R script – para treinar o modelo
        - Copie e cole o script 06-CreateModel.R, alterando o dataset e a flag do Azure
    - Scorer R script – para fazer as previsões do modelo
        - Copie e cole o script 07-ScoreModel.R, alterando o dataset e a flag do Azure
    - Obs.: o módulo Create R Model não possui porta de entrada!
3. Refaça os passos 7 (agora com o Create R Model) a 14 da lista anterior!

Computação de Resíduos do Modelo no Azure ML

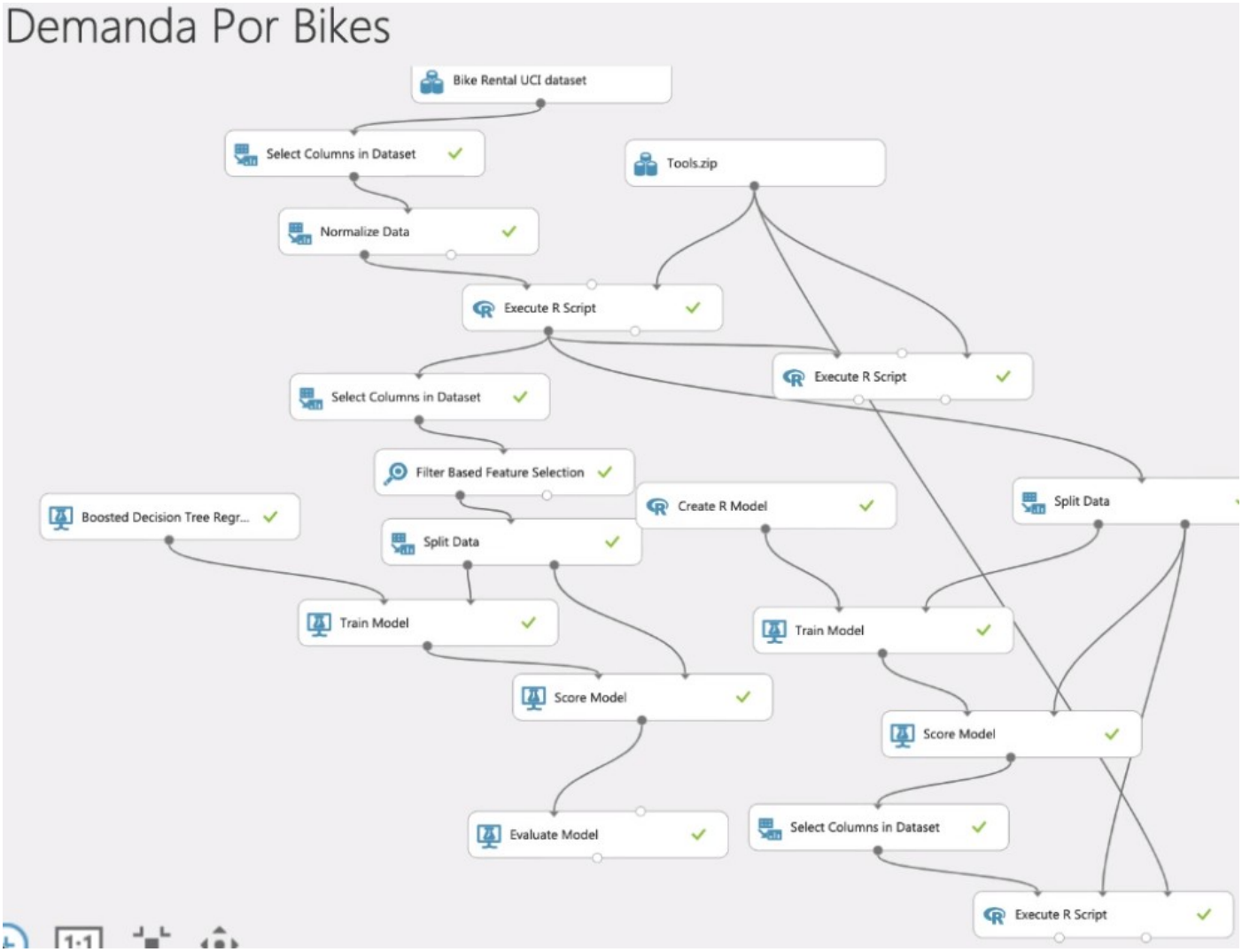
O objetivo é calcular a diferença entre valores históricos e valores previstos, utilizando a linguagem R.

É importante que a distribuição dos resíduos siga um padrão de distribuição normal. Isso significa que, mesmo possuindo erros, o modelo está correto. Mas não significa que não há necessidade de otimização da performance do modelo, para redução de taxa de erros.

Obs.: esse procedimento necessita da sequência de passos anteriores do estudo do capítulo, bem como os seus módulos contidos no experimento (Dataset, SplitData, etc.)

1. Na busca, procure por Select Columns in Dataset e arraste para o experimento
2. Na busca, procure por Execute R Script e arraste para o experimento
3. Conecte a saída do Score Model à entrada do Select Columns in Dataset
4. Clique em Select Columns in Dataset
    1. Selecione as variáveis de trabalho em Launch column selector
        - Valores reais e previsões
5. Conecte a porta de saída do Select Columns in Dataset à primeira porta do Execute R Script
6. Conecte a segunda porta de saída do Split Data (dados de teste) à segunda porta do Execute R Script
7. Conecte o script de Tools.zip na terceira porta de entrada do Execute R Script
8. Selectione o Execute R Script > Copie o script de 08-AvaliaModel.R e cole em R Script, alterando flag Azure
9. Salve e Execute o experimento

Visão do Modelo Finalizado do Capítulo:



Storytelling

Uma forma de mostrar o resultado do trabalho, através de histórias.

Como resumir todo um trabalho de análise e explicar isso a quem não conhece Data Science, Estatística, Programação ou Machine Learning?

- Não utilize linguagem técnica em apresentações executivas.
- Use apenas um ou dois gráficos para explicar seus resultados (use tabelas se necessário).
- Tente construir uma linha do tempo, explicando como os dados geraram o resultado preditivo esperado.
- Não use termos como “variáveis” e sim “atributos” ou “características”.
- Documente todo seu trabalho, pois isso servirá de apoio para que você possa contar a história dos dados.

- Os tomadores de decisão não estão interessados em saber se você usou um algoritmo de regressão ou de classificação e nem como você otimizou seu modelo. Eles querem saber o que precisam fazer para aumentar as vendas e o faturamento!
- Responda o que é importante para eles e não o que é importante para você!
- Com o objetivo de negócio que foi traçado no início, mostre o que deve ser feito, com base na análise realizada. O processo de análise não interessa aos tomadores de decisão, mas sim o resultado final!
- A fase de Análise Exploratória dos Dados pode apresentar insights preciosos. Use isso ao seu favor e guarde todos os resultados intermediários do seu processo de análise!

Storytelling é uma arte e requer prática para ser exercida com maestria!

3.15. Classificação com Linguagem R e Microsoft Azure Machine Learning

O Que é Classificação?

Exemplos de Classificação:

- Seleção de rotas num app de mapas (rota é boa ou não)
- Solicitação de empréstimo num banco (conceder ou não o empréstimo)

Classificação é comumente mais usada do que Regressão, em Machine Learning.

Principais Caraterísticas de Classificação

- Aprendizagem Supervisionada

  Treinamento do modelo com dados de entrada e dados de saída devidamente rotulados.
- Classe de modelos para categorizar valores
- Métodos Two-class e Multi-class
- Erros são medidos pelas taxas de classificações incorretas

  Obs.: em Regressão, isso é feito com o cálculo de resíduos.
- Alguns erros podem ser mais críticos que outros e trade-offs terão que ser feitos

Nos modelos de regressão, usamos o cálculo dos resíduos para avaliar a perfomance do modelo. Já em modelos de classificação, usamos as taxas de erro ou acerto no processo de classificação, para avaliar a perfomance.

Performance dos Modelos de Classificação

Confusion Matrix (Matriz de Confusão) é uma tabela de suporte para calcular medidas de performance. Ela compara as previsões do modelo com os valores históricos.

	Previstos	
Atuais	Sim	Não
Sim	True Positive (TP)	False Negative (FN)
Não	False Positive (FP)	True Negative (TN)

A Confusion Matrix não é o objetivo final. Trata-se de uma ferramenta e, com ela calculada, é possível extrair as medidas de performance para avaliação do modelo de Machine Learning.

Quando trabalhamos com classificadores binários, ou seja, apenas 2 valores são possíveis, a principal forma de avaliar a perfomance do nosso modelo é através de uma Confusion Matrix.

Medidas de Performance

- Accuracy (acurácia): Total de resultados corretos / Total de casos analisados
- Recall: Total de resultados positivos / Total de resultados corretos
- Precision (precisão): Proporção de ”true” / Total de resultados corretos
- F-Score: F = 2 * TP / (2 * TP + FP + FN) (É o Balanceamento entre Precision e Recall)
- AUC (gráfico): AUC = Area Under the Curve. Plot de TP no eixo y e FP no eixo x

	Previstos	
Atuais	Sim	Não
Sim	True Positive (TP)	False Negative (FN)
Não	False Positive (FP)	True Negative (TN)

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{F-Score} = 2 * TP / (2 * TP + FP + FN)$$

Aplicando Engenharia de Atributos em Variáveis Numéricas no Azure ML

1. Inclua Dataset
2. Edite os dados, convertendo (quando necessário) atributos não-categóricos em categóricos, mude o nome das colunas (para melhor entendimento) – Procure por Edit Metadata > Arraste para o experimento
3. Crie grupos em atributos onde existam muitos números contínuos únicos (*quantization*) – Procure por Group Data into Bins > Arraste para o experimento

Balanceamento do Dataset – Synthetic Minority Oversampling Technique (SMOTE)

Para resolução de problemas de classificação com Machine Learning, é importante que o número de classes target esteja balanceado dentro do conjunto de dados. Caso contrário, o modelo pode aprender mais sobre um determinado tipo de classe e, para a generalização do modelo, isso pode se tornar um problema.

A função do SMOTE é balancear os dados a fim de evitar modelos preditivos tendenciosos.

1. Inclua o módulo Split Data > Divida os dados entre treino e teste
2. Inclua o módulo Select Columns in Dataset > Retire do dataset as variáveis que deram origem às variáveis com “_f” (passaram pelo processo de quantization)

3. Inclua o módulo Edit Metadata > Selecione a variável CreditStatus, pois será ela a variável a ser balanceada
4. Inclua o módulo SMOTE (técnica para balanceamento de dados)
    - Launch column selector – seleciona CreditSatus
    - SMOTE percentage – percentual de balanceamento – 100(%)
    - Number of nearest neighbors – número de classes da variável selecionada
    - Random seed – para simular de forma controlada

Feature Selection (Seleção de Variáveis)

O principal objetivo do Feature Selection é identificar que variáveis são mais relevantes no dataset.

Método 1: Random Forest – no R Studio

É possível utilizar o algoritmo Random Forest para seleção das variáveis mais significantes para a construção do modelo. Isso é realizado com cálculo do nível de significância de cada variável (parâmetro “importance = T”).

Método 2: Filter Based Feature Selection – no Azure ML

1. Inclua o módulo Filter Based Feature Selection >
    1. Feature scoring method > Selecione Mutual Information
    2. Launch column selectio > Selecione a variável Target
    3. Number of desired features > Selecione o número de variáveis preditoras

Método 3: Permutation Feature Importance – no Azure ML

Esse método deve ser utilizado após a criação do modelo.

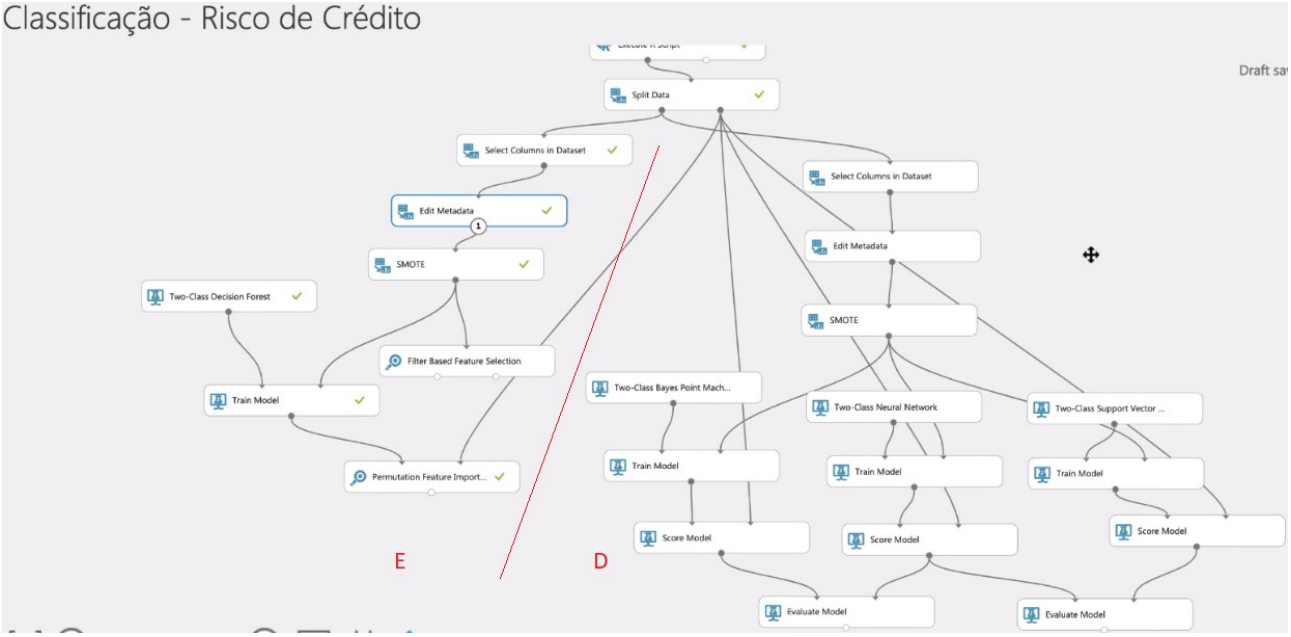
1. Treine o modelo:
    1. Inclua o módulo Train Model
    2. Inclua o módulo do algoritmo desejado – Ex: Two-Class Decision Forest
2. Inclua o módulo Permutation Feature Importance > Selecione o método
    - Random seed – controle sobre aleatoriedade
    - Metric for measuring performance – métrica para mensurar performance
        1. Classification – Accuracy
        2. Classification – Precision
        3. Classification – Recall
        4. Classification – Average Log Loss
        5. Regression – Mean Absolute Error
        6. Regression – Root Mean Squared Error

7. Regression – Relative Absolute Error
8. Regression – Relative Squared Error
9. Regression – Coefficient of Determination

3. Conecte os dados de teste de Split Data ao módulo Permutation Feature Importance

Construção do Experimento – Visão Geral

Visão Geral do experimento:



Do lado esquerdo, tem-se todo o processo de seleção de variáveis.

Do lado direito, tem-se o treinamento e comparação de 3 modelos.

Construção do Experimento – Visão Geral

1. Split Data
2. Select Column in Dataset
    1. Variáveis selecionadas a partir da indicação do R Script com Random Forest (gráfico Mean Decrease Accuracy)
3. Edit Metadata
    1. Em Column, selecione a variável target
    2. Em Fields, selecione Label
4. SMOTE
    1. Em Label column, mantenha All labels
    2. Em Smote percentage, informe 100(%)
    3. Em Number of nearest neighbor, informe 2 (são 2 classes apenas)
5. Modelo 1:
    1. Train Model
        1. Em Label column, selecione a variável target

2. Conecte o algoritmo Two-Class Bayes Point Match
    1. Em Number of training iterations, informe 100
    2. Mantenha selecionado Include bias e Allow unknown values
6. Modelo 2:
    1. Train Model
        1. Em Label column, selecione a variável target
    2. Conecte o algoritmo Two-Class Neural Network
        1. Em Create trainer mode, selecione Single Parameter
        2. Hidden layer specification, selecione Fully-connected case
        3. Number of hidden rode, informe 100
        4. Learning rate, informe 0.1
        5. Number of learning iterations, informe 100
        6. The initial learning weight, informe 0.1
        7. The momentum, informe 0
        8. The type of normalizer, selecione Min-Max normalizer (ou seja, esse módulo já faz normalização dos dados automaticamente)
        9. Mantenha selecionado Shuffle examples e Allow unknown categories
7. Modelo 3:
    1. Train Model
        1. Em Label column, selecione a variável target
    2. Conecte o algoritmo Two-Class Support Vector Machine (SVM)
        1. Em Create trainer mode, selecione Single Parameter
        2. Em Number of iterations, informe 1
        3. Lambda, informe 0.001
        4. Selecione Normalize features e Allow unknown categories
8. Score Model (um para cada modelo)
    1. Mantenha selecionado Append score column
    2. Esse módulo recebe a saída do Train Model e os dados de teste do Split Data
9. Evaluate Model
    1. Não possui parâmetros
    2. Cada módulo compara a execução de dois modelos
10. Save e Run

Como Interpretar os Modelos de Classificação:

Há duas subcategorias de problemas de classificação:

- Problemas com apenas duas classes (Classificação de duas classes ou binária)
- Problemas com mais de duas classes (Classificação multiclasse)

O Azure ML tem diferentes módulos para lidar com cada um desses tipos de classificação. Mas as maneiras de interpretar os resultados de previsão são muito semelhantes. Vamos estudar sobre os problemas de classificação de duas classes.

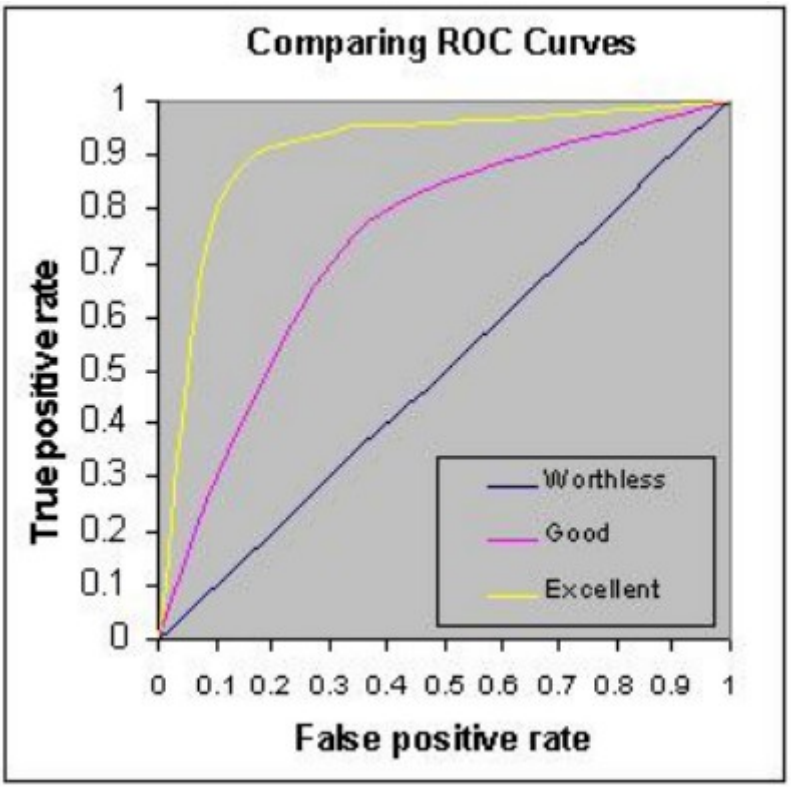
Avaliar um modelo de classificação binária

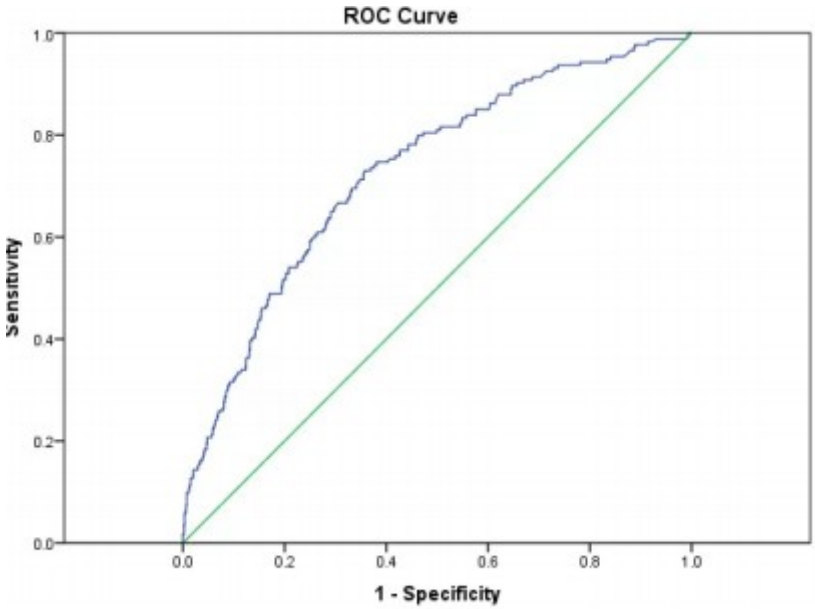
Em um cenário de classificação binária, a variável a ser prevista tem somente dois resultados possíveis, por exemplo: {0, 1}, {falso, verdadeiro}, {negativo, positivo}, {bom pagador, mau pagador}.

Depois de executar o experimento no Azure ML, você pode clicar na porta de saída do módulo Avaliar Modelo (Evaluate Model) e selecionar Visualizar para ver os resultados da avaliação. As métricas de avaliação disponíveis para modelos de classificação binária são:

- Exatidão (Accuracy)
    - Accuracy à mede a exatidão do modelo de classificação, como a proporção de resultados verdadeiros em relação ao total de casos analisados. Quanto maior, melhor!
- Precisão (Precision)
    - Precision à é a proporção de resultados verdadeiros sobre os resultados positivos. Quanto maior, melhor!
- Recuperação (Recall)
    - Recall à é a fração de resultados corretos retornados pelo modelo. Quanto maior, melhor!
- Pontuação F1 (F-Score)
    - F-Score à é a média ponderada entre a precisão e o recall. O valor ideal para o FScore é igual a 1.
- AUC (Area Under the Curve)
    - AUC à mede a área sob a curva, sendo os verdadeiros positivos no eixo y e os falsos positivos no eixo x. Esta métrica é útil, pois gera uma única medida que permite comparar modelos diferentes.

Além disso, o módulo gera uma matriz de confusão (Confusion Matrix) mostrando o número de verdadeiros positivos, de falsos negativos, de falsos positivos e de verdadeiros negativos, bem como as curvas ROC (Receiver Operating Characteristic).





Quanto mais à esquerda a curva estiver, maior a precisão do seu modelo!

A precisão é simplesmente a proporção das instâncias classificadas corretamente. Geralmente, ela é a primeira métrica a observar ao avaliar um classificador. No entanto, quando os dados de teste estão desbalanceados (em que a maioria das instâncias pertence a uma das classes) ou você está mais interessado no desempenho em qualquer uma das classes, a precisão pode não capturar a eficácia de um classificador.

Por esse motivo, é útil computar métricas adicionais que capturam aspectos mais específicos da avaliação. Antes de entrar em detalhes sobre essas métricas, é importante compreender a Confusion

Matrix de uma avaliação de classificação binária. Os rótulos de classe no conjunto de treinamento podem assumir apenas 2 valores possíveis, o que normalmente chamamos como positivo ou negativo. As instâncias positivas e negativas que um classificador prevê corretamente são chamadas verdadeiros positivos (TP) e verdadeiros negativos (TN), respectivamente. Da mesma forma, as instâncias classificadas incorretamente são chamadas de falsos positivos (FP) e falsos negativos (FN). A Confusion Matrix é simplesmente uma tabela que mostra o número de instâncias que se enquadram em cada uma dessas 4 categorias.

O Azure ML decide automaticamente qual das duas classes no conjunto de dados é a classe positiva.

Dados Reais:

1 = Crédito Bom

2 = Crédito Ruim

Azure ML:

Positive Label = 2 = Crédito Ruim

Negative Label = 1 = Crédito Bom

- TP - Crédito Ruim classificado como crédito Ruim
- FN - Crédito Ruim classificado como crédito Bom
- FP - Crédito Bom classificado como Crédito Ruim
- TN - Crédito Bom classificado como Crédito Bom

Você observará que há uma relação óbvia entre a precisão (Precision) e a recuperação (Recall). Por exemplo, dado um conjunto de dados relativamente equilibrado, uma classificação que prevê principalmente instâncias positivas, teria um Recall alto, mas um Precision bastante baixo com muitas instâncias negativas, resultando em um grande número de falsos positivos. Para ver um gráfico de como essas duas métricas variam, você pode clicar na curva ‘PRECISION/RECALL’ na página de saída do resultado de avaliação.

Outra métrica usada com frequência é a Pontuação F1, que calcula a relação entre Precision e Recall. Ela é a média ponderada dessas 2 métricas e é calculada como tal: F1 = 2 (precisão x recuperação) / (precisão + recuperação). O score F1 é uma boa maneira de resumir a avaliação de um único número, mas é sempre uma boa prática examinar Precision e Recall para entender melhor como um classificador se comporta.

Além disso, é possível inspecionar a taxa de verdadeiros positivos versus a taxa de falsos positivos na curva ROC (Característica de Operação do Receptor) e o valor correspondente AUC (Área sob a Curva). Quanto mais próxima essa curva estiver do canto superior esquerdo, melhor será o desempenho do classificador (que é maximizar a taxa de positivos verdadeiros enquanto minimiza os falsos positivos).

Existem ainda 2 importantes métricas que podem ser consideradas:

- Average log loss é um único score usado para expressar as penalidades para resultados errados. Este score é calculado como a diferença entre 2 distribuições de probabilidade: a distribuição real dos dados e a distribuição gerada pelo modelo.
- Training log loss é um único score que representa a vantagem do classificador sobre predição randômica.

Recomendações Sobre Otimização

Dicas Gerais:

- Diferentes conjuntos de variáveis
- Utilizar outros algoritmos
- Aplicar quantization a variáveis numéricas (transformá-las em variáveis categóricas)
- Otimizar os parâmetros dos algoritmos

Decisões de Negócio:

- Quais features (variáveis) são mais relevantes?
- Trade-off entre falsos positivos e falsos negativos
- O problema pode ser resolvido com esses dados?

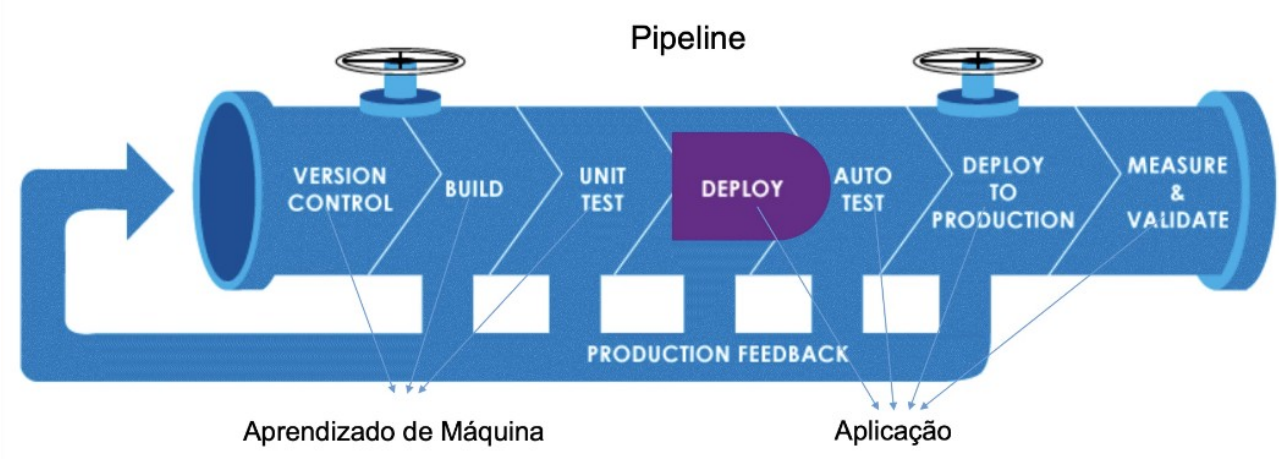
Cada etapa do processo pode ser otimizada:

- Limpeza e Preparação de Dados
- Exploração dos Dados
- Feature Selection
- Testar e Avaliar o Modelo
- Otimizar o Modelo

3.16. Publicação Online do Modelo Preditivo

O Que é Deploy de Modelos de Machine Learning?

Deploy do Modelo em Produção é o processo de fazer o modelo resolver o problema para o qual ele foi criado.



Pipeline.

Aprendizado de Máquina é um meio e não um fim. Para fazer o Deploy do modelo em produção, você precisa criar uma aplicação que faça a leitura do resultado do modelo e então entregue ao usuário final esse resultado.

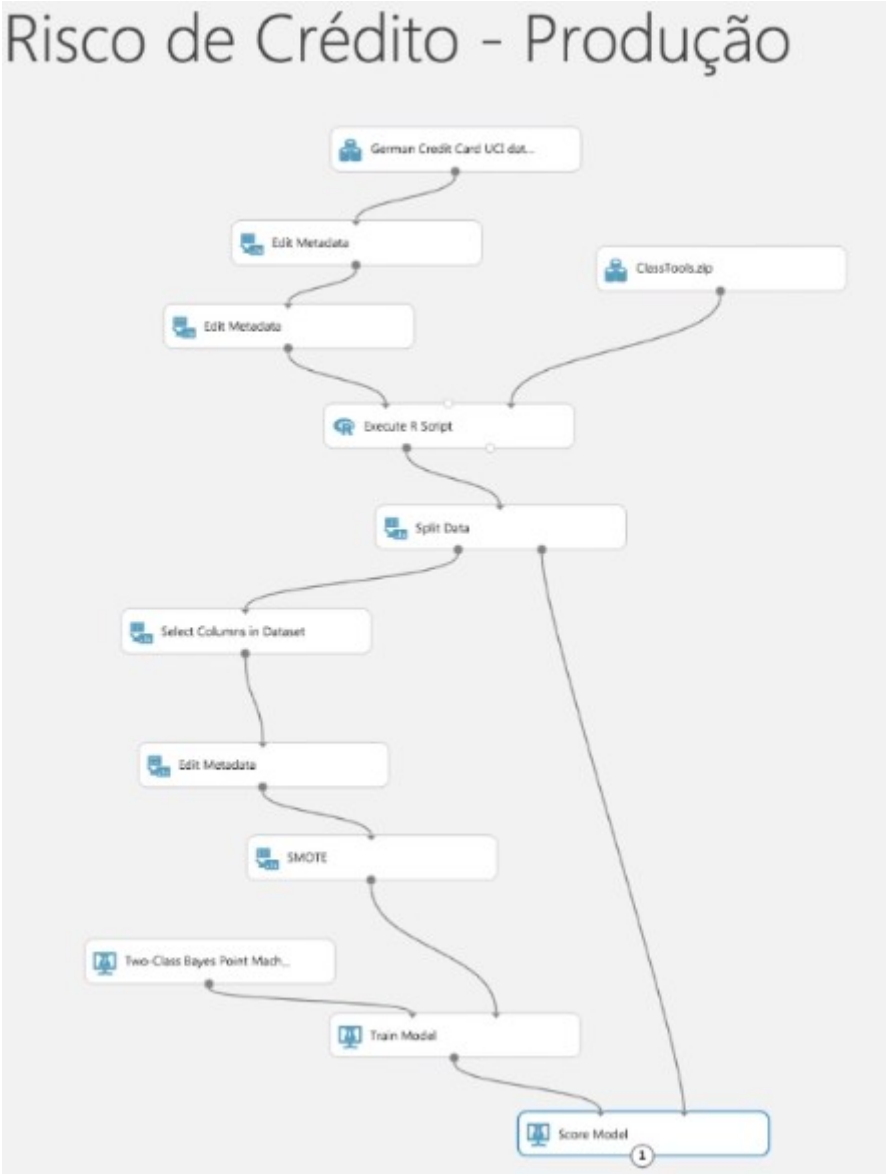
E como podemos fazer o Deploy de um Modelo de Machine Learning?

1. Deploy do Modelo em nuvem com Google Cloud Platform, AWS ou Azure.
2. Aplicação Web com o Framework Web (Flask para linguagem Python e Shiny para linguagem R são os mais comuns).
3. Deploy do Modelo com TensorFlow Serving.
4. Desenvolvimento de APIs.

Em geral, o Deploy de um Modelo de Machine Learning é trabalho de um Engenheiro de Dados e não do Cientista de Dados.

Publicando seu Modelo no Azure Machine Learning como um Serviço Web

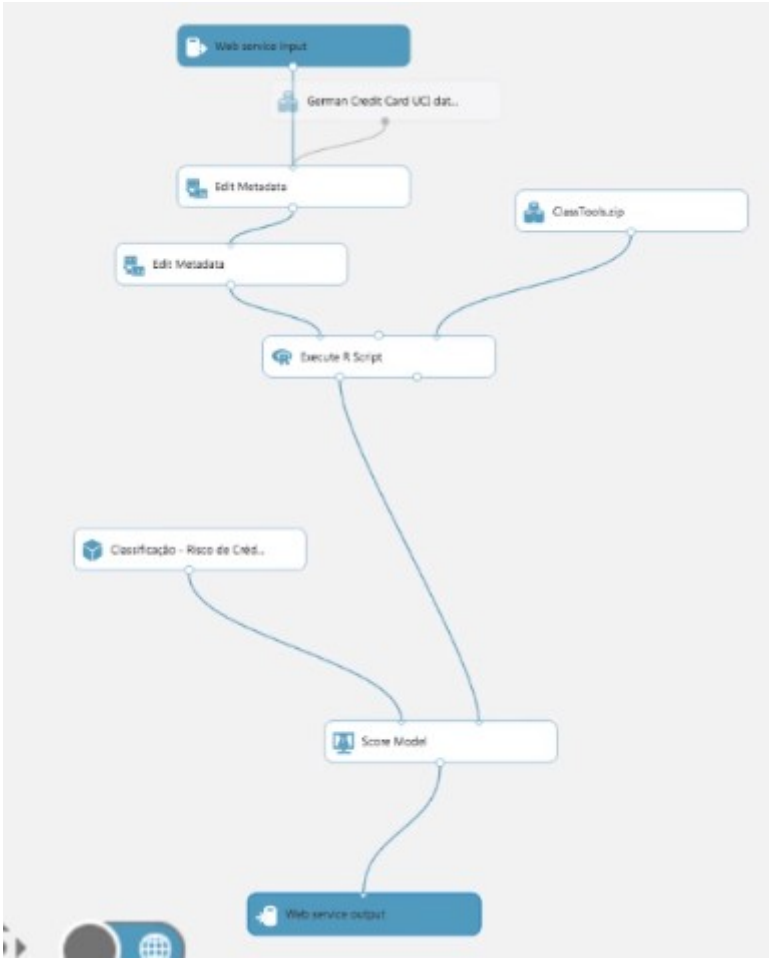
Visão do projeto para deploy, retirando módulos intermediários usados para análise do modelo:



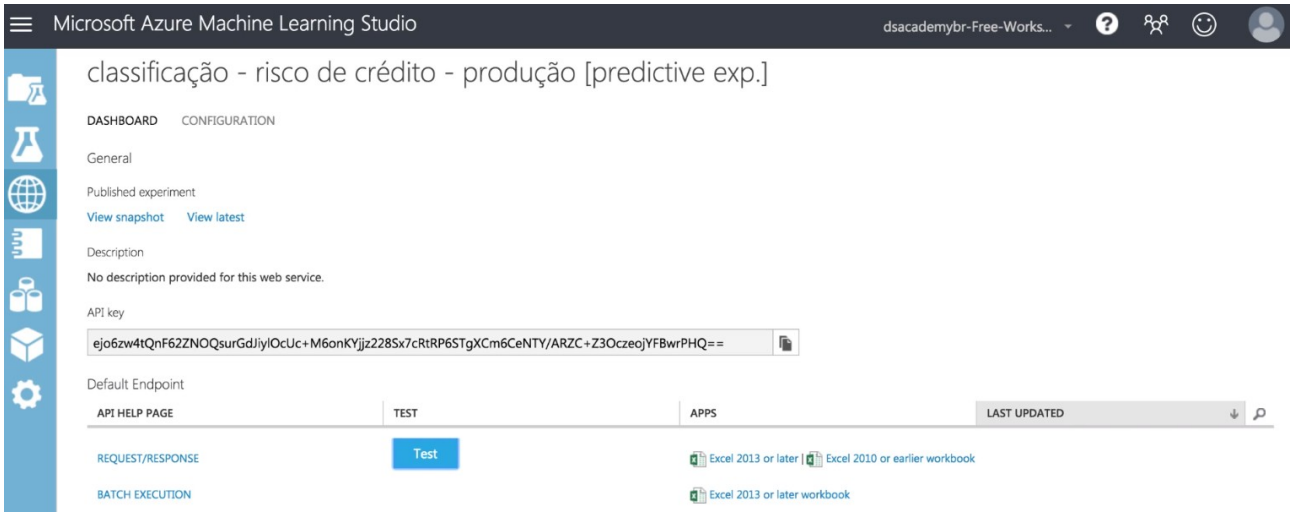
1. Clique em Setup Web Service > Predictive Webservice

   Obs.: Caso queira atualizar o modelo, selecione Retraining Web Service
2. Após execução acima, clique em Run (o Azure solicita para verificação final do modelo)
3. Clique em Deploy Web Service

Após Execução do comando, o modelo ainda sofre algumas alterações, como inclusão de dois novos módulos Web Service Input e Web Service Output, bem como exclusão do Split Data, Train Model, Smote, etc.



Tela de Dashboard do Deploy:



Linhas de Código do Curso

Capítulo 02:

```
# Primeiros Passos na Linguagem R

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:\Desenvolvedor\CienciaDeDados\Estudos\DSA-Cursos\FCD\03_BDataAnalyticsR_MAzureML/Cap02")
getwd()


# Nome dos Contributors - pessoas que contribuem para a linguagem R
contributors()


# Licença - Termos de uso
license()

# gives you a list of all the environment variables
Sys.getenv()

# alterando a linguagem das mensagens para inglês:
Sys.setenv(LANG = "en")

# Informações sobre a sessão
sessionInfo()


# Imprimir na tela
print('Estou iniciando minha caminhada na carreira de Cientista de Dados')


# Criar gráficos
plot(1:25)


# Instalar pacotes (ou conjunto de funções)
install.packages('randomForest')
install.packages('ggplot2')
install.packages("dplyr")
install.packages("devtools")
install.packages("caret")


# Como carregar o pacote instalado na memória:
library(ggplot2)


# Como descarregar o pacote da memória:
detach(package:ggplot2)


# Ajuda de funções - se souber o nome da função
help(mean)
?mean # outra maneira de realizar o comando help
```

```
# Para buscar mais opções sobre uma função, use o pacote SOS
install.packages("sos") # instala
library(sos) # carrega
findFn("fread") # roda a função do pacote sos


# Ajuda - Se não souber o nome da função
help.search('randomForest')
help.search('matplot')
??matplot # outra maneira de realizar o comando help.search
RSiteSearch('matplot') # website de busca do R
example('matplot') # cria exemplo de utilização de pacote ou função


# Sair - finalizar RStudio
q()


# Operadores Básicos, Relacionais e Lógicos em R

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
getwd()
setwd("D:\Desenvolvedor\CienciaDeDados\Estudos\DSA-Cursos\FCD\03_BDataAnalyticsR_MAzureML/Cap02")
getwd()

# Operadores Básicos

# Soma
7 + 7

# Subtração
7 - 4

# Multiplicação
5 * 5

# Divisão
6 / 6

# Potência
3^2
3**2

# Módulo (resto da divisão)
16 %% 3


# Operadores Relacionais

# Atribuindo variáveis
x = 7
y = 5

# Operadores
x > 8
x < 8
x <= 8
```

```
x >= 8
x == 8
x != 8


# Operadores lógicos

# And
(x==8) & (x==6)
(x==7) & (x>=5)
(x==8) & (x==7)

# Or
(x==8) | (x>5)
(x==8) | (x>=5)

# Not
x > 8
print(!x > 8)

# Variáveis em R

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:\Desenvolvedor\CienciaDeDados\Estudos\DSA-Cursos\FCD\03_BDataAnalyticsR_MAzureML/Cap02")
getwd()

# Criando Variáveis
var1 = 100
var1
mode(var1)
help("mode")
sqrt(var1)


# Podemos atribuir o valor de uma variável a outra variável
var2 = var1
var2
mode(var2)
typeof(var2)
help("typeof")


# Uma variável pode ser uma lista de elementos - matriz
var3 = c("primeiro", "segundo", "terceiro")
var3
mode(var3)


# Uma variável pode ser uma função
var4 = function(x) {x+3}
var4
mode(var4)


# Podemos também mudar o modo do dado.
var5 = as.character(var1)
var5
mode(var5)
```

```
# Atribuindo valores a objetos
x <- c(1,2,3)
x
x1 = c(1,2,3)
x1
c(1,2,3) -> y
y
assign("x", c(6.3,4,-2))
x


# Verificando o valor em uma posição específica
x[1]


# Verificar objetos da memoria
ls()
objects()


# Remover objetos
rm(x)
x
# Tipos Básicos de Dados em R

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
getwd()

setwd("???/Cap02")
getwd()

# Numeric - Todos os números criados em R são do modo numeric
# São armazenados como números decimais (double)
num1 <- 7
num1
class(num1)
mode(num1)
typeof(num1)

## obs: em Environment, mude de List p/ Grid !

num2 = 16.82
num2
mode(num2)
typeof(num2)


# Integer
# Convertemos tipos numeric para integer
is.integer(num2) # pergunta se eh inteiro
y = as.integer(num2) # converte p/ inteiro
y
class(y)
mode(y)
typeof(y)
```

```
# Testando conversão:
as.integer('3.17')
as.integer("Joe")
as.integer('Joe')
as.integer(TRUE)
as.integer(FALSE)
as.integer('TRUE')


# Character (string)
char1 = 'A'
char1
mode(char1)
typeof(char1)

char2 = "cientista"
char2
mode(char2)
typeof(char2)

char3 = c("Data", "Scicence", "Academy")
char3
mode(char3)
typeof(char3)


# Complex
compl = 2.5 + 4i
compl
mode(compl)
typeof(compl)

sqrt(-1)
sqrt(-1+0i)
sqrt(as.complex(-1))


# Logic
x = 1; y = 2 # separando variaveis na mesma linha
z = x > y
z
class(z)

u = TRUE; v = FALSE
class(u)
u & v
u | v
!u


# Operações com 0
5/0 # Infinito (erro divisão)
0/5

# Erro
'Joe'/5

# Tipos Avançados de Dados em R

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
```

```
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
getwd()

setwd("???/Cap02")
getwd()

# Vetor: possui 1 dimensão e 1 tipo de dado

vetor1 <- c(1:20) # função c
vetor1
length(vetor1)
mode(vetor1)
class(vetor1)
typeof(vetor1)


# Matriz: possui 2 dimensões e 1 tipo de dado

matriz1 <- matrix(1:20, nrow = 2) #função matrix
matriz1
length(matriz1)
mode(matriz1)
class(matriz1)
typeof(matriz1)


# Array: possui 2 ou mais dimensões e 1 tipo de dado

array1 <- array(1:5, dim = c(3,3,3))
array1
length(array1)
mode(array1)
class(array1)
typeof(array1)


# Data Frames: dados de diferentes tipos
# Maneira mais fácil de explicar data frames: é uma matriz com diferentes tipos de dados

View(iris)
length(iris)
mode(iris)
class(iris)
typeof(iris)


# Listas: coleção de diferentes objetos
# Diferentes tipos de dados são possíveis e comuns

lista1 <- list(a = matriz1, b = vetor1)
lista1
length(lista1)
mode(lista1)
class(lista1)
typeof(lista1)


# Funções também são vistas como objetos em R

func1 <- function(x) {
  var1 <- x * x
  return(var1)
```

```
}

func1(5)
class(func1)


# Removendo objetos
objects()
rm(array1, func1)
objects()

# Vetores, Operações com Vetores e Vetores Nomeados

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
getwd()

setwd("???/Cap02")
getwd()

# Vetor de strings
vetor_caracter = c("Data", "Science", "Academy")
vetor_caracter


# Vetor de floats (decimais)
vetor_numerico = c(1.90, 45.3, 300.5)
vetor_numerico


# Vetor de valores complexos
vetor_complexo = c(5.2+3i, 3.8+4i)
vetor_complexo


# Vetor de valores lógicos
vetor_logico = c(TRUE, FALSE, TRUE, FALSE, FALSE)
vetor_logico


# Vetor de números inteiros (cria como numeric por padrao)
vetor_integer  = c(2, 4, 6)
vetor_integer


# Utilizando funcao seq()
vetor1 = seq(1:10)
vetor1
is.vector(vetor1) # verifica se é vetor


# Utilizando funcao rep()
vetor2 = rep(1:5)
vetor2
is.vector(vetor2)


# Indexação de vetores
a <- c(1,2,3,4,5)
```

```
a
a[1] # diferente de outras linguagens que o indice inicial é o 0 !!!
a[6]

b <- c("Data", "Science", "Academy")
b
b[1]
b[2]
b[3]
b[4]


# Combinando vetores
v1 = c(2, 3, 5)
v2 = c("aa", "bb", "cc", "dd", "ee")
c(v1, v2) # combina transformando tudo em string


# Operações com Vetores
x = c(1, 3, 5, 7)
y = c(2, 4, 6, 8)

x * 5 # multiplica cada item por 5
x + y # operacoes entre itens de indices iguais
x - y
x * y
x / y


# Somando vetores com números diferentes de elementos
alfa = c(10, 20, 30)
beta = c(1, 2, 3, 4, 5, 6, 7, 8, 9)
alfa + beta # ao fim do menor vetor, a operacao continua voltando ao inicio, ateh o fim do vetor maior


# Vetor Nomeado ("cabeçalho")
v = c("Nelson", "Mandela")
v
names(v) = c("Nome", "Sobrenome") # funcao names nomeia as colunas do vetor
v
v["Nome"] # agora é possível recuperar as informações pelo título dado à coluna

# Matrizes, Operações com Matrizes e Matrizes Nomeados

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
getwd()

setwd("???/Cap02")
getwd()

# Criando Matrizes

# Número de Linhas
matrix (c(1,2,3,4,5,6), nr = 2) # cria a matriz, passando o vetor e a quantidade de linhas
matrix (c(1,2,3,4,5,6), nr = 3)
matrix (c(1,2,3,4,5,6), nr = 6)
```

```
# Número de Colunas
matrix (c( 1,2,3,4,5,6), nc = 2)


# Help da funcao matrix
?matrix


# Matrizes precisam ter um número de elementos que seja múltiplo do número de linhas
matrix (c(1,2,3,4,5), nc = 2)


# Criando matrizes a partir de vetores e preenchendo a partir das linhas
meus_dados = c(1:10)
matrix(data = meus_dados, nrow = 5, ncol = 2, byrow = T) # byrow=True muda o preenchimento p/ linha
matrix(data = meus_dados, nrow = 5, ncol = 2)


# Fatiando a Matriz
mat <- matrix(c(2,3,4,5), nr = 2)
mat
mat[1,2] # linha, coluna
mat[2,2]
mat[1,3]
mat[,2] # todas as linhas, coluna


# Criando uma matriz diagonal (todos os elementos são zero, exceto em sua diagonal)
matriz = 1:3 # outra forma de criar matrizes (vetores)
matriz
diag(matriz) # função diag converte um vetor em matriz diagonal


# Extraindo vetor de uma matriz diagonal
vetor = diag(matriz) # aqui, vetor é uma matriz diagonal
vetor
diag(vetor) # função diag converte uma matriz diagonal em vetor


# Transposta da matriz (linhas viram colunas e colunas viram linhas)
W <- matrix (c(2,4,8,12 ), nr = 2, ncol = 2)
W
t(W) # função t converte uma matriz em transposta
U <- t(W)
U


# Obtendo uma matriz inversa
solve(W) # função solve cria matriz inversa de outra matriz


# Multiplicação de Matrizes
mat1 <- matrix(c(2,3,4,5), nr = 2)
mat1
mat2 <- matrix(c(6,7,8,9), nr = 2)
mat2
mat1 * mat2
mat1 / mat2
mat1 + mat2
mat1 - mat2


# Multiplicando Matriz com Vetor
```

```
x = c(1:4)
x
y <- matrix(c(2,3,4,5), nr = 2)
x * y


# Nomeando a Matriz
mat3 <- matrix(c('Terra', 'Marte', 'Saturno', 'Netuno'), nr = 2)
mat3
dimnames(mat3) = (list( c("Linha1", "Linha2"), c("Coluna1", "Coluna2"))) #função dimnames nomeia as dimensões de uma matriz
mat3 # observe que usou-se um vetor para linhas e outro para colunas, pois são 2 dimensões diferentes


# Identificando linhas e colunas no momento de criação da Matriz
matrix (c(1,2,3,4), nr = 2, nc = 2, dimnames = list(c("Linha 1", "Linha 2" ), c( "Coluna 1", " Coluna 2") ))


# Combinando Matrizes (unindo matrizes)
mat4 <- matrix(c(2,3,4,5), nr = 2)
mat4
mat5 <- matrix(c(6,7,8,9), nr = 2)
mat5
cbind(mat4, mat5) # cbind combina por coluna
rbind(mat4, mat5) # rbind combina por linha


# Desconstruindo a Matriz (transformando-a em vetor)
c(mat4)

# Listas, Operações com Listas e Listas Nomeadas

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
getwd()

setwd("???/Cap02")
getwd()

# Use list() para criar listas

# Lista de strings
lista_caracter1 = list('A', 'B', 'C') # "[[1]]" - dimensão; "[1]" - elemento
lista_caracter1
lista_caracter2 = list(c("A", "A"), 'B', 'C')
lista_caracter2
lista_caracter3 = list(matrix(c("A", "A", "A", "A"), nr = 2), 'B', 'C')
lista_caracter3


# Lista de números inteiros
lista_inteiros = list(2, 3, 4)
lista_inteiros


# Lista de floats
lista_numerico = list(1.90, 45.3, 300.5)
lista_numerico
```

```
# Lista de números complexos
lista_complexos = list(5.2+3i, 2.4+8i)
lista_complexos


# Lista de valores lógicos
lista_logicos = list(TRUE, FALSE, FALSE)
lista_logicos


# Listas Compostas
lista_composta1 = list("A", 3, TRUE)
lista_composta1

lista1 <- list(1:10, c("Zico", "Ronaldo", "Garrincha"), rnorm(10))
lista1

?rnorm # distribuição normal

# Slicing (Fatiamento) da Lista
lista1[1] # neste caso, um vetor completo
lista1[2] # neste caso, um vetor de strings
lista1[[2]][1] # zico
lista1[[2]][1] = "Monica" # substitui o zico
lista1

# Para nomear as dimensões - Listas Nomeadas
names(lista1) <- c("inteiros", "caracteres", "numéricos")
lista1

# Criando uma lista nomeada (nomear dimensões diretamente)
vec_num <- 1:4
vec_char <- c("A", "B", "C", "D")
lista2 <- list(Numeros = vec_num, Letras = vec_char)
lista2
# ou
lista2 <- list(elemento1 = 3:5, elemento2 = c(7.2,3.5))
lista2


# Trabalhando com elementos específicos da lista
names(lista1) <- c("inteiros", "caracteres", "numéricos")
lista1

lista1$caracteres # lista$dimensao
length(lista1$inteiros) # tamanho da lista$dimensao
lista1$inteiros


# Verificar o comprimento da lista
length(lista1) # tamanho da lista (quantas dimensões)


# Podemos extrair um elemento específico dentro de cada nível da lista
lista1$caracteres[2] #lista$dimensao[elemento]


# Mode dos elementos (qual o tipo de dado)
mode(lista1$numéricos)
mode(lista1$caracteres)
```

```
# Combinando 2 listas com a função c
lista3 <- c(lista1, lista2)
lista3


# Transformando um vetor em lista (para receber qualquer tipo de dado)
v = c(1:3)
v
l  = as.list(v) # as.list converte um vetor em lista
l


# Unindo 2 elementos em uma lista
mat = matrix(1:4, nrow = 2)
mat
vec = c(1:9)
vec
lst = list(mat, vec) # transforma tudo numa lista
lst

# Operações com Strings

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
getwd()

setwd("???/Cap02")
getwd()

# String
texto <- "Isso é uma string!" # use aspas duplas ou simples
texto # variavel do tipo character (nao eh string em R)

x = as.character(3.14) # converte numero em texto
x
class(x) # função class informa o tipo do objeto


# Concatenando Strings
nome = "Nelson"; sobrenome = "Mandela"
paste(nome, sobrenome) # paste une strings em outra string
class(paste(nome, sobrenome))
cat(nome, sobrenome) # concatena
class(cat(nome, sobrenome))


# Formatando a saída com placeholder
sprintf("A %s é nota %d", "Data Science Academy", 10) # s - string; d - decimal



# Extraindo parte da string com substring
texto <- "Isso é uma string!"
substr(texto, start=12, stop=17)
?substr

# Contando o número de caracteres
nchar(texto)
```

```
# Alterando a capitalização entre maiúsculas e minúsculas
tolower("Histogramas e Elementos de Dados")
toupper("Histogramas e Elementos de Dados")


# Usando stringr - carregando o pacote para usar suas funções ----------------------
library(stringr)


# Dividindo uma string numa lista de caracteres
?strsplit
strsplit("Histogramas e Elementos de Dados", NULL) # null indica split um a um


# Dividindo uma string numa liste de caracteres, após o caracter espaço
strsplit("Histogramas e Elementos de Dados", " ")


# Trabalhando com strings
string1 <- c("Esta é a primeira parte da minha string e será a primeira parte do meu vetor",
             "Aqui a minha string continua, mas será transformada no segundo vetor")

string1 # vetor de strings

string2 <- c("Precisamos testar outras strings - @???!§$",
             "Análise de Dados em R")

string2

# Adicionando 2 strings (concatenando os vetores com str_c)
str_c(c(string1, string2), sep = "")


# Podemos contar quantas vezes um caracter aparece no texto (usando str_count)
str_count(string2, "s")


# Localiza a primeira e última posição em que o caracter aparece na string
str_locate_all(string2, "s")


# Substitui a primeira ocorrência de um caracter
str_replace(string2, "\\s", "") # \\s - espaço em branco


# Substitui todas as ocorrências de um caracter
str_replace_all(string2, "\\s", "")


# Detectando padrões nas strings - faz busca por caracteres(como um like), retornando True ou False
string1 <- "17 jan 2001"
string2 <- "1 jan 2001"
padrao <- "jan 20"
grepl(pattern = padrao, x = string1)
padrao <- "jan20"
grepl(pattern = padrao, x = string1)

# DataFrames e Operações com DataFrame

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding
```

```
# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
getwd()

setwd("???/Cap02")
getwd()

# Criando um dataframe (como uma planilha com linhas e colunas) vazio
# Muito útil por permitir representar qualquer tipo de dado
df <- data.frame()
class(df) # tipo do objeto - df(dataframe)
df


# Criando vetores vazios
nomes <- character()
idades <- numeric()
itens <- numeric()
codigos <- integer()

df <- data.frame(c(nomes, idades, itens, codigos))
df # vetor de vetores, onde cada vetor será uma coluna do df


# Criando vetores
pais = c("Portugal", "Inglaterra", "Irlanda", "Egito", "Brasil")
nome = c("Bruno", "Tiago", "Amanda", "Bianca", "Marta")
altura = c(1.88, 1.76, 1.53, 1.69, 1.68)
codigo = c(5001, 2183, 4702, 7965, 8890)


# Criando um dataframe de diversos vetores
pesquisa = data.frame(pais, nome, altura, codigo)
pesquisa


# Adicionando um novo vetor a um dataframe existente
olhos = c("verde", "azul", "azul", "castanho", "castanho")
pesq = cbind(pesquisa, olhos) # nova coluna com cbind
pesq


# Informações sobre o dataframe
str(pesq) # resumo dos tipos de dados do dataframe
dim(pesq) # numero de dimensões (linhas e colunas)
length(pesq) # tamanho do df


# Obtendo um vetor de um dataframe
pesq$pais # nomedataframe$nomecoluna
pesq$nome


# Extraindo um único valor (slicing do df, como de uma matriz)
pesq[1,1]
pesq[3,2]


# Número de Linhas e Colunas do df
nrow(pesq)
ncol(pesq)
```

```
# Primeiros elementos do dataframe - função head
head(pesq) # head traz 5 elementos por padrão
head(mtcars)


# Últimos elementos do dataframe
tail(pesq)
tail(mtcars)


# Data frames built-in do R
?mtcars # dataset de automóveis, usado para testes
mtcars
View(mtcars) # cria uma view (um df) do dataset no formato de tabela


# Filtro para um subset de dados que atendem a um critério
pesq[altura < 1.60,]
pesq[altura < 1.60, c('codigo', 'olhos')] # regra, trazendo apenas colunas específicas
pesq # fatiamento nao altera o df original


# Dataframes Nomeados
names(pesq) <- c("País", "Nome", "Altura", "Código", "Olhos")
pesq

colnames(pesq) <- c("Var 1", "Var 2", "Var 3", "Var 4", "Var 5")
rownames(pesq) <- c("Obs 1", "Obs 2", "Obs 3", "Obs 4", "Obs 5")
pesq


# Carregando um arquivo csv, montando um dataframe
?read.csv
pacientes <- data.frame(read.csv(file = 'pacientes.csv', header = TRUE, sep = ","))


# Visualizando o dataset
View(pacientes) # atenção: função View é com V maiúsculo
head(pacientes)
summary(pacientes) # resumos estatísticos para cada uma das variáveis


# Visualizando as variáveis
pacientes$Diabete
pacientes$status
pacientes$Status


# Histograma
hist(pacientes$Idade)


# Combinando dataframes
dataset_final <- merge(pesq, pacientes)
dataset_final
```

Capítulo 03:

```
# Solução Lista de Exercícios Capítulo 2

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/
03_BigDataAnalyticsRMicrosoftAzureMachineLearning/Cap03")
getwd()

# Exercício 1 - Crie um vetor com 12 números inteiros
vec <- c(1:12)
vec


# Exercício 2 - Crie uma matriz com 4 linhas e 4 colunas preenchida com números inteiros
mat <- matrix(c(1:16), nrow = 4, ncol = 4)
mat


# Exercício 3 - Crie uma lista unindo o vetor e matriz criados anteriormente
lst <- list(vec, mat)
lst


# Exercício 4 - Usando a função read.table() leia o arquivo do link abaixo para uma dataframe
# http://data.princeton.edu/wws509/datasets/effort.dat
df <- data.frame(read.table("http://data.princeton.edu/wws509/datasets/effort.dat"))
class(df)
df
View(df) # para visualizar em formato de tabela


# Exercício 5 - Transforme o dataframe anterior, em um dataframe nomeado com os seguintes labels:
# c("config", "esfc", "chang")
# dataframe nomeado é aquele com títulos nas colunas
names(df) = c("config", "esfc", "chang")
names(df) = c("Col1", "Col2", "Col3")
df


# Exercício 6 - Imprima na tela o dataframe iris,
# verifique quantas dimensões existem no dataframe iris, imprima um resumo do dataset
iris
class(iris) # classe
dim(iris) # dimensões
summary(iris) # sumario ou resumo estatístico
str(iris) # resumo e o tipo de dado de cada uma das variaveis
View(iris) # ver em formato de tabela


# Exercício 7 - Crie um plot simples usando as duas primeiras colunas do dataframe iris
# dataframe é o nome do objeto; dataset é o conjunto de dados (são basicamente a mesma coisa)
plot(iris$Sepal.Length, iris$Sepal.Width)


# Exercício 8 - Usando s função subset, crie um novo dataframe com o conjunto de dados do dataframe iris em que
Sepal.Length > 7
```

```
# Dica: consulte o help para aprender como usar a função subset()
?subset
iris1 <- subset(iris, Sepal.Length > 7) # função subset(objeto, regra)
View(iris1)


# Exercícios 9 (Desafio) - Crie um dataframe que seja cópia do dataframe iris e usando a função slice(),
# divida o dataframe em um subset de 15 linhas
# Dica 1: você vai ter que instalar e carregar o pacote dplyr
# Dica 2: Consulte o help para aprender como usar a função slice()
novo_iris <- iris # criando o novo dataset (para nao alterar o ds original)
novo_iris
install.packages("dplyr") # instalando o pacote
library(dplyr) # carregando o pacote na sessão
?slice # help da função slice do pacote dplyr
slice(novo_iris, 1:15) # função slice(objeto, regra)
class(slice(novo_iris, 1:15))


# Exercícios 10 - Use a função filter no seu novo dataframe criado no item anterior
# e retorne apenas valores em que Sepal.Length > 6
# Dica: Use o RSiteSearch para aprender como usar a função filter
RSiteSearch('filter') # help da função filter num site buscador
filter(novo_iris, Sepal.Length > 7) # função filter(objeto, regra)

# Big Data na Prática 1 - Analisando a Temperatura Média nas Cidades Brasileiras

# 1. DEFINIÇÃO DO PROBLEMA

# O primeiro passo é definir o problema de negócio.
# Não se analisa dados de maneira aleatória.

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/
03_BigDataAnalyticsRMicrosoftAzureMachineLearning/Cap03")
getwd()

# 2. COLETA DOS DADOS

# Dataset:
# Berkeley Earth (Temperaturas de várias cidades do mundo nos últimos 100 anos)
# http://berkeleyearth.org/data
# TemperaturasGlobais.csv ~ 78 MB (zip) ~ 496 MB (unzip)
# Façs o download do arquivo zip no link abaixo e descompacte na mesma pasta onde está este script.
# https://drive.google.com/open?id=1nSwP3Y0V7gncbnG_DccNhrTRxmUNqMqa
# Obs: Descompacte os arquivos dentro do diretório onde se encontram os scripts!

# Instalando e Carregando Pacotes
# Obs: os pacotes precisam ser instalados apenas uma vez. Se já instalou em outros scripts, não é necessário instalar
novamente!
install.packages("readr")
install.packages("data.table")
install.packages("dplyr")
install.packages("ggplot2")
library(readr)
library(dplyr)
library(ggplot2)
library(scales) # essa função só precisa carregar
```

```
library(data.table)


# Carregando os dados (Usando um timer para comparar o tempo de carregamento com diferentes funções)
# As 3 funções fazem a mesma coisa, só que em tempos diferentes.

# Usando read.csv2() - uma das mais antigas e mais lentas (leitura do arquivo em 4 minutos)
# carregando dataframe com função read.csv2(diretorio/arquivo), e
# usando system.time para medir o tempo de execução.
system.time(df_teste1 <- read.csv2("TemperaturasGlobais/TemperaturasGlobais.csv"))

# Usando read.table() - mais rápida que a função read.csv2, mas ainda antiga e lenta
system.time(df_teste2 <- read.table("TemperaturasGlobais/TemperaturasGlobais.csv"))

# Usando fread() - muito mais rápida (leitura do arquivo em até 2 segundos e menos da metade do tamanho em
memória)
?fread # para ajuda e uma descrição melhor da função fread
system.time(df <- fread("TemperaturasGlobais/TemperaturasGlobais.csv"))


# Criando subsets dos dados carregados
cidadesBrasil <- subset(df, Country == 'Brazil')
cidadesBrasil <- na.omit(cidadesBrasil) # na.omit: omitindo valores missing (not available)
head(cidadesBrasil)
nrow(df) # nrow: identificando o número de linhas do dataset ou dataframe
nrow(cidadesBrasil)
dim(cidadesBrasil)

# 3. PRE-PROCESSAMENTO DOS DADOS

# Preparação e Organização

# Convertendo as Datas
# Transformando uma coluna e atualizando ela mesma
cidadesBrasil$dt <- as.POSIXct(cidadesBrasil$dt,format='%Y-%m-%d')
cidadesBrasil$Month <- month(cidadesBrasil$dt)
cidadesBrasil$Year <- year(cidadesBrasil$dt)

# Carregando os subsets - para criar gráficos de cidades específicas (não todo o subset)
# Palmas
plm <- subset(cidadesBrasil, City == 'Palmas')
plm <- subset(plm, Year %in% c(1796,1846,1896,1946,1996,2012))

# Curitiba
crt <- subset(cidadesBrasil, City == 'Curitiba')
crt <- subset(crt, Year %in% c(1796,1846,1896,1946,1996,2012))

# Recife
recf <- subset(cidadesBrasil, City=='Recife')
recf <- subset(recf,Year %in% c(1796,1846,1896,1946,1996,2012))


# Construindo os Plots
p_plm <- ggplot(plm, aes(x = (Month), y = AverageTemperature, color = as.factor(Year))) +
  geom_smooth(se = FALSE,fill = NA, size = 2) +
  theme_light(base_size = 20) +
  xlab("Mês")+
  ylab("Temperatura Média") +
  scale_color_discrete("") +
  ggtitle("Temperatura Média ao longo dos anos em Palmas") +
  theme(plot.title = element_text(size = 18))

p_crt <- ggplot(crt, aes(x = (Month), y = AverageTemperature, color = as.factor(Year))) +
```

```
  geom_smooth(se = FALSE,fill = NA, size = 2) +
  theme_light(base_size = 20) +
  xlab("Mês")+
  ylab("Temperatura") +
  scale_color_discrete("") +
  ggtitle("Temperatura Média ao longo dos anos em Curitiba") +
  theme(plot.title = element_text(size = 18))

p_recf <- ggplot(recf, aes(x = (Month), y = AverageTemperature, color = as.factor(Year))) +
  geom_smooth(se = FALSE,fill = NA, size = 2) +
  theme_light(base_size = 20) +
  xlab("Mês")+
  ylab("Temperatura Média") +
  scale_color_discrete("") +
  ggtitle("Temperatura Média ao longo dos anos em Recife") +
  theme(plot.title = element_text(size = 18))

# 4. ANALISE DOS DADOS

# Plotando - imprimindo na tela
p_plm
p_crt
p_recf

# Fatores

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/
03_BigDataAnalyticsRMicrosoftAzureMachineLearning/Cap03")
getwd()

# Criando um vetor de palavras:
vec1 <- c("Macho","Femea","Femea","Macho","Macho")
vec1
# Convertendo o vetor em um fator:
fac_vec1 <- factor(vec1)
fac_vec1
class(vec1)
class(fac_vec1)

# Variáveis categóricas nominais
# Não existe uma ordem implícita
animais <- c("Zebra", "Pantera", "Rinoceronte", "Macaco", "Tigre")
animais
class(animais)
fac_animais <- factor(animais)
fac_animais
class(fac_animais)
levels(fac_animais) # levels são níveis do fator

# Variáveis categóricas ordinais
# Possuem uma ordem natural
grad <- c("Mestrado", "Doutorado", "Bacharelado", "Mestrado", "Mestrado")
grad
fac_grad <- factor(grad, order = TRUE, levels = c("Doutorado", "Mestrado", "Bacharelado"))
fac_grad
levels(fac_grad)
```

```
# Sumarizar os dados fornece uma visão geral sobre o conteúdo das variáveis
summary(fac_grad)
summary(grad)

vec2 <- c("M", "F", "F", "M", "M", "M", "F", "F", "M", "M", "M", "F", "F", "M", "M")
vec2
fac_vec2 <- factor(vec2)
fac_vec2
levels(fac_vec2) <- c("Femea", "Macho") # renomeando os níveis de F ou M para Fêmea ou Macho
fac_vec2
summary(fac_vec2)
summary(vec2)


# Mais exemplos
# Categorizando valores numéricos:
data = c(1,2,2,3,1,2,3,3,1,2,3,3,1)
fdata = factor(data)
fdata


rdata = factor(data, labels = c("I","II","III"))
rdata

# Fatores Não-Ordenados
# R apenas criou os níveis, o que não significa que exista uma hierarquia.
set1 <- c("AA", "B", "BA", "CC", "CA", "AA", "BA", "CC", "CC")
set1

# Transformando os dados.
f.set1 <- factor(set1)
f.set1
class(f.set1)
is.ordered(f.set1) # perguntando se o fator é ordenando - FALSE

# Fatores Ordenados
# especificando os níveis e definindo sua ordenação:
o.set1 <- factor(set1,
                 levels = c("CA", "BA", "AA", "CC", "B"),
                 ordered = TRUE)

o.set1
is.ordered(o.set1)

as.numeric(o.set1)
table(o.set1)


# Fatores e Dataframes
df <- read.csv2("etnias.csv", sep = ',') # criando df a partir de um csv
View(df)


# Variáveis do tipo fator
# Importante observar que o R gera fatores de maneira automatica. Isso, às vezes, pode ser um problema.
str(df)

# Níveis dos fatores
# Internamente, o R armazena valores inteiros e faz um mapeamento para as strings (em ordem alfabética)
# e agrupa as estatísticas por níveis. Agora, se fizermos sumarização de estatísticas, é possível visualizar
# a contabilização para cada categoria
levels(df$Etnia) # Mostra os níveis da variável Etnia
summary(df$Etnia) # sumarizando a variável
```

```
# Plot
# Agora se fizermos um plot, temos um boxplot para estas variáveis categóricas
plot(df$Idade~df$Etnia, xlab = 'Etnia', ylab = 'Idade', main = 'Idade por Etnia')


# Regressão
summary(lm(Idade~Etnia, data = df))

# Convertendo uma coluna em variável categórica. Isso criará um fator não-ordenado
df
str(df)
df$Estado_Civil.cat <- factor(df$Estado_Civil, labels = c("Solteiro", "Casado", "Divorciado"))
df
str(df)

# Fatores e Dataframes - Compreendendo a Ordem dos Fatores

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/
03_BigDataAnalyticsRMicrosoftAzureMachineLearning/Cap03")
getwd()

# Níveis dos fatores
# Internamente, o R armazena valores inteiros e faz um mapeamento para as strings (em ordem alfabética)
# e agrupa as estatísticas por níveis.

# Criando vetores
vec1 <- c(1001, 1002, 1003, 1004, 1005)
vec2 <- c(0, 1, 1, 0, 2)
vec3 <- c('Verde','Laranja','Azul','Laranja','Verde')

# Unindo os vetores em um dataframe
df <- data.frame(vec1, vec2, vec3)
df

# Verificando que o R categorizou a última coluna como fator
str(df)

# Verificando os níveis do fator. Perceba que os níveis estão categorizados em ordem alfabética
levels(df$vec3)

# Criando uma outra coluna e atribuindo labels
df$cat1 <- factor(df$vec3, labels = c("cor2", "cor1", "cor3"))
df

# Internamente, os fatores são registrados como inteiros, mas a ordenação segue a ordem alfabética
# das strings
str(df)

# Veja como foi feita a atribuição:
# Azul = cor2
# Laranja = cor1
# Verde = cor3
# Ou seja, os vetores com os labels, seguiram a ordem alfabética dos níveis classificados pelo R

# Criando uma outra coluna e atribuindo labels
```

```
# Ao aplicarmos a função factor() a coluna vec2, internamente o R classificou em ordem alfabética
# e quando atribuímos os labels, foi feita a associação.
df$cat2 <- factor(df$vec2, labels = c("Divorciado", "Casado", "Solteiro"))
df
str(df)
levels(df$cat2)

# Estruturas de Controle

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/
03_BigDataAnalyticsRMicrosoftAzureMachineLearning/Cap03")
getwd()

# If-else
x = 25
if (x < 30)
  {"Este número é menor que 30"}


# Chaves não são obrigatórios, mas altamente recomendados
if (x < 30)
  "Este número é menor que 30"


# Else
if (x < 7) {
  "Este número é menor que 7"
} else {
  "Este número não é menor que 7"
  }


# Comandos podem ser ainhados
x = 7
if (x < 7) {
  "Este número é menor que 7"
} else if(x == 7) {
  "Este é o número 7"
}else{
  "Este número não é menor que 7"
}


# Ifelse
x = 5
ifelse (x < 6, "Correto!", NA)

x = 9
ifelse (x < 6, "Correto!", NA)


# Expressões ifelse aninhadas
x = c(7,5,4)
ifelse(x < 5, "Menor que 5",
       ifelse(x == 5, "Igual a 5", "Maior que 5"))
```

```
# Estruturas if dentro de funções
func1 <- function(x,y){
  ifelse(y < 7, x + y, "Não encontrado")
}

func1(4,2)
func1(40,7)


# Rep - rep(repita x, y vezes)
rep(rnorm(10), 5)


# Repeat
x = 1
repeat {
  x = x + 3
  if (x > 99)
    break
  print(x)}


# Loop For
for (i in 1:20) {print(i)}
for (q in rnorm(10)) {print(q)}


# Ignora alguns elementos dentro do loop
for(i in 1:22){
  if(i == 13 | i == 15)
    next # se condicao verdadeira, nao imprime
  print (i)}


# Interromper o loop
for(i in 1:22){
  if(i == 13)
    break
  print (i)}


# Loop While
x = 1
while(x < 5){
  x = x + 1
  print(x)
}

# O loop while não será executado
y = 6
while(y < 5){
  y = y+10
  print(y)
}

# Funções

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
```

```
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/
03_BigDataAnalyticsRMicrosoftAzureMachineLearning/Cap03")
getwd()

# Help
?sample
args(sample) # args(): mostra os argumentos da função
args(mean)
args(sd)


# Funções Built-in
abs(-43) # valor absoluto de um número
sum(c(1:5)) # soma dos elementos
mean(c(1:5)) # média dos elementos
round(c(1.1:5.8)) # arredonda valor
rev(c(1:5)) # reverte os elementos
seq(1:5) # sequencia de numeros
sort(rev(c(1:5))) # ordena elementos
append(c(1:5), 6) # adiciona elemento ao final de uma lista

vec1 <- c(1.5, 2.5, 8.4, 3.7, 6.3)
vec2 <- rev(vec1)
vec2


# Funções dentro de funções
plot(rnorm(10))
mean(c(abs(vec1), abs(vec2)))


# Criando funções
myfunc <- function(x) { x + x }
myfunc(10)
class(myfunc)

myfunc2 <- function(a, b) {
  valor = a ^ b
  print(valor)
}
myfunc2(3, 2)

jogando_dados <- function() { # função sem parâmetros
  num <- sample(1:6, size = 1) #Local
  num
}
jogando_dados()

# Escopo
print(num) # variavel num tem escopo local (dentro de jogando_dados)
num <- c(1:6)
num #Global


# Funções sem número definido de argumentos
vec1 <- (10:13)
vec2 <- c("a", "b", "c", "d")
vec3 <- c(6.5, 9.2, 11.9, 5.1)

myfunc3 <- function(...){
  df = data.frame(cbind(...))
  print(df)
}
```

```
myfunc3(vec1)
myfunc3(vec2, vec3)
myfunc3(vec1, vec2, vec3)


# Funções Built-in - Não tente recriar a roda
# Comparação de eficiência entre função vetorizada e função "vetorizada no R"

x <- 1:10000000

# Função que calcula a raiz quadrada de cada elemento de um vetor de números
meu_sqrt <- function(numeros) {
  resp <- numeric(length(numeros))
  for(i in seq_along(numeros)) {
    resp[i] <- sqrt(numeros[i])
  }
  return(resp)
}

# calculando tempo de execução entre as duas funções:
system.time(x2a <- meu_sqrt(x))
system.time(x2b <- sqrt(x))

# Sua máquina pode apresentar resultado diferente por conta da precisão de cálculo do processador.
identical(x2a, x2b)

# Família Apply - Uma Forma Elegante de Fazer Loops

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/
03_BigDataAnalyticsRMicrosoftAzureMachineLearning/Cap03")
getwd()

#  apply() - arrays e matrizes
# tapply() - os vetores podem ser divididos em diferentes subsets
# lapply() - vetores e listas
# sapply() - versão amigável da lapply
# vapply() - similar a sapply, com valor de retorno modificado
# rapply() - similar a lapply()
# eapply() - gera uma lista
# mapply() - similar a sapply, multivariada
# by - Não pertence à família apply - função base, similar em coportamento

# Se você estiver trabalhando com os objetos:

# list, numeric, character (list/vecor) => sapply ou lapply
# matrix, data.frame (agregação por coluna) => by / tapply
# Operações por linha ou operações específicas => apply

# Usando um Loop
lista1 <- list(a = (1:20), b = (35:67))

# Calculando o total de cada elemento da lista com loop for
valor_a = 0
valor_b = 0

for (i in lista1$a){
```

```
  valor_a = valor_a + i
}

for (j in lista1$b){
  valor_b = valor_b + j
}

print(valor_a)
print(valor_b)

# Calculando cada elemento da lista com sapply
?sapply
sapply(lista1, sum)

# Aplicando funções com sapply
sapply(lista1, mean)


# apply()
?apply

x <- matrix(c(20, 13, 65, 32, 45, 12, 76, 49, 82), nr = 3, byrow = T)
x

apply(x, mean) # erro: faltando parametro Margin (1-linha/2-coluna)
apply(x, 1, mean)
apply(x, 2, mean)
apply(x, 1, plot)

resultapply <- apply(x, 1, mean)
resultapply

# Aplicando apply() a um Dataframe
escola <- data.frame(Aluno = c('Bob', 'Tereza', 'Marta', 'Felipe', 'Zacarias', 'Elton'),
                     Fisica = c(91, 82, 75, 97, 62, 74),
                     Matematica = c(99, 100, 86, 92, 91, 87),
                     Quimica = c(56, 72, 49, 68, 59, 77))

escola
escola$Matematica

# Calculando a média por aluno
escola$Media = NA # criando a coluna Media no df escola
escola

escola$Media = apply(escola[,c(2, 3, 4)], 1, mean) # para todas as linhas, colunas 2, 3, 4
escola
escola$Media = round(escola$Media) # arredondando valor da media
escola


### tapply()
?gl # cria niveis de fator
tabela_basquete <-
  data.frame(equipe = gl(5, 5, labels = paste("Equipe", LETTERS[1:5])),
             jogador = sample(letters, 25), # letters - conjunto de letras do R
             num_cestas = floor(runif(25, min=0, max=50)))

View(tabela_basquete)
summary(tabela_basquete)

# Como calcular o total de cestas por Equipe?
# tapply() vs sqldf
```

```
?tapply

install.packages('sqldf') # sqldf permite usar linguagem sql no R
library(sqldf)

sqldf("select equipe, sum(num_cestas) from tabela_basquete group by equipe")
tapply(tabela_basquete$num_cestas, tabela_basquete$equipe, sum)
tapply(tabela_basquete$num_cestas, tabela_basquete$equipe, mean)


### by - Aplica uma função a um data frame, dividido por fatores
?by

dat <- data.frame(species=c(rep(c(1,2,3), each=5)), # coluna
                  petal.length=c(rnorm(5, 4.5, 1), # coluna
                                 rnorm(5, 4.5, 1),
                                 rnorm(5, 5.5, 1)),
                  petal.width=c(rnorm(5, 2.5, 1), # coluna
                                rnorm(5, 2.5, 1),
                                rnorm(5, 4, 1)))

dat$species <- factor(dat$species) # convertendo em fator
View(dat)

# calcular o comprimento médio da pétala para cada espécie
by(dat, dat$species, function(x){
  mean.pl <- mean(x$petal.length)
})


### lapply() e sapply - mesmos resultados, com saídas um pouco diferentes
?lapply

lista1 <- list(a = (1:10), b = (45:77))
lista1
lapply(lista1, sum)
sapply(lista1, sum)


### vapply() - aplica função a uma lista ou vetor
?vapply

# A função fivenum() retorna 5 estatísticas do conjunto de dados: (minimum, lower-hinge, median, upper-hinge,
maximum)
# https://stat.ethz.ch/R-manual/R-patched/library/stats/html/fivenum.html
vapply(lista1,
       fivenum,
       c(Min. = 0,
         "1stQu." = 0,
         Median = 0,
         "3rd Qu." = 0,
         Max = 0))


# replicate
?replicate
replicate(7, runif(10))


### mapply() - aplica uma função a multiplas listas ou argumentos de um vetor
?mapply
mapply(rep, 1:4, 4:1)
```

```
### rapply() - recursivamente, aplica uma função a uma lista
?rapply

lista2 <- list(a = c(1:5), b = c(6:10))
lista2

rapply(lista2, sum)
rapply(lista2, sum, how = "list") # how muda o formato da impressão/saída

# Funções Especiais

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/
03_BigDataAnalyticsRMicrosoftAzureMachineLearning/Cap03")
getwd()

### unlist() - Produz um vetor com os elementos da lista (Flatten Lists)
?unlist

lst1 <- list(6, "b", 15)
lst1
class(lst1)

unlist(lst1)
vec1 <- unlist(lst1)
class(vec1)
vec1

lst2 <- list(v1 = 6, v2 = list(381, 2190), v3 = c(30, 217))
lst2

unlist(lst2)

mean(unlist(lst2))
round(mean(unlist(lst2))) # unlist > media > arredonda


### do.call()
# Executa uma função em um objeto
# *** ATENÇÃO ***
# As funções da família apply aplicam uma função a cada elemento de um objeto (substitui um loop)
# A função do.call aplica uma função ao objeto inteiro e não a cada elemento individualmente
?do.call

# criando um conjunto de dados:
data <- list()
N <- 100

for (n in 1:N) {
  data[[n]] = data.frame(index = n, char = sample(letters, 1), z = rnorm(1))
}

head(data)

do.call(rbind, data) # rbind - une os elementos da lista
class(do.call(rbind, data))
```

```
### lapply() x do.call() - observando as diferenças entre do.call e familia apply:
y <- list(1:3, 4:6, 7:9)
y

lapply(y, sum) # soma para cada elemento da lista (soma agrupada)
do.call(sum, y) # soma de todos os elementos de uma vez (soma geral)


### Pacote plyr
# O resultado da função lapply() pode ser obtido de outras formas

install.packages('plyr')
library(plyr)

y <- list(1:3, 4:6, 7:9)
y

ldply(y, sum) # ldply é função do pacote plyr


### strsplit()
# Divide uma string ou vetor de caracteres

texto <- "Data Science Academy"
strsplit(texto, " ") # saída um vetor de palavras

texto <- "Data Science Academy"
strsplit(texto, "")


dates <- c("1998-05-23", "2008-12-30", "2009-11-29")
temp  <- strsplit(dates, "-")
temp
class(temp) # DICA: sempre que criar um novo objeto, verifique o tipo desse objeto

# Transforma a lista em matriz, fazendo antes um unlist()
matrix(unlist(temp), ncol = 3, byrow = TRUE)

Names <- c("Brin, Sergey", "Page, Larry",
           "Dorsey, Jack", "Glass, Noah",
           "Williams, Evan", "Stone, Biz")

temp <- strsplit(Names, ", ")
temp

# Retirando palavras repetidas
frase <- "Muitas vezes temos que repetir algo diversas vezes e essas diversas vezes parecem algo estranho"
palavras <- strsplit(frase, " ")[[1]]
palavras
unique(tolower(palavras))


### strplit() com dataframes
antes = data.frame(attr = c(1,30,4,6), tipo = c('pao_e_agua','pao_e_agua_2'))
antes
strsplit(as.character(antes$tipo),'_e_') # converte tipo para caracter e faz o split, montando uma lista

library(stringr) # carrega pacote stringr
str_split_fixed(antes$tipo, "_e_", 2) # aqui, o resultado final é uma matriz

# Usando do.call()
antes = data.frame(attr = c(1,30,4,6), tipo = c('pao_e_agua','pao_e_agua_2'))
antes
```

```
depois <- strsplit(as.character(antes$tipo),'_e_') # resultado uma lista
depois
do.call(rbind, depois) # converte a lista em matriz

# Usando dplyr e tidyr
install.packages("dplyr")
install.packages("tidyr")
library(dplyr)
library(tidyr)

antes <- data.frame(
  attr = c(1, 30 ,4 ,6 ),
  tipo = c('pao_e_agua','pao_e_agua_2')
)

antes %>%
  separate(tipo, c("pao", "agua"), "_e_")

# Pacotes e Instalação de Pacotes

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap03")
getwd()

# De onde vem as funções? Pacotes (conjuntos de funções)
# Quando você inicia o RStudio, alguns pacotes são
# carregados por padrão

# Busca os pacotes carregados
search()

### DICA:
# Para buscar a lista de pacotes, entre no site do R > CRAN > Selecione o Mirror >
# > Clique em Packages > Selecione Table of available packages (por data ou nome)

# Instala e carrega os pacotes
# install.packages(c("ggvis", "tm", "dplyr")) # soh se instala 1x
library(ggvis)
library(tm)
require(dplyr)

search()
?require
detach(package:dplyr) # descarregar o pacote da memoria

# Lista o conteúdo dos pacotes
?ls
ls(pos = "package:tm") # lista todas as funções do pacote tm
ls(getNamespace("tm"), all.names = TRUE) # lista tudo do pacote

# Lista as funções de um pacote
lsf.str("package:tm") # lsf.str - lista funções e seus argumentos
lsf.str("package:ggplot2")
# install.packages("ggplot2")
library(ggplot2)
lsf.str("package:ggplot2")
detach(package:ggplot2)
```

```
# R possui um conjunto de datasets preinstalados.

library(MASS)
data() # lista os datasets disponiveis em R

?lynx
head(lynx)
head(iris)
tail(lynx)
summary(lynx)

plot(lynx)
hist(lynx)
head(iris)
iris$Sepal.Length

# erro: necessário incluir o caminho, pois foi usado apenas o nome da coluna
sum(Sepal.Length) # erro
?attach
attach(iris)
sum(Sepal.Length) # igual a sum(iris$Sepal.Length)


### Para criação dos pacotes em R:

# Podemos criar pacotes R usando o RStudio. Para testar os pacotes
# que você criar, será necessário instalar estes 2 pacotes abaixo:
# install.packages('testthat')
# install.packages('devtools')

# Você pode criar seus pacotes R usando apenas o RStudio.
# Entretanto, dependendo dos recursos que serão usados no seu pacote,
#
# você precisa instalar as ferramentas de desenvolvimento para pacotes
# R, de acordo com seu sistema operacional:

# Windows
# Baixar o Rtools: https://cran.rstudio.com/bin/windows/Rtools
# Baixar o LaTeX: http://miktex.org/download

# MacOS:
# Baixar o Xcode:
# http://itunes.apple.com/us/app/xcode/id497799835?mt=12
# Instalar Command Line Tools dentro do Xcode

# Linux:
# sudo apt-get install r-base-dev texlive-full
# sudo apt-get build-dep r-base-core

# Expressões Regulares

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
getwd() # depois, adicione a pasta do capítulo (abaixo:)
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap03")
getwd()

### LISTA DE FUNCOES:
# grep(pattern, x, ignore.case = FALSE, perl = FALSE, value = FALSE, fixed = FALSE, useBytes = FALSE, invert =
```

```
FALSE)
# grepl(pattern, x, ignore.case = FALSE, perl = FALSE, fixed = FALSE, useBytes = FALSE)
# sub(pattern, replacement, x, ignore.case = FALSE, perl = FALSE, fixed = FALSE, useBytes = FALSE)
# gsub(pattern, replacement, x, ignore.case = FALSE, perl = FALSE, fixed = FALSE, useBytes = FALSE)
# regexpr(pattern, text, ignore.case = FALSE, perl = FALSE, fixed = FALSE, useBytes = FALSE)
# gregexpr(pattern, text, ignore.case = FALSE, perl = FALSE, fixed = FALSE, useBytes = FALSE)

# Vetor de String
str <- c("Expressões", "regulares", "em linguagem R",
         "permitem a busca de padrões", "e exploração de textos",
         "podemos buscar padrões em dígitos",
         "como por exemplo",
         "10992451280")

length(str) # tamanho
str

# grep() - Verificação de Padrão
?grep
grep("ex", str, value = F) # com F traz apenas o indice do vetor
grep("ex", str, value = T) # com T traz o conteudo do indice
grep("\\d", str, value = F)
grep("\\d", str, value = T) # d é dígito


# grepl() - Verificação de Padrão, retornando True ou False
?grepl
grepl("\\d+", str) # d+ é digito e palavras
grepl("\\D", str) # D é não dígito


# gsub() - Substitui o padrão por outra coisa
?gsub
gsub("em", "***", str)
gsub("ex", "EX", str, ignore.case = T)

# sub() - Substitui o padrão por outra coisa
sub("em", "EM", str)


# regexpr() - retorna posição onde foi encontrando o padrão
frase <- "Isso é uma string."
regexpr(pattern = "u", frase)

# gregexpr() - retorna posição onde foi encontrando o padrão
gregexpr(pattern = "u", frase)


### Removento caracteres indesejados:
str2 <- c("2678 é maior que 45 - @???!§$",
          "Vamos escrever 14 scripts R")
str2

# gsub()
gsub("\\d", "", str2)
gsub("\\D", "", str2)
gsub("\\s", "", str2) # \\ s é espaço
gsub("[iot]", "Q", str2)
gsub("[[:punct:]]", "", str2)


### TABELA COM EXPRESSOES REGULARES EM R:
# D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap03/
```

TabelaExpRegularesemR.pdf

```
# Datas e Hora

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap03")
getwd()

# Hora e Data do sistema
hoje <- Sys.Date() # data do sistema operacional aaaa-mm-dd
hoje
class(hoje)
Sys.time() # data e hora do sistema operacional aaaa-mm-dd hh:mm:ss fuso
Sys.timezone()


# Data - representada por Date
# Armazenados como número de dias desde 1 de Janeiro de 1970 (internamente)

# Time - representado por POSIXct
# Armazenados como número de segundos desde 1 de Janeiro de 1970

# Formatando Datas
# %d: dia do mês em 2 dígitos (13)
# %m: mês em 2 digitos (01)
# %y: ano em 2 dígitos (82)
# %Y: ano em 4 dígitos (1982)
# %A: dia da semana (Friday)
# %a: dia da semana abreviado (Fri)
# %B: mês (July)
# %b: mês abreviado (Jul)


# Formatando hora
# %H: hora (00-23)
# %M: minuto
# %S: segundo
# %T: formado reduzido para %H:%M:%S
?strptime # funções para conversões em formatos de data hora


# Formatando a saída - as.Date()
as.Date("2018-06-28") # converte string para data
as.Date("Jun-28-18", format = "%b-%d-%y")
as.Date("28 June, 2018", format = "%d %B, %Y")


# Função format() - transforma data em formato desejado
Sys.Date()
?format
format(Sys.Date(), format = "%d %B, %Y")
format(Sys.Date(), format = "Hoje é %A!")


# Convertendo Datas - as.POSIXct
date1 <- "Jun 13, '96 hours:23 minutes:01 seconds:45"
date1_convert <- as.POSIXct(date1, format = "%B %d, '%y hours:%H minutes:%M seconds:%S")
date1_convert
```

```
### Operações com Datas
data_de_hoje <- as.Date("2016-06-25", format = "%Y-%m-%d")
data_de_hoje
data_de_hoje + 1

my_time <- as.POSIXct("2016-05-14 11:24:134")
my_time
my_time + 1

data_de_hoje - as.Date(my_time) # diferença entre as datas, uma no formato time
data_de_hoje - my_time # erro - necessario conversao


# Convertendo Data em formato específico
# O vetor de números pode representar o número de dias, horas ou minutos (de acordo com o que você quer converter)
# A Linguagem R considera o ponto de início a data de 01 de Janeiro de 1970 e contabiliza o total
# de horas, minutos ou segundos, aquilo que o vetor numérico representar
dts = c(1127056501,1104295502,1129233601,1113547501,1119826801,1132519502,1125298801,1113289201)
mydates = dts

# POSIXct, armazena os segundos desde uma data específica,
# convertendo os valores numéricos (que podem representar horas, minutos ou segundos) desde 01 de Janeiro de 1970
# POSIXt é a classe principal e POSIXct e POSIXlt são subclasses.
# Poderíamos usar aqui apenas POSIXct, que é a subclasse (mas não podemos usar apenas a classe principal)
class(mydates) = c('POSIXt','POSIXct')
mydates
class(mydates)

mydates = structure(dts, class = c('POSIXt','POSIXct'))
mydates


# Função ISODate - data no formato padrao
b1 = ISOdate(2011,3,23)
b1
b2 = ISOdate(2012,9,19)
b2
b2 - b1 # diferença em dias

difftime(b2, b1, units = 'weeks') # diferença em semanas
?difftime


### Pacote lubridate - manipulação de datas, de forma mais amigável
?lubridate
install.packages("lubridate")
require(lubridate)

ymd("20180604")
mdy("06-04-2018")
dmy("04/06/2018")

chegada <- ymd_hms("2016-06-04 12:00:00", tz = "Pacific/Auckland")
partida <- ymd_hms("2011-08-10 14:00:00", tz = "Pacific/Auckland")

chegada
partida

second(chegada)
second(chegada) <- 23 # atribuindo segundos a data
chegada
```

```
wday(chegada) # dia da semana
wday(chegada, label = TRUE) # fator de dia da semana
class(chegada)

# Cria um objeto que especifica a data de início e data de fim
interval(chegada, partida)


tm1.lub <- ymd_hms("2020-05-24 23:55:26")
tm1.lub

tm2.lub <- mdy_hm("05/25/20 08:32")
tm2.lub

year(tm1.lub)
week(tm1.lub)

tm1.dechr <- hour(tm1.lub) + minute(tm1.lub)/60 + second(tm1.lub)/3600
tm1.dechr
force_tz(tm1.lub, "Pacific/Auckland")


### Gerando um dataframe de datas
sono <- data.frame(bed.time = ymd_hms("2013-09-01 23:05:24", "2013-09-02 22:51:09",
                                      "2013-09-04 00:09:16", "2013-09-04 23:43:31", "2013-09-06 00:17:41", "2013-09-06
22:42:27",
                                      "2013-09-08 00:22:27"), rise.time = ymd_hms("2013-09-02 08:03:29", "2013-09-03 07:34:21",
                                                                                  "2013-09-04 07:45:06", "2013-09-05 07:07:17", "2013-09-06
08:17:13", "2013-09-07 06:52:11",
                                                                                  "2013-09-08 07:15:19"), sleep.time = dhours(c(6.74, 7.92, 7.01,
6.23, 6.34, 7.42, 6.45)))
sono
# aplicando uma formula e incluindo o atributo eficiencia no data frame
sono$eficiencia <- round(sono$sleep.time/(sono$rise.time - sono$bed.time) * 100, 1)
sono


# Gerando um plot a partir de datas
par(mar = c(5, 4, 4, 4))
plot(round_date(sono$rise.time, "day"), sono$eficiencia, type = "o", col = "blue", xlab = "Manhã", ylab = NA)
par(new = TRUE)
plot(round_date(sono$rise.time, "day"), sono$sleep.time/3600, type = "o", col = "red", axes = FALSE, ylab = NA, xlab
= NA)
axis(side = 4)
mtext(side = 4, line = 2.5, col = "red", "Duração do Sono")
mtext(side = 2, line = 2.5, col = "blue", "Eficiência do Sono")

# Operadores de Atribuição

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap03")
getwd()

# Nesse contexto, tudo igual:
vec1 = 1:4
vec2 <- 1:4

class(vec1)
```

```
class(vec2)

typeof(vec1)
typeof(vec2)

# No contexto de uma função, isso muda:
mean(x = 1:10) # x é descartado depois da função (só existem em tempo de execução)
x

mean(x <- 1:10) # x é mantido depois da função (mais utilizado pelos profissionais de R)
x
```

Capítulo 04:

```
# Solução Lista de Exercícios - Capítulo 3

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap04")
getwd()

# Exercício 1 - Pesquise pela função que permita listar todos os arquivos no
# diretório de trabalho
list.files()

# Exercício 2 - Crie um dataframe a partir de 3 vetores: um de caracteres,
# um lógico e um de números
charac_vec <- c("A", "B", "C")
num_vec <- c(4.5, 3.9, 7.2)
logic_vec <- c(TRUE, TRUE, FALSE)
df1 <- data.frame(charac_vec, num_vec, logic_vec)
df1

# Exercício 3 - Considere o vetor abaixo.
# Crie um loop que verifique se há números maiores que 10 e imprima o número
# e uma mensagem no console.

# Criando um Vetor
vec1 <- c(12, 3, 4, 19, 34)
vec1

for (i in 1:length(vec1)){
  if (vec1[i] > 10) {
    print(vec1[i])
    print('Este elemento do vetor é maior que 10')
  } else  {
    print(vec1[i])
    print('Este elemento do vetor é menor que 10')
  }
}


# Exercício 4 - Conisdere a lista abaixo. Crie um loop que imprima no console
# cada elemento da lista
lst2 <- list(2, 3, 5, 7, 11, 13)
lst2

for (i in 1:length(lst2)) {
  print(lst2[[i]]) # possível com a utilização de colchetes duplos [[]]
}


# Exercício 5 - Considere as duas matrizes abaixo.
# Faça uma multiplicação element-wise e multiplicação normal entre as matrizes
# https://www.mathwarehouse.com/algebra/matrix/multiply-matrix.php
mat1 <- matrix(c(1:50), nrow = 5, ncol = 5, byrow = T)
mat1
mat2 <- t(mat1) # matriz transposta
mat2
```

```
# Multiplicação element-wise - em nivel de elemento (muito usado em ML)
mat3 <- mat1 * mat2
mat3

# Multiplicação de matrizes - maneira tradicional (linhas x colunas)
# (multiplica primeira linha da mat1 com a primeira coluna de mat2
# Item [1,1] ==> (1 x 1) + (2 x 2) + (3 x 3) x (4 x 4) x (5 x 5) = 55
# Item [2,1] ==> (6 x 1) + (7 x 2) + (8 x 3) x (9 x 4) x (10 x 5) = 130
# Item [3,1] ==> (11 x 1) + (12) x 2) + (13 x 3) x (14 x 4) x (15 x 5) = 205
mat4 <- mat1 %*% mat2
mat4

# Exercício 6 - Crie um vetor, matriz, lista e dataframe e faça a nomeação
# de cada um dos objetos
vec1 <- c(12, 3, 4, 19, 34)
names(vec1) <- c('Col1', 'Col2', 'Col3', 'Col4', 'Col5') # names() - nomear vetor
vec1

mat1 <- matrix(c(1:50), nrow = 5, ncol = 5, byrow = T)
dimnames(mat1) = (list( c("Obs1", "Obs2", "Obs3", "Obs4", "Obs5"), c("Var1", "Var2", "Var3", "Var4", "Var5"))) #
dnames() - nomear matrizes
mat1

lst1 <- list(2, 3, c(1, 2, 3))
names(lst1) <- c('dim1', 'dim2', 'dim3') # names() - nomear dimensões de listas
lst1

df1 <- data.frame(c("A", "B", "C"), c(4.5, 3.9, 7.2), c(TRUE, TRUE, FALSE))
colnames(df1) <- c('Caracteres', 'Float', 'Logico') # colnames() - nomear colunas do data frame
rownames(df1) <- c("Obs1", "Obs2", "Obs3") # rownames() - nomear linhas do data frame
df1

# Exercício 7 - Considere a matriz abaixo. Atribua valores NA de forma aletória
# para 50 elementos da matriz
# Dica: use a função sample()
mat2 <- matrix(1:90, 10)
mat2
?sample # gera amostras aleatórias
mat2[sample(1:50, 10)] = NA # sample(numeros entre 1 e 50, pegue 10 valores aleatórios) = 10 índices aleatórios
mat2


# Exercício 8 - Para a matriz abaixo, calcule a soma por linha e por coluna
mat1 <- matrix(c(1:50), nrow = 5, ncol = 5, byrow = T)
mat1
rowSums(mat1)
colSums(mat1)


# Exercício 9 - Para o vetor abaixo, ordene os valores em ordem crescente
a <- c(100, 10, 10000, 1000)
a
order(a) # order() ordena e mostra os indices
a[order(a)] # mostrando o vetor ordenado com a funcao order()


# # Exercício 10 - Imprima no console todos os elementos da matriz abaixo
# que forem maiores que 15
mat1 <- matrix(c(1:50), nrow = 5, ncol = 5, byrow = T)
mat1

for (i in mat1){
  if(i > 15){
```

```
    print(i)
  }
}
# Gráficos em R - Base Plotting System

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap04")
getwd()

# Lista de pacotes base carregados
search()


# Demo - demonstra como utilizar as funções de um determinado pacote (não funciona para tudo, mas é muito útil)
demo("graphics")


# Plot Básico
x = 5:7
y = 8:10
plot(x,y)
?plot

altura <- c(145, 167, 176, 123, 150)
largura <- c(51, 63, 64, 40, 55)

plot(altura, largura)


# Plotando um Dataframe
?lynx # lynk é um dos dataframes que vêm junto da linguagem R
plot(lynx)
plot(lynx, ylab = "Plots com Dataframes", xlab = "") # xlab ou ylab - label do eixo x ou y
plot(lynx, ylab = "Plots com Dataframes", xlab = "Observações")
plot(lynx, main = "Plots com Dataframes", col = 'red') # main é o título do gráfico e col é a cor
plot(lynx, main = "Plots com Dataframes", col = 'red', col.main = 52, cex.main = 1.5)

# Fazendo um histograma de algum data set padrão R
library(datasets) # carregando o conjunto de datasets disponíveis em R
hist(warpbreaks$breaks) # função para gráficos do tipo histograma

# Boxplot de outro data set
airquality # data set
transform(airquality, Month = factor(Month)) # transformando a coluna Month em fator
boxplot(Ozone ~ Month, airquality, xlab = "Month", ylab = "Ozone (ppb)")


### Especificando os parâmetros

# col - cor do plot
# lty - tipo de linha
# lwd - largura de linha
# pch - símbolo no plot (tipo de ponto)
# xlab - label do eixo x
# ylab - label do eixo y
# las - como os labels dos eixos são orientados
# bg - background color
# mfrow - número de plots por linha
```

```
# mfcool - número de plots por coluna

### Funções Básicas de Plot

# plot() - scatterplots
# lines() -  adiciona linhas ao gráfico
# points() - adiciona pontos ao gráfico
# text() - adiciona label ao gráfico
# title() - adiciona título ao gráfico

### Parâmetros dos Gráficos
?par # permite configurar e consultar os parâmetros dos gráficos - função global e influencia todos os gráficos da sessão
par()
par('pch')
par('lty')

x = 2:4
plot(x, pch = "*")

par(mfrow = c(2,2), col.axis = "red")
plot(1:8, las = 0, xlab = "xlab", ylab = "ylab", main = "LAS = 0")
plot(1:8, las = 1, xlab = "xlab", ylab = "ylab", main = "LAS = 1")
plot(1:8, las = 2, xlab = "xlab", ylab = "ylab", main = "LAS = 2")
plot(1:8, las = 3, xlab = "xlab", ylab = "ylab", main = "LAS = 3")
legend("topright", pch = 1, col = c("blue", "red"), legend = c("Var1","Var2"))
par(mfrow = c(1,1))

### Cores disponíveis
colors()


### Salvando os gráficos

# png - faz com que o gráfico não apareça na área de plotagem, mas seja gravado num arquivo .png
png("Grafico1.png", width = 500, height = 500, res = 72)

plot(iris$Sepal.Length, iris$Petal.Length,
    col = iris$Species,
    main = "Gráfico gerado a partir do Iris")

dev.off() # dev.off() retorna geração dos gráficos para área de plotagem


# pdf
pdf("Grafico2.pdf")

plot(iris$Sepal.Length, iris$Petal.Length,
    col = iris$Species,
    main = "Gráfico gerado a partir do Iris")

dev.off()


### Estendendo as funções do plot
install.packages('plotrix')
library(plotrix) # plotrix traz mais funções para o plot do pacote basico
?plotrix

par(mfrow = c(1,1), col.axis = "red") # mfrow define área de plotagem
plot(1:6, las = 3, xlab = "lty 1:6", ylab = "", main = "Mais opções ao plot")
ablineclip(v=1, lty=1, col="sienna2", lwd=2)
ablineclip(v=2, lty=1, col="sienna2", lwd=2)
ablineclip(v=3, lty=1, col="sienna2", lwd=2)
```

```
ablineclip(v=4, lty=1, col="sienna2", lwd=2)
ablineclip(v=5, lty=1, col="sienna2", lwd=2)
ablineclip(v=6, lty=1, col="sienna2", lwd=2)
ablineclip(v=7, lty=1, col="sienna2", lwd=2)


plot(lynx)
plot(lynx, type="p", main="Type p")
plot(lynx, type="l", main="Type l")
plot(lynx, type="b", main="Type b")
plot(lynx, type="o", main="Type o")
plot(lynx, type="h", main="Type h")
plot(lynx, type="s", main="Type s")
plot(lynx, type="n", main="Type n")


### Dois plots juntos
par(mar=c(4,3,3,3), col.axis="black")

# plot1:
plot(cars$speed, type="s", col="red", bty="n", xlab="Cars ID", ylab="")
text(8, 14, "Velocidade", cex=0.85, col="red")
par(new=T)

#plot2:
plot(cars$dist, type="s", bty="n", ann=F, axes=F, col="darkblue")
axis(side=4)
text(37, 18, "Distância", cex=0.85, col="darkblue")

title(main="Velocidade x Distância")


### Plots a partir de datasets
df <- read.csv('pibpercap.csv', stringsAsFactors = F) # carregando arquivo csv
df_1982 <- subset(df, ano == 1982)
plot(expectativa ~ pibpercap, data = df_1982, log = "x")
View(df)

# Nomes paar as colunas
mycol <- c(Asia = "tomato", Europe = "chocolate4", Africa = "dodgerblue2",
         Americas = "darkgoldenrod1", Oceania = "green4")

# Plot
plot(expectativa ~ pibpercap, data = df_1982, log = "x", col = mycol[continente])

# Função para a escala - gráfico bolha:
mycex <- function(var, r, f = sqrt){
  x = f(var)
  x_scaled = (x - min(x))/(max(x) - min(x))
  r[1] + x_scaled * (r[2] - r[1])
}

# Plot
plot(expectativa ~ pibpercap, data = df_1982, log = "x",
     col = mycol[continente],
     cex = mycex(pop, r = c(0.2, 10))
)

# Scatterplots

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding
```

```
# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap04")
getwd()

# Define os dados
set.seed(67) # usado para reproduzir um mesmo trabalho (usando criações randomicas)
x = rnorm(10,5,7)
y = rpois(10,7)
z = rnorm(10,6,7)
t = rpois(10,9)


### Cria o Plot
plot(x, y,
     col = 123,
     pch = 10,
     main = "Multi Scatterplot",
     col.main = "red",
     cex.main = 1.5,
     xlab = "Variável Independente",
     ylab = "Variável Dependente")

### Adiciona outros dados no plot - Multi Scatterplot:
points(z, t, col = "blue", pch = 4)

# Adiciona outros dados no plot
points(y, t, col = 777, pch = 9)

### Cria legenda
legend(-6,5.9,
       legend = c("Nível 1", "Nível 2", "Nível 3"),
       col = c(123, "blue", 777),
       pch = c(10,4,9),
       cex = 0.65,
       bty = "n")

# Boxplots

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap04")
getwd()

?boxplot # pacote graphics
?sleep # dataset

# Permite utilizar as colunas sem especificar o nome do dataset
attach(sleep)
View(sleep)

# Construção do boxplot
sleepboxplot = boxplot(data = sleep,
                       extra ~ group, # relação entre as 2 variáveis extra e group
                       main = "Duração do Sono",
                       col.main = "red",
                       ylab = "Horas",
                       xlab = "Droga")
```

```
# Cálculo da média
medias = by(extra, group, mean) # by substitui o loop, aplicando a função mean (media) para as 2 variáveis

# Adiciona a média ao gráfico
points(medias, col = "red")


# Boxplot horizontal
horizontalboxplot = boxplot(data = sleep, extra ~ group,
                            ylab = "", xlab = "", horizontal = T)

horizontalboxplot = boxplot(data = sleep, extra ~ group,
                            ylab = "", xlab = "", horizontal = T,
                            col = c("blue", "red") )

# Histogramas

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap04")
getwd()

# Definindo os dados
?cars # dataset
View(cars) # visualiza o conteúdo
dados = cars$speed # extraindo a coluna

# Construindo um histograma
?hist
hist(dados)


# Conforme consta no help, o parâmetro breaks pode ser um dos itens abaixo:
# Um vetor para os pontos de quebra entre as células do histograma
# Uma função para calcular o vetor de breakpoints
# Um único número que representa o número de células para o histograma
# Uma cadeia de caracteres que nomeia um algoritmo para calcular o número de células
# Uma função para calcular o número de células.
hist(dados, breaks = 10, main = "Histograma das Velocidades") # definindo as escalas
hist(dados, labels = T, breaks = c(0,5,10,20,30), main = "Histograma das Velocidades")
hist(dados, labels = T, breaks = 10, main = "Histograma das Velocidades")
hist(dados, labels = T, ylim = c(0,10), breaks = 10, main = "Histograma das Velocidades")


# Adicionando linhas ao histograma
grafico <- hist(dados, breaks = 10, main = "Histograma das Velocidades")

xaxis = seq(min(dados), max(dados), length = 10) # sequencia de valores minimos e máximos
yaxis = dnorm(xaxis, mean = mean(dados), sd = sd(dados)) # distribuição normal, com média e desvio padrão
yaxis = yaxis*diff(grafico$mids)*length(dados) # calcula dif entre eixos x e y

lines(xaxis, yaxis, col = "red")
# Bar Plots

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
```

```
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap04")
getwd()

?barplot

# Preparando os dados - número de casamentos em uma igreja de SP
dados <- matrix(c(652,1537,598,242,36,46,38,21,218,327,106,67), nrow = 3, byrow = T)
dados

# Nomeando linhas e colunas na matriz
colnames(dados) <- c("0","1-150","151-300",">300")
rownames(dados) <- c("Jovem","Adulto","Idoso")
dados

# Construindo o Barplot
barplot(dados, beside = T) # uma barra para cada faixa etária, lado a lado
barplot(dados) # stacked bar plot (gráfico de dados empilhado)

# Construindo o plot - Stacked Bar Plot
# As 3 faixas de idade são representadas na mesma coluna para as diferentes quantidades
barplot(dados, col = c("steelblue1", "tan3", "seagreen3"))

# Crie uma legenda para o gráfico anterior
colors() # lista de cores
?legend
legend("topright",
       pch = 1, # bolinha
       col = c("steelblue1", "tan3", "seagreen3"), # vetor de cores
       legend = c("Jovem","Adulto","Idoso")) # vetor de legendas

# Agora temos uma coluna para cada faixa etária, mas na mesma faixa de quantidade
barplot(dados, beside = T, col = c("steelblue1", "tan3", "seagreen3"))
legend("topright", pch = 1, col = c("steelblue1", "tan3", "seagreen3"), legend = c("Jovem","Adulto","Idoso"))

# Com a Transposta da matriz, invertemos as posições entre faixa etária e faixa de quantidade
t(dados)
barplot(t(dados), beside = T, col = c("steelblue1", "tan3", "seagreen3", "peachpuff1"))
legend("topright", pch = 1, col = c("steelblue1", "tan3", "seagreen3", "peachpuff1"), legend = c("0","1-150","151-
300",">300"))

# Gráficos de Pizza

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap04")
getwd()

?pie

# Criando as fatias
fatias = c(40, 20, 40)

# Nomeando os labels
paises = c("Brasil", "Argentina", "Chile")

# Unindo paises e fatias
paises = paste(paises, fatias)
```

```
# Incluindo mais detalhes no label - concatenando strings
paises = paste(paises, "%", sep = "")

colors() # opcoes de cores

# Construindo um gráfico
pie(fatias,
    labels = paises,
    col = c("darksalmon", "gainsboro", "lemonchiffon4"),
    main ="Dsitribuição de Vendas")


# Trabalhando com dataframes
?iris
attach(iris)
Values = table(Species) # faz distribuição de frequência
labels = paste(names(Values))
pie(Values, labels = labels, main = "Distribuição de Espécies")


### 3D
install.packages("plotrix")
library(plotrix)

pie3D(fatias,
      labels = paises,
      explode = 0.05, # distancia entre as fatias
      col = c("steelblue1", "tomato2", "tan3"),
      main = "Dsitribuição de Vendas")

# ggplot2

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap04")
getwd()

# Um sistema gráfico completo, alternativo ao sistema básico de gráficos do R.
# Oferece mais opções de modificação, legendas prontas e formatação mais sólida.

# https://cran.r-project.org/web/packages/ggplot2/ggplot2.pdf
# https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

# Instalando e carregando o pacote
install.packages("ggplot2")
library(ggplot2)

# Plotando um gráfico básico com qplot()
data(tips, package = 'reshape2')
View(tips)
qplot(total_bill, tip, data = tips, geom = "point")


### Camada 1
camada1 <- geom_point(
  mapping = aes(x = total_bill, y = tip, color = sex), # cor automatica por sex
  data = tips,
  size = 3
```

```
)
ggplot() + camada1

?aes # permite mapeamento de estética
??aes # ?? permite buscar exemplos de utilização

### Contruindo um modelo de regressão, para a camada 2
modelo_base <- lm(tip ~ total_bill, data = tips) # lm - função linear model
modelo_fit <- data.frame(
  total_bill = tips$total_bill,
  predict(modelo_base, interval = "confidence")
)

head(modelo_fit)

# Camada 2
camada2 <- geom_line( # formato de linha (linha de regressão)
  mapping = aes(x = total_bill, y = fit),
  data = modelo_fit,
  color = "darkred"
)
ggplot() + camada1 + camada2


### Camada 3 - limite de confiança (margem de erro)
camada3 <- geom_ribbon(
  mapping = aes(x = total_bill, ymin = lwr, ymax = upr),
  data = modelo_fit,
  alpha = 0.3
)
ggplot() + camada1 + camada2 + camada3


### Versão final otimizada - sem a necessidade de fazer por camada
ggplot(tips,                          # conjunto de dados
       aes(x = total_bill, y = tip)) + # variaveis
  geom_point(aes(color = sex)) +       # pontos coloridos por sexo
  geom_smooth(method = 'lm')           # faixa com modelo linear


# Gravando o gráfico em um objeto
myplot <- ggplot(tips, aes(x = total_bill, y = tip)) +
  geom_point(aes(color = sex)) +
  geom_smooth(method = 'lm')

class(myplot)
print(myplot)


### ScatterPlot com linha de regressão

# Dados
data = data.frame(cond = rep(c("Obs 1", "Obs 2"),
                             each = 10), var1 = 1:100 +
                             rnorm(100,sd = 9), var2 = 1:100 +
                             rnorm(100,sd = 16))


# Plot
ggplot(data, aes(x = var1, y = var2)) +
  geom_point(shape = 1) +
  geom_smooth(method = lm , color = "red", se = FALSE)
```

```
?lm

### Bar Plot

# Dados
data = data.frame(grupo = c("A ","B ","C ","D ") ,
                  valor = c(33,62,56,67) ,
                  num_obs = c(100,500,459,342))

# Gerando a massa de dados
data$right = cumsum(data$num_obs) + 30 * c(0:(nrow(data)-1))
data$left = data$right - data$num_obs

# Plot
ggplot(data, aes(ymin = 0)) +
   geom_rect(aes(xmin = left, xmax = right,
                 ymax = valor, colour = grupo, fill = grupo)) +
                 xlab("Número de Observações") + ylab("Valor")



### Usando mtcars

head(mtcars)
ggplot(data = mtcars, aes(x = disp, y = mpg)) + geom_point()


# Outro aspecto que pode ser mapeado nesse gráfico é a cor dos pontos
ggplot(data = mtcars,
      aes(x = disp, y = mpg,
          colour = as.factor(am))) + geom_point()


# No entanto, tambem podemos mapear uma variável contínua à cor dos pontos:
View(mtcars)
ggplot(mtcars, aes(x = disp, y = mpg, colour = cyl)) + geom_point()


# Também podemos mapear o tamanho dos pontos à uma variável de interesse:
# A legenda é inserida no gráfico automaticamente
ggplot(mtcars, aes(x = disp, y = mpg, colour = cyl, size = wt)) + geom_point()


### Os geoms definem qual forma geométrica será utilizada para a visualização dos dados no gráfico.
ggplot(mtcars, aes(x = as.factor(cyl), y = mpg)) + geom_boxplot()


# Histogramas
ggplot(mtcars, aes(x = mpg), binwidth = 30) + geom_histogram()


# Gráfico de Barras
ggplot(mtcars, aes(x = as.factor(cyl))) + geom_bar()


### Personalizando os gráficos
colors()

ggplot(mtcars, aes(x = as.factor(cyl), y = mpg,
                   colour = as.factor(cyl))) + geom_boxplot()

ggplot(mtcars, aes(x = as.factor(cyl), y = mpg,
                   fill = as.factor(cyl))) + geom_boxplot()
```

```
ggplot(mtcars,
       aes(x = as.factor(cyl), y = mpg)) +
  geom_boxplot(color = "blue", fill = "seagreen4")


# Podemos alterar os eixos
ggplot(mtcars, aes(x = mpg)) +
  geom_histogram() +
  xlab("Milhas por galão") + ylab("Frequência")


# Legendas
ggplot(mtcars, aes(x = as.factor(cyl), fill = as.factor(cyl))) +
  geom_bar() +
  labs(fill = "cyl")

# Trocando a posição da legenda
ggplot(mtcars, aes(x = as.factor(cyl), fill = as.factor(cyl))) +
  geom_bar() +
  labs(fill = "cyl") +
  theme(legend.position = "top")


# Sem legenda
ggplot(mtcars, aes(x = as.factor(cyl), fill = as.factor(cyl))) +
  geom_bar() +
  guides(fill = FALSE)


### Facets - gráficos do mesmo tipo, com valores diferentes
ggplot(mtcars, aes(x = mpg, y = disp, colour = as.factor(cyl))) +
  geom_point() +
  facet_grid(am~.) # variavel am, associada (~) a todas outras variaveis(.)

ggplot(mtcars, aes(x = mpg, y = disp, colour = as.factor(cyl))) +
  geom_point() +
  facet_grid(.~am) # mudando a orientação linha x coluna


### Plots diferentes juntos (diferente de Facet) - tipos de gráficos diferentes
install.packages("gridExtra")
library(gridExtra)
library(ggplot2)

# Dataset diamonds
data(diamonds)


# Histograma como plot1
plot1 <- qplot(price, data = diamonds, binwidth = 1000)

# ScatterPlot como plot2
plot2 <- qplot(carat, price, data = diamonds, colour = cut)

# Combina os 2 plots na mesma área
grid.arrange(plot1, plot2, ncol = 1)


### Gráficos de Densidade

ggplot(data = diamonds, aes(x = price, group = cut, fill = cut)) +
    geom_density(adjust = 1.5)
```

```
ggplot(data = diamonds, aes(x = price, group = cut, fill = cut)) +
   geom_density(adjust = 1.5 , alpha = 0.2)

ggplot(data = diamonds,aes(x = price, group = cut, fill = cut)) +
   geom_density(adjust = 1.5, position = "fill")


### Facets com reshape - ajuste nos gráficos
install.packages("reshape2")
install.packages("plotly")
library(reshape2)
library(plotly)

sp <- ggplot(tips, aes(x = total_bill, y = tip/total_bill)) + geom_point(shape = 1)
sp + facet_grid(sex ~ .)
ggplotly()
sp + facet_grid(. ~ sex)
ggplotly()
sp + facet_wrap( ~ day, ncol = 2)
ggplotly() # transforma o gráfico num interativo


ggplot(mpg, aes(displ, hwy)) + geom_point() + facet_wrap(~manufacturer)
ggplotly()
# Lattice

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap04")
getwd()

# O pacote Lattice é um sistema de visualização de dados
# de alto nível poderoso e elegante, com ênfase em dados
# multivariados.

# Na criação de gráficos, condições e agrupamentos são 2 conceitos
# importantes, que nos permitem compreender mais facilmente
# os dados que temos em mãos. O conceito por trás do Lattice
# é agrupar os dados e criar visualizaçãoes de forma que fique
# mais fácil a busca por padrões.

# Instala e carrega o pacote
install.packages('lattice')
library(lattice)

### ScatterPlot com Lattice
View(iris)
xyplot(data = iris, groups = Species, Sepal.Length ~ Petal.Length)

### BarPlots com dataset Titanic
?Titanic

barchart(Class ~ Freq | Sex + Age,
        data = as.data.frame(Titanic),
        groups = Survived,
        stack = T,
        layout = c(4, 1),
        auto.key = list(title = "Dados Titanic", columns = 2))
```

```
barchart(Class ~ Freq | Sex + Age,
         data = as.data.frame(Titanic),
         groups = Survived,
         stack = T,
         layout = c(4, 1),
         auto.key = list(title = "Dados Titanic", columns = 2),
         scales = list(x = "free"))


### ScatterPlots

# Contagem de elementos
PetalLength <- equal.count(iris$Petal.Length, 4)
PetalLength

# Plot
xyplot(Sepal.Length~Sepal.Width | PetalLength, # relacao de 2 variáveis com outra variável
       data = iris)

# com grid no gráfico
xyplot(Sepal.Length~Sepal.Width | PetalLength,
       data = iris,
       panel = function(...) {
         panel.grid(h = -1, v = -1, col.line = "skyblue")
         panel.xyplot(...)})

# com modelo de regressão
xyplot(Sepal.Length~Sepal.Width | PetalLength, data = iris,
       panel = function(x,y,...) {
         panel.xyplot(x,y,...)
         mylm <- lm(y~x) # modelo de regressão
         panel.abline(mylm)})

### Histograma
histogram(~Sepal.Length | Species, xlab = "",
          data = iris, layout=c(3,1), type = "density",
          main = "Histograma Lattice", sub = "Iris Dataset, Sepal Length")


### Distribuição dos dados
qqmath(~ Sepal.Length | Species, data = iris, distribution = qunif)


### Boxplot
bwplot(Species~Sepal.Length, data = iris)


### ViolinPlot
bwplot(Species~Sepal.Length, data = iris,
       panel = panel.violin)

# Mapas

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap04")
getwd()
```

```
# Instala os pacotes
install.packages(c("ggplot2", "maps", "mapdata"))

# Carrega os pacotes
library(ggplot2)
library(maps)
library(mapdata)

# Função para buscar as coordenadas dos países
?map_data
mapa <- map_data("world") # buscando dados do mundo (país, longetude e latitude)

# Visualizando o dataframe
dim(mapa)
View(mapa)

# Gerando o Mapa
ggplot() +
  geom_polygon(data = mapa,
                   aes(x=long,
                       y = lat,
                       group = group)) +
  coord_fixed(1.3)

ggplot() +
  geom_polygon(data = mapa,
               aes(x=long,
                   y = lat,
                   group = group),
               fill = NA,          # sem preenchimento
               color = "blue") +   # contorno em azul
  coord_fixed(1.3)

gg1 <- ggplot() +
  geom_polygon(data = mapa, aes(x=long, y = lat, group = group), fill = "seagreen1", color = "blue") +
  coord_fixed(1.3)
gg1

# Marcando alguns pontos no mapa
# Podemos usar um shapefile (para geração de long e lat, quando indisponível no arquivo)
labs <- data.frame(
  long = c(69.24140, -2.8456051),
  lat = c(-78.38995, 22.44512),
  names = c("Ponto1", "Ponto2"),
  stringsAsFactors = FALSE
)

# Pontos no mapa
gg1 +
  geom_point(data = labs, aes(x = long, y = lat), color = "black", size = 2) +
  geom_point(data = labs, aes(x = long, y = lat), color = "yellow", size = 2)

# Divisão por países
ggplot(data = mapa) +
  geom_polygon(aes(x = long,
                   y = lat,
                   fill = region,
                   group = group),
               color = "white") +
  coord_fixed(1.3) +
  guides(fill=FALSE)
```

```
# rMaps - forma de criar mapas em R
# http://rmaps.github.io

# Big Data na Prática 2 - Visualizações Interativas

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap04")
getwd()

# Baseado no Google Chart (visualização de dados na web)
# googlevis é um pacote que fornece interface entre R e o Google Charts
# Utiliza JavaScript e arquivos JSON. Os dados são transformados em formato JSON
# O resultado é gerado em HTML5 ou Flash
# Pode-se criar os mais variados tipos de gráficos e mapas. Inclusive Google Maps
# Você pode incorporar os gráficos criados em páginas HTML ou apps
# Verifique termos de uso, antes de utilizar

install.packages("googleVis")
library(googleVis)
?googleVis


# Criando um Datafrane
df = data.frame(País = c("BR", "CH", "AR"),
                Exportações = c(10,13,14),
                Importações = c(23,12,32))


# Gráfico de Linha
Line <- gvisLineChart(df)
plot(Line)


# Gráfico de Barras
Column <- gvisColumnChart(df)
plot(Column)


# Gráfico de Barras Horizontais
Bar <- gvisBarChart(df)
plot(Bar)


# Gráfico de Pizza
Pie <- gvisPieChart(CityPopularity)
plot(Pie)


# Gráficos Combinados
Combo <- gvisComboChart(df,
                        xvar = "País",
                        yvar = c("Exportações", "Importações"),
                        options = list(seriesType = "bars",
                                       series='{1: {type:"line"}}'))
plot(Combo)


# Scatter Chart
```

```
Scatter <- gvisScatterChart(women,
                        options=list(
                          legend="none",
                          lineWidth=2, pointSize=0,
                          title="Women", vAxis="{title:'weight (lbs)'}",
                          hAxis="{title:'height (in)'}",
                          width=300, height=300))
plot(Scatter)


# Bubble
Bubble <- gvisBubbleChart(Fruits, idvar="Fruit",
                      xvar="Sales", yvar="Expenses",
                      colorvar="Year", sizevar="Profit",
                      options=list(
                        hAxis='{minValue:75, maxValue:125}'))
plot(Bubble)


# Customizando
M <- matrix(nrow=6,ncol=6)
M[col(M)==row(M)] <- 1:6
dat <- data.frame(X=1:6, M)
SC <- gvisScatterChart(dat,
                     options=list(
                       title="Customizing points",
                       legend="right",
                       pointSize=30,
                       series="{
                       0: { pointShape: 'circle' },
                       1: { pointShape: 'triangle' },
                       2: { pointShape: 'square' },
                       3: { pointShape: 'diamond' },
                       4: { pointShape: 'star' },
                       5: { pointShape: 'polygon' }
                       }"))
plot(SC)


# Gauge
Gauge <- gvisGauge(CityPopularity,
                 options=list(min=0, max=800,
                              greenFrom=500,
                              greenTo=800,
                              yellowFrom=300,
                              yellowTo=500,
                              redFrom=0,
                              redTo=300,
                              width=400,
                              height=300))
plot(Gauge)


# Mapas
Intensity <- gvisIntensityMap(df)
plot(Intensity)


# Geo Chart
Geo=gvisGeoChart(Exports, locationvar="Country",
                colorvar="Profit",
                options=list(projection="kavrayskiy-vii"))
plot(Geo)
```

```
# Google Maps
AndrewMap <- gvisMap(Andrew, "LatLong" , "Tip",
                  options=list(showTip=TRUE,
                               showLine=TRUE,
                               enableScrollWheel=TRUE,
                               mapType='terrain',
                               useMapTypeControl=TRUE))
plot(AndrewMap)


# Dados em Gráfico. Nível de analfabetismo nos EUA
require(datasets)
states <- data.frame(state.name, state.x77)
GeoStates <- gvisGeoChart(states, "state.name", "Illiteracy",
                       options=list(region="US",
                                    displayMode="regions",
                                    resolution="provinces",
                                    width=600, height=400))
plot(GeoStates)
```

Capítulo 05:

```
# Solução Lista de Exercícios - Capítulo 4

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap05")
getwd()

# Exercicio 1 - Crie uma função que receba os dois vetores abaixo como parâmetro,
# converta-os em um dataframe e imprima no console
vec1 <- (10:13)
vec2 <- c("a", "b", "c", "d")

myfunc <- function(x, y){
  df = data.frame(cbind(x, y))
  print(df)
}

myfunc(vec1, vec2)


# Exercicio 2 - Crie uma matriz com 4 linhas e 4 colunas preenchida com
# números inteiros e calcule a média de cada linha
# DICA: sempre precisar fazer loops, tente utilizar a família apply.
mat <- matrix(c(1:16), nrow = 4, ncol = 4)
mat
apply(mat, 1, mean) # (matriz, por linha, media)


# Exercicio 3 - Considere o dataframe abaixo.
# Calcule a média por disciplina e depois calcule a média de apenas uma disciplina
escola <- data.frame(Aluno = c('Alan', 'Alice', 'Alana', 'Aline', 'Alex', 'Ajay'),
                     Matematica = c(90, 80, 85, 87, 56, 79),
                     Geografia = c(100, 78, 86, 90, 98, 67),
                     Quimica = c(76, 56, 89, 90, 100, 87))
escola
apply(escola[, c(2, 3, 4)], 2, mean) # (df[todas as linhas, colunas 2 a 4], por coluna, media)

# Obs: Se você tentar calcular a média de apenas uma disciplina, assim,
# vai receber uma mensagem de erro:
apply(escola$Matemática,2,mean)

# Isso acontece porque o interpretador do R tenta usar um vetor de qualquer dimensão,
# enquanto apply espera que o objeto tenha algumas dimensões.
# Você pode evitar a coerção, adicionando drop = F ao seu comando, ou seja:
apply(escola[, c(2), drop=F], 2, mean)


# Exercicio 4 - Crie uma lista com 3 elementos, todos numéricos
# e calcule a soma de todos os elementos da lista
lst <- list(a = 1:10, b = 1:5)
lst
do.call(sum, lst)


# Exercicio 5 - Transforme a lista anterior em um vetor
unlist(lst)
```

```
# Exercicio 6 - Considere a string abaixo. Substitua a palavra "textos" por "frases"
str <- c("Expressoes", "regulares", "em linguagem R",
         "permitem a busca de padroes", "e exploracao de textos",
         "podemos buscar padroes em digitos",
         "como por exemplo",
         "10992451280")

gsub("textos", "frases", str)


# Exercicio 7 - Usando o dataset mtcars, crie um grafico com ggplot do tipo
# scatter plot. Use as colunas disp e mpg nos eixos x e y respectivamente
library(ggplot2)
ggplot(data = mtcars, aes(x = disp, y = mpg)) + geom_point()


# Exercicio 8 - Considere a matriz abaixo.
# Crie um bar plot que represente os dados em barras individuais.
mat1 <- matrix(c(652,1537,598,242,36,46,38,21,218,327,106,67), nrow = 3, byrow = T)
mat1
barplot(mat1, beside = T) # barras individuais
barplot(mat1) # barras empilhadas stacked bar plot


# Exercício 9 - Qual o erro do código abaixo?
# Troubleshooting (analisar erros)
data(diamonds)
View(diamonds)
ggplot(data = diamonds, aes(x = price, group = fill, fill = cut)) +
  geom_density(adjust = 1.5)

# codigo correto - substituir valor de group
ggplot(data = diamonds,
       aes(x = price,
           group = cut,
           fill = cut)) +
  geom_density(adjust = 1.5)


# Exercício 10 - Qual o erro do código abaixo?
ggplot(mtcars,
       aes(x = as.factor(cyl),
           fill = as.factor(cyl))) +
  geom_barplot() +
  labs(fill = "cyl")

# codigo correto - substituir função por geom_bar
ggplot(mtcars,
       aes(x = as.factor(cyl),
           fill = as.factor(cyl))) +
  geom_bar() +
  labs(fill = "cyl")
# Trabalhando com Arquivos txt

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap05")
```

```
getwd()

# Usando as funções base do R (pacote utils)
search()


### Importando arquivo com read.table()
?read.table

# dim() - dimensões
df1 <- read.table("arquivos/pedidos.txt")
df1
dim(df1) # carregou com apenas uma coluna

df1 <- read.table("arquivos/pedidos.txt", header = TRUE, sep = ',')
df1
dim(df1) # carregou com varias colunas

df1 <- read.table("arquivos/pedidos.txt", header = TRUE, sep = ',',
                  col.names = c("var1", "var2", "var3"))
df1

df1 <- read.table("arquivos/pedidos.txt", header = TRUE,
                  sep = ',',
                  col.names = c("var1", "var2", "var3"),
                  na.strings = c('Zico', 'Maradona'))
df1

str(df1) # str permite visualizar o resultado da importação

df1 <- read.table("arquivos/pedidos.txt", header = TRUE,
                  sep = ',',
                  col.names = c("var1", "var2", "var3"),
                  na.strings = c('Zico', 'Maradona'),
                  stringsAsFactors = FALSE)
df1

str(df1)



### Importando arquivo com read.csv()
df2 <- read.csv("arquivos/pedidos.txt")
df2
dim(df2)

df3 <- read.csv2("arquivos/pedidos.txt")
df3
dim(df3)

df3 <- read.csv2("arquivos/pedidos.txt", sep = ',')
df3
dim(df3)


### Importando arquivo com read.delim()
df4 <- read.delim("arquivos/pedidos.txt")
df4

df4 <- read.delim("arquivos/pedidos.txt", sep = ',')
df4
dim(df4)
```

```
### Importando / Exportando

#Gerando arquivo
write.table(mtcars, file = 'arquivos/mtcars.txt')
dir("arquivos/.") # mostra o conteúdo do diretorio

df_mtcars <- read.table("arquivos/mtcars.txt")
df_mtcars
dim(df_mtcars)

write.table(mtcars,
            file = "arquivos/mtcars2.txt",
            sep = "|",
            col.names = NA,
            qmethod = "double")

list.files("arquivos/.")

read.table("arquivos/mtcars2.txt")
df_mtcars2 <- read.table("arquivos/mtcars2.txt")
df_mtcars2

df_mtcars2 <- read.table("arquivos/mtcars2.txt", sep = '|')
df_mtcars2

df_mtcars2 <- read.table("arquivos/mtcars2.txt", sep = '|', encoding = 'UTF-8')
df_mtcars2
# Trabalhando com arquivos csv

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap05")
getwd()

### Usando o pacote readr
install.packages("readr")
library(readr)

# Abre o promt para escolher o arquivo
meu_arquivo <- read_csv(file.choose())


### Importando arquivos
df1 <- read_table("arquivos/temperaturas.txt",
                  col_names = c("DAY","MONTH","YEAR","TEMP"))

head(df1) # cabeçalho ou primeiros registros
View(df1) # formato de tabela
str(df1)  # tipos de dados

# leitura linha por linha
read_lines("arquivos/temperaturas.txt", skip = 0, n_max = -1L)

# importar o arquivo inteiro, sem nenhum parametro, perde-se a formatação
read_file("arquivos/temperaturas.txt")

### Exportando e Importando
write_csv(iris, "iris.csv")
```

```
dir()


### definindo os tipos de dados para cada uma das colunas
# col_integer():
# col_double():
# col_logical():
# col_character():
# col_factor():
# col_skip():
# col_date() (alias = “D”), col_datetime() (alias = “T”), col_time() (“t”)

df_iris <- read_csv("iris.csv",
                    col_types = list(
                      Sepal.Length = col_double(),
                      Sepal.Width = col_double(),
                      Petal.Length = col_double(),
                      Petal.Width = col_double(),
                      Species = col_factor(c("setosa", "versicolor", "virginica"))
                      )
                    )

dim(df_iris)
str(df_iris)


### Manipulando arquivos csv
df_cad <- read_csv("arquivos/cadastro.csv")
View(df_cad) # ATENÇÃO: observa-se a necessidade de subcolunas separadas por "|"
class(df_cad)

# Pacote para manipulação dos dados
install.packages("dplyr")
library(dplyr)

options(warn=-1) # caso não queira visualizar mensagens de warning

df_cad <- tbl_df(df_cad)
head(df_cad)
View(df_cad)

write.csv(df_cad, "arquivos/df_cad_bkp.csv")

### Importando vários arquivos simultaneamente
list.files()
lista_arquivos <- list.files('D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/
03_BDataAnalyticsR_MAzureML/Cap05/arquivos', full.names = TRUE)
class(lista_arquivos)
lista_arquivos # caractere com o nome completo dos arquivos do diretorio

# Faz o loop na lista de arquivos com a função read_csv
lista_arquivos2 <- lapply(lista_arquivos, read_csv)
class(lista_arquivos2) # lista de arquivos
View(lista_arquivos2)


### Parsing
# ajuste de formato de data
parse_date("01/02/15", "%m/%d/%y")
parse_date("01/02/15", "%d/%m/%y")
parse_date("01/02/34", "%y/%m/%d")

# especifica o idioma e seus formatos, que se irá trabalhar
```

```
locale("en")
locale("fr")
locale("pt")
# Manipulação de Arquivos Excel

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap05")
getwd()

# Instalar o Java - JDK
# https://www.oracle.com/technetwork/java/javase/downloads/index.html

# Windows
# Faça o download do JDK gratuitamente no site da Oracle
# Instale no seu computador
# Configure a variável de ambiente JAVA_HOME apontantando para o diretório de instalação do JDK
# Inclua o diretório JAVA_HOME/bin na variável de ambiente PATH

# Mac e Linux
# Faça o download do JDK gratuitamente no site da Oracle
# Instale no seu computador
# Abra um terminal e execute: sudo R CMD javareconf

# ********** Pacotes que requerem Java **********
# XLConnect
# xlsx
# rJava

# ********** Pacotes que requerem Perl **********
# É necessário instalar o interpretador da linguagem Perl e adicionar o diretório bin da instalação do Perl
# na variável de ambiente PATH
# Download: https://www.activestate.com/products/activeperl/
# gdata

# Instalando os pacotes
# Este pacote deve estar carregado para poder usar todos os pacotes que requerem Java
install.packages('rJava')
library(rJava)
# Os manipuladores de arquivos excel:
install.packages("xlsx")
install.packages("XLConnect")
install.packages("gdata")
install.packages("readxl")


# Obs: Ao carregar todos os pacotes que manipulam excel, pode gerar problema de namespace, pois alguns pacotes
# possuem o mesmo nome de funções (que são diferentes entre os pacotes). Para evitar problemas, carregue e use
# os pacotes de forma individual (não carregue todos os pacotes de uma única vez).

# Pacote readxl
library(readxl)

# Lista as worksheet no arquivo Excel
excel_sheets("UrbanPop.xlsx")

# Lendo a planilha do Excel
read_excel("UrbanPop.xlsx")
head(read_excel("UrbanPop.xlsx"))
```

```
read_excel("UrbanPop.xlsx", sheet = "Period2") # informando a planilha pelo nome
read_excel("UrbanPop.xlsx", sheet = 3) # informando a planilha pelo índice
read_excel("UrbanPop.xlsx", sheet = 4)

# Importando uma worksheet para um dataframe
df <- read_excel("UrbanPop.xlsx", sheet = 3)
head(df)

# Importando todas as worksheets
df_todas <- lapply(excel_sheets("UrbanPop.xlsx"), # carregando o workbook - arquivo inteiro excel
                   read_excel, # lendo cada uma das planilhas do arquivo
                   path = 'UrbanPop.xlsx') # nome do arquivo
df_todas
class(df_todas) # lista


# Pacote XLConnect
detach(package:readxl) #descarrega o pacote anterior - evita conflitos de funções com mesmo nome em pacotes
diferentes
library(XLConnect)

# Namespace - evita conflitos de funções com mesmo nome em pacotes diferentes (use quando não descarregar o
pacote anterior)
arq1 = XLConnect::loadWorkbook("UrbanPop.xlsx") # carregando o arquivo ou o workbook (ponteiro para o arquivo)
df1 = readWorksheet(arq1, sheet = "Period1", header = TRUE) # carregando a planilha ou o worksheet
df1
class(df1)


# Pacote xlsx
detach(package:XLConnect)
library(xlsx)
?xlsx # help da função

?read.xlsx
df2 <- read.xlsx("UrbanPop.xlsx", sheetIndex = 1)
df2



# Pacote gdata
detach(package:xlsx)
library(gdata)

arq1 <- xls2csv("planilha1.xlsx",
                sheet = 1,
                na.strings = "EMPTY")
arq1
read.csv(arq1)

# Big Data na Prática 3 - Mineração de Regra de Associação

# Esse é um material de bônus.
# Conteúdo sobre ese tema pode ser encontrado nos cursos:

# Business Analytics
# Análise em Grafos Para Big Data
# Data Mining e Modelagem Preditiva

# Embora tenha mais de 20 anos, o Market Basket Analysis (MBA) (ou Mineração de Regras de Associação)
# ainda pode ser uma técnica muito útil para obter insights em grandes conjuntos de dados transacionais.

# O exemplo clássico são dados transacionais em um supermercado. Para cada cliente, sabemos quais são os produtos
```

```
# individuais (itens) que ele colocou na cesta e comprou. Outros casos de uso para o MBA podem ser dados de clique da web,
# arquivos de log e até questionários.

# Com a análise de cesta de compras, podemos identificar itens que são frequentemente comprados juntos.
# Normalmente, os resultados de um MBA são apresentados sob a forma de regras.
# As regras podem ser tão simples quanto {A ==> B}, quando um cliente compra o item A então é (muito) provável
# que o cliente compre o item B. Regras mais complexas também são possíveis {A, B ==> D, F}, quando um cliente
# compra os itens A e B, é provável que ele compre os itens D e F.

# Neste Big Data na Pratica, vamos buscar a associação entre os clubes de futebol da Europa e responder a pergunta:

# Quais clubes mais realizam transações de compra e venda de jogadores, entre si?

# Usaremos um dataset oferecido pelo Kaggle: https://www.kaggle.com/hugomathien/soccer

# O dataset contêm cerca de 25.000 partidas de onze ligas de futebol europeias a partir da temporada 2008/2009
# até a temporada 2015/2016.

# Depois de realizar o trabalho de Data Wrangling (manipulação dos dados), vamos gerar um conjunto de dados transacionais adequado para análise
# de cesta de compras.

# Portanto, não temos clientes, mas jogadores de futebol, e não temos produtos, mas clubes de futebol.

# No total, o conjunto de dados transacionais de futebol contém cerca de 18.000 registros.
# Obviamente, esses registros não incluem apenas as transferências multimilionárias cobertas pela mídia,
# mas também todas as transferências de jogadores que ninguém nunca ouviu falar.

# Como vamos aplicar o MBA?
# Em R você pode usar o pacote arules para mineração de regras de associação / MBA.
# Alternativamente, quando a ordem das transações é importante, você deve usar o pacote arulesSequences.
# Depois de executar o algoritmo, obteremos alguns resultados interessantes.

# Por exemplo: neste conjunto de dados, a transferência mais frequente é da Fiorentina para o Gênova
# (12 transferências no total). Vamos imprimir a tabela com todos os resultados ao final do processo.

# Visualização de gráfico de rede
# Todas as regras que obtemos da mineração de regras de associação formam um gráfico de rede.
# Os clubes de futebol individuais são os nós do gráfico e cada regra "de ==> para" é uma aresta (edge) do gráfico de rede.

# Em R, os gráficos de rede podem ser visualizados bem por meio do pacote visNetwork.

# Vamos começar?

# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap05")
getwd()

# Pacotes
install.packages("RSQLite") # manipular banco de dados em formato sqlite
install.packages("dplyr") # manipulação de dados
install.packages("tidyr") # manipulação de dados
install.packages("arules") # algoritmo de machine learning
install.packages("arulesSequences") # algoritmo de machine learning
install.packages("readr") # leitura de arquivo txt
install.packages("visNetwork") # visualização de dados
install.packages("igraph") # visualização de dados
install.packages("lubridate") # manipulação de datas
install.packages("DT") # criação de tabela de resumo

library(RSQLite)
```

```
library(dplyr)
library(tidyr)
library(arules)
library(arulesSequences)
library(readr)
library(stringr)
library(visNetwork)
library(igraph)
library(lubridate)
library(DT)


# Os dados estão disponibilizados em um banco de dados SQLITE
# que pode ser baixado do kaggle, mas está anexo a este script.

# Conectando no banco de dados
con = dbConnect(RSQLite::SQLite(), dbname="database.sqlite")


# Obtendo a lista de tabelas
alltables = dbListTables(con)
alltables


# Extraindo as tabelas em data frames
players       = dbReadTable(con, "Player")
players_stats = dbReadTable(con, "Player_Attributes")
teams         = dbReadTable(con, "Team")
league        = dbReadTable(con, "League")
Matches       = dbReadTable(con, "Match")

View(players)
View(players_stats)
View(teams)
View(league)
View(Matches)


# Substituindo espaço por underline nos nomes muito longos
teams$team_long_name = str_replace_all(teams$team_long_name, "\\s", "_")
teams$team_long_name = str_replace_all(teams$team_long_name, "\\.", "_")
teams$team_long_name = str_replace_all(teams$team_long_name, "-", "_")
View(teams)

# Agrupando as equipes por país
CountryClub = Matches %>%
  group_by(home_team_api_id,country_id) %>%
  summarise(n=n()) %>%
  left_join(league) %>%
  left_join(teams, by=c("home_team_api_id" = "team_api_id"))


# Preparando os dados para mineração das regras de associação

# Os jogadores estão em colunas separadas, mas precisamos deles empilhados em uma coluna
tmp = Matches %>%
  select(
    season,
    home_team_api_id,
    home_player_1:home_player_11
  )%>%
  gather(
    player,
```

```
    player_api_id,
    -c(season, home_team_api_id)
  ) %>%
  group_by(player_api_id, home_team_api_id ) %>%
  summarise(season = min(season))


# Unindo dados de jogador e clube
playerClubSequence = left_join(
  tmp,
  players
  ) %>%
  left_join(
    teams,
    by=c("home_team_api_id"="team_api_id")
  )

playerClubSequence = playerClubSequence %>%
  filter(
    !is.na(player_name), !is.na(team_short_name)
  )  %>%
  arrange(
    player_api_id,
    season
  )


# Adicionando um número sequencial por jogador
playerClubSequence$seqnr = ave( playerClubSequence$player_api_id, playerClubSequence$player_api_id, FUN =
seq_along)
playerClubSequence$size = 1


# Mineração de sequências com algoritmo cSPade do pacote arulesSequences

# Grava o conjunto de dados em um arquivo txt para facilitar a manipulação
# da função read_basket em arulesSequence para criar um objeto de transação
write_delim(
  playerClubSequence %>% select( c(player_api_id, seqnr, size, team_long_name)) ,
  delim ="\t", path = "player_transactions.txt", col_names = FALSE
  )


# Agora importamos as transações registradas no item anterior
playerstrxs <- read_baskets("player_transactions.txt", sep = "[ \t]+",info =  c("sequenceID","eventID","size"))
summary(playerstrxs)


# Executar mineração de sequência, por enquanto apenas com comprimento de duas sequências
?cspade

playersClubSeq <- cspade(
  playerstrxs,
  parameter = list(support = 0.00010, maxlen=2),
  control   = list(verbose = TRUE)
)

summary(playersClubSeq)

# Fazendo Data Wrangling para colocar os resultados do cspade em um organizado conjunto de dados
# que é adequado para a visNetwork. A visNetwork precisa de dois conjuntos de dados:
# um conjunto de dados com as arestas "de --> para" e um conjunto de dados com os nós exclusivos
seqResult = as(playersClubSeq, "data.frame")
```

```
seqResult = seqResult %>%
  mutate(
    sequence = as.character(sequence)
  )

seqResult = bind_cols(
  seqResult,
  as.data.frame(
    str_split_fixed(seqResult$sequence, pattern =",", 2),
    stringsAsFactors = FALSE)
  )

seqResult$from = str_extract_all(seqResult$V1,"\\w+", simplify = TRUE)[,1]
seqResult$to   = str_extract_all(seqResult$V2,"\\w+",simplify = TRUE)[,1]


seqResult$width = exp(3000*seqResult$support)
seqResult = seqResult %>% filter(V2 !="")
seqResult$title = paste(seqResult$sequence, "<br>", round(100*seqResult$support,2), "%")

seqResult$support_perc = paste(sprintf("%.4f", 100*seqResult$support), "%")


# Criando o dataframe com os nodes
nodes = unique(c(seqResult$from, seqResult$to))
nodesData = data.frame(id = unique(nodes), title = unique(nodes), label = unique(nodes), stringsAsFactors = FALSE)
%>%
  left_join(CountryClub, by = c("id"="team_long_name")) %>%
  rename(group = name)

View(nodes)

# Calcula as medidas de centralidade de betweeness
# usando o igraph, para que possamos ter tamanhos diferentes de
# nós no gráfico de rede
transferGraph = graph_from_data_frame(seqResult[,c(5,6)], directed = TRUE)

tmp = betweenness(transferGraph)
Clubs_betweenness = data.frame(id = names(tmp), value = tmp, stringsAsFactors = FALSE)
nodesData = nodesData %>%
  left_join(Clubs_betweenness) %>%
  mutate(title = paste(id, "betweeness ", round(value))) %>%
  arrange(id)


# Criando a rede interativa

# Preparando o dataframe final e removendo duplicidades
nodes = nodesData
nodes = nodes[!duplicated(nodes$id),]


# Cria a rede
?visNetwork

visNetwork(nodes, edges = seqResult, width = 900, height = 700) %>%
  visNodes(size = 10) %>%
  visLegend() %>%
  visEdges(smooth = FALSE) %>%
  visOptions(highlightNearest = TRUE, nodesIdSelection = TRUE) %>%
  visInteraction(navigationButtons = TRUE) %>%
  visEdges(arrows = 'from') %>%
  visPhysics(
```

```
    solver = "barnesHut",
    maxVelocity = 35,
    forceAtlas2Based = list(gravitationalConstant = -6000)
  )


# Cria a tabela final para suportar a análise
seqResult$Ntransctions = seqResult$support*10542
DT::datatable(
  seqResult[,c(5,6,9,10)],
  rownames = FALSE,
  options = list(
    pageLength=25)
  )
```

Capítulo 06:

```
# Solução Lista de Exercícios - Capítulo 5

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("C:/FCD/BigDataRAzure/Cap06")
getwd()


# Exercicio 1 - Encontre e faça a correção do erro na instrução abaixo:
write.table(mtcars, file = "mtcars2.txt", sep = "|", col.names = N, qmethod = "double")

## Correto:
write.table(mtcars, file = "mtcars2.txt", sep = "|", col.names = NA, qmethod = "double")



# Exercicio 2 - Encontre e faça a correção do erro na instrução abaixo:
# Troubleshooting
library(readr)
?read_csv
df_iris <- read_csv("iris.csv", col_types = matrix(
  Sepal.Length = col_double(1),
  Sepal.Width = col_double(1),
  Petal.Length = col_double(1),
  Petal.Width = col_double(1),
  Species = col_factor(c("setosa", "versicolor", "virginica"))
))

## Correto:
library(readr)
df_iris <- read_csv("iris.csv", col_types = list(
    Sepal.Length = col_double(),
    Sepal.Width = col_double(),
    Petal.Length = col_double(),
    Petal.Width = col_double(),
    Species = col_factor(c("setosa", "versicolor", "virginica"))
))


# Exercício 3 - Abaixo você encontra dois histogramas criados separadamente.
# Mas isso dificulta a comparação das distribuições. Crie um novo plot que faça a união
# de ambos histogramas em apenas uma área de plotagem.

# Dados aleatórios
dataset1=rnorm(4000 , 100 , 30)
dataset2=rnorm(4000 , 200 , 30)

# Histogramas
par(mfrow=c(1,2))
hist(dataset1, breaks=30 , xlim=c(0,300) , col=rgb(1,0,0,0.5) , xlab="Altura" , ylab="Peso" , main="" )
hist(dataset2, breaks=30 , xlim=c(0,300) , col=rgb(0,0,1,0.5) , xlab="Altura" , ylab="Peso" , main="")

hist(dataset1, breaks=30, xlim=c(0,300), col=rgb(1,0,0,0.5), xlab="height", ylab="nbr of plants", main="distribution of
height of 2 durum wheat varieties" )
hist(dataset2, breaks=30, xlim=c(0,300), col=rgb(0,0,1,0.5), add=T)
legend("topright", legend=c("Dataset 1","Dataset 2"), col=c(rgb(1,0,0,0.5),
```

```
rgb(0,0,1,0.5)), pt.cex=2, pch=15 )



# Exercício 4 - Encontre e corrija o erro no gráfico abaixo
install.packages("plotly")
library(plotly)
View(iris)

plot_ly(iris,
        x = ~Petal.Length,
        y = ~Petal.Width,
        type="scatter",
        mode = "markers" ,
        color = Species , marker=list(size=20 , opacity=0.5))

## Correto
?plot_ly
plot_ly(iris,
        x = ~Petal.Length,
        y = ~Petal.Width,
        type="scatter",
        mode = "markers" ,
        color = ~Species , marker=list( size=20 , opacity=0.5))



# Exercício 5 - Em anexo você encontra o arquivo exercicio5.png. Crie o gráfico que resulta nesta imagem.
# https://www.r-graph-gallery.com/111-interactive-3d-plot-plotly/
library(plotly)
head(volcano)
?volcano

# 3D plot :
p = plot_ly(z = volcano, type = "surface")
p


# Exercício 6 - Carregue o arquivo input.json anexo a este script e imprima o conteúdo no console
# Dica: Use o pacote rjson
library("rjson")
result <- fromJSON(file = "input.json")
print(result)
class(result)



# Exercício 7 - Converta o objeto criado ao carregar o arquivo json do exercício anterior
# em um dataframe e imprima o conteúdo no console.
library("rjson")
result <- fromJSON(file = "input.json")
class(result)
json_data_frame <- as.data.frame(result)
print(json_data_frame)
class(json_data_frame)
View(json_data_frame)


# Exercício 8 - Carregue o arquivo input.xml anexo a este script.
# Dica: Use o pacote XML
install.packages("XML")
library("XML")
```

```
library("methods")

result <- xmlParse(file = "input.xml")
print(result)
class(result)


# Exercício 9 - Converta o objeto criado no exercício anterior em um dataframe
xmldataframe <- xmlToDataFrame("input.xml")
print(xmldataframe)
class(xmldataframe)
View(xmldataframe)


# Exercício 10 - Carregue o arquivo input.csv em anexo e então responda às seguintes perguntas:

# Pergunta 1 - Quantas linhas e quantas colunas tem o objeto resultante em R?
# Pergunta 2 - Qual o maior salário?
# Pergunta 3 - Imprima no console o registro da pessoa com o maior salário.
# Pergunta 4 - Imprima no console todas as pessoas que trabalham no departamento de IT.
# Pergunta 5 - Imprima no console as pessoas do departamento de IT com salário superior a 600.


data <- read.csv("input.csv")
print(data)
View(data)

# 1
print(is.data.frame(data))
print(ncol(data))
print(nrow(data))

# 2
# Get the max salary from data frame.
sal <- max(data$salary)
print(sal)

# 3
# Get the person detail having max salary.
retval <- subset(data, salary == max(salary))
print(retval)

# 4
retval <- subset(data, dept == "IT")
print(retval)

# 5
info <- subset(data, salary > 600 & dept == "IT")
print(info)

# Introdução a Linguagem SQL


# Mostra todos os bancos de dados
show databases;


# Cria um banco de dados
CREATE DATABASE titanicDB;


# Habilita a sessão para o banco de dados
use titanicDB;
```

```
# Cria tabela
CREATE TABLE titanic (
	pclass		char(3),
	survived	char(1),
	name		varchar(100),
	sex			char(6),
	age			char(11),
	sibsp		varchar(16),
	parch		varchar(16),
	ticket		varchar(20),
	fare		varchar(100),
	cabin		varchar(16),
	embarked	varchar(12),
	boat		varchar(50),
	body		varchar(32),
	home_dest	varchar(200)
);


# Carregar os dados do arquivo csv na tabela, usando o wizard.


# Instruções select
select name, age, sex, pclass from titanic where age > 70;
select name, age from titanic where age > 70;
select pclass, survived, avg(age) as media from titanic group by pclass, survived;


# Cria outro banco de dados
CREATE DATABASE cadastroDB;


# Habilita a sessão para o banco de dados
use cadastroDB;


# Cria tabela
CREATE TABLE FUNCIONARIOS(
  ID   INT            NOT NULL,
  NOME VARCHAR (20)     NOT NULL,
  IDADE  INT            NOT NULL,
  CIDADE  CHAR (25) ,
  SALARIO   DECIMAL (18, 2),
  PRIMARY KEY (ID)
);


# Descreve a tabela
DESC FUNCIONARIOS;

# Instruções insert
INSERT INTO FUNCIONARIOS (ID,NOME,IDADE,CIDADE,SALARIO)
VALUES (1, 'Pele', 32, 'Roma', 2000.00 );

INSERT INTO FUNCIONARIOS (ID,NOME,IDADE,CIDADE,SALARIO)
VALUES (2, 'Zico', 25, 'Paris', 1500.00 );

INSERT INTO FUNCIONARIOS (ID,NOME,IDADE,CIDADE,SALARIO)
VALUES (3, 'Rivelino', 23, 'Santiago', 4000.00 );

INSERT INTO FUNCIONARIOS (ID,NOME,IDADE,CIDADE,SALARIO)
```

```
VALUES (4, 'Garrincha', 25, 'Vienna', 6500.00 );

INSERT INTO FUNCIONARIOS (ID,NOME,IDADE,CIDADE,SALARIO)
VALUES (5, 'Jair', 64, 'Vienna', 7800.00 );

# Esse registro gera erro de duplicidade
INSERT INTO FUNCIONARIOS (ID,NOME,IDADE,CIDADE,SALARIO)
VALUES (5, 'Careca', 25, 'Milao', 8900.00 );

INSERT INTO FUNCIONARIOS (ID,NOME,IDADE,CIDADE,SALARIO)
VALUES (6, 'Careca', 25, 'Milao', 8900.00 );


# Instruções select
select * from FUNCIONARIOS;

select nome, salario from FUNCIONARIOS;

SELECT ID, NOME, SALARIO
FROM FUNCIONARIOS
WHERE SALARIO > 2000;


SELECT ID, NOME, SALARIO
FROM FUNCIONARIOS
WHERE NOME = 'Zico';


SELECT ID, NOME, SALARIO
FROM FUNCIONARIOS
WHERE SALARIO > 2000 AND IDADE < 55;


# Instrução Update
UPDATE FUNCIONARIOS
SET cidade = 'Boston'
WHERE ID = 6;


# Instrução Delete
DELETE FROM FUNCIONARIOS
WHERE ID = 6;


# Instruções select
select distinct cidade from FUNCIONARIOS;

select count(*) from FUNCIONARIOS;

select cidade, sum(salario) from FUNCIONARIOS group by cidade;

select cidade, sum(salario) from FUNCIONARIOS group by cidade having sum(salario) > 2000;


# Cria índice
create index cidade_idx on FUNCIONARIOS(cidade);


# Mostra os índices
SHOW INDEX FROM funcionarios;


# Cria uma tabela a partir de outra tabela
```

```
create table funcionarios_bkp
as select * from funcionarios;

select * from funcionarios_bkp;


# Remove a tabela
drop table funcionarios_bkp;


# Remove o banco de dados
drop database titanicDB;
drop database cadastroDB;

# Carregando e Analisando Dados do MySQL com Linguagem R

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap06")
getwd()

install.packages('RMySQL') # instalando pacote para acesso ao MySQL
install.packages("ggplot2")
install.packages("dbplyr") # usado para conectar a bancos de dados relacionais
library(RMySQL)
library(ggplot2)
library(dbplyr)
library(dplyr)

# Antes de trabalha com MySQL e R, acesse o shell do MySQL (no prompt ou terminal):
# Abra o prompt > cd "diretório do MySQL" > cd bin > mysql -u root -p > digite sua senha
# Após confirmação da senha, digite o comando:
# ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'dsa1234';

# Conexão com o banco de dados
?dbConnect
con = dbConnect(MySQL(), user = "root", password = "dsa1234", dbname = "titanicDB", host = "localhost")


# Query - Salvando a query numa string:
qry <- "select pclass, survived, avg(age) as media_idade from titanic where survived = 1 group by pclass, survived;"
# Query - Executando a query:
dbGetQuery(con, qry)


# Plot
dados <- dbGetQuery(con, qry)
head(dados)
class(dados)
ggplot(dados, aes(pclass, media_idade)) + geom_bar(stat = "identity")


# Conectando no MySQL com dplyr
?src_mysql
con2 <- src_mysql(dbname = "titanicdb", user = "root", password = "dsa1234", host = "localhost")


# Coletando e agrupando os dados - usando R ao invés de SQL
dados2 <- con2 %>%
```

```
  tbl("titanic") %>%
  select(pclass, sex, age, survived, fare) %>%
  filter(survived == 0) %>%
  collect()

head(dados2)


# Manipulando dados
dados2 <- con2 %>%
  tbl("titanic") %>%
  select(pclass, sex, survived) %>%
  group_by(pclass, sex) %>%
  summarise(survival_ratio = avg(survived)) %>%
  collect()

View(dados2)


# Plot
ggplot(dados2, aes(pclass,survival_ratio, color=sex, group=sex)) +
  geom_point(size=3) + geom_line()


# Sumarizando os dados
dados2 <- con2 %>%
  tbl("titanic") %>%
  filter(fare > 150) %>%
  select(pclass,sex,age,fare) %>%
  group_by(pclass,sex) %>%
  summarise(avg_age = avg(age),
            avg_fare = avg(fare))

head(dados2)


# --> Para outros bancos de dados, use RODBC

# Trabalhando com R e SQLite

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap06")
getwd()

# Instalar SQLite
install.packages("RSQLite")
library(RSQLite)

# Remover o banco SQLite, caso exista - Não é obrigatório
system("del exemplo.db") # --> no Windows
# system("rm exemplo.db")    # --> no Mac e Linux

# Criando driver e conexão ao banco de dados
drv = dbDriver("SQLite")
con = dbConnect(drv, dbname = "exemplo.db") # como o banco não existe, está sendo criado automaticamente, no
diretorio atual
dbListTables(con)
```

```
# Criando uma tabela e carregando com dados do dataset mtcars
dbWriteTable(con, "mtcars", mtcars, row.names = TRUE)

# Listando uma tabela
dbListTables(con)
dbExistsTable(con,"mtcars") # verifica se a tabela existe
dbExistsTable(con,"mtcars2")
dbListFields(con,"mtcars") # lista os campos da tabela

# Lendo o conteúdo da tabela
dbReadTable(con, "mtcars")

# Criando apenas a definição da tabela.
dbWriteTable(con, "mtcars2", mtcars[0, ], row.names = TRUE) # mtcars2[0,]  indica que será criado com as mesmas
colunas, mas sem os dados
dbListTables(con)
dbReadTable(con, "mtcars2")

# Executando consultas no banco de dados
query = "select row_names from mtcars"
rs = dbSendQuery(con, query) # enviando a query para o banco de dados
dados = fetch(rs, n = -1) # colocando todos o resultado de uma vez
dados
class(dados)

# Executando consultas no banco de dados
query = "select row_names from mtcars"
rs = dbSendQuery(con, query)
while (!dbHasCompleted(rs))
{
  dados = fetch(rs, n = 1) # colocando o resultado linha a linha
  print(dados$row_names)
}

# Executando consultas no banco de dados
query = "select disp, hp from mtcars where disp > 160"
rs = dbSendQuery(con, query)
dados = fetch(rs, n = -1)
dados

# Executando consultas no banco de dados
query = "select row_names, avg(hp) from mtcars group by row_names"
rs = dbSendQuery(con, query)
dados = fetch(rs, n = -1)
dados

# Criando uma tabela a partir de um arquivo
dbWriteTable(con, "iris", "iris.csv", sep = ",", header = T)
dbListTables(con)
dbReadTable(con, "iris")

# Encerrando a conexão
dbDisconnect(con)

# Carregando dados no banco de dados
# http://dados.gov.br/dataset/iq-indice-nacional-de-precos-ao-consumidor-amplo-15-ipca15
# Criando driver e conexão ao banco de dados
drv = dbDriver("SQLite")
con = dbConnect(drv, dbname = "exemplo.db")

dbWriteTable(con,"indices", "indice.csv",
             sep = "|", header = T)
```

```
dbRemoveTable(con, "indices")

dbWriteTable(con,"indices", "indice.csv",
             sep = "|", header = T)

dbListTables(con)
dbReadTable(con, "indices")

df <- dbReadTable(con, "indices")
df
str(df)
sapply(df, class)

hist(df$setembro)
df_mean <- unlist(lapply(df[, c(4, 5, 6, 7, 8)], mean))
df_mean

dbDisconnect(con)

# Trabalhando com R e MongoDB

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap06")
getwd()


# RMongoDB
# Instalação do Pacote
install.packages("devtools")
library("devtools")
# Instalando a partir do install_github (pacote do devtools): Necessário porque o rmongodb não está mais sendo
mantido no CRAN
install_github("mongosoup/rmongodb", force = TRUE)
library(rmongodb)

# Criando a conexão
# OBS: Antes, é necessário inicializar o mongodb
help("mongo.create")
mongo <- mongo.create()
mongo

# Checando a conexão
mongo.is.connected(mongo)

# Se estiver conectado, traga os bancos de dados:
if(mongo.is.connected(mongo) == TRUE) {
  mongo.get.databases(mongo)
}

# Armazenando o nome de uma das coleções
coll <- "users.contatos"

# Contando os registros em uma coleção
help("mongo.count")
mongo.count(mongo, coll)

# Buscando um registro em uma coleção
mongo.find.one(mongo, coll)
```

```
# Obtendo um vetor de valores distintos das chaves de uma coleção
res <- mongo.distinct(mongo, coll, "city")
head(res)

# Obtendo um vetor de valores distintos das chaves de uma coleção
# E gerando um histograma
pop <- mongo.distinct(mongo, coll, "pop")
hist(pop)

# Contando os elementos
nr <- mongo.count(mongo, coll, list('pop' = list('$lte' = 2)))
print( nr )

# Buscando todos os elementos
pops <- mongo.find.all(mongo, coll, list('pop' = list('$lte' = 2)))
head(pops, 2)

# Encerrando a conexão
mongo.destroy(mongo)

# Criando e validando um arquivo json
library(jsonlite)
json_arquivo <- '{"pop":{"$lte":2}, "pop":{"$gte":1}}'
cat(prettify(json_arquivo))

validate(json_arquivo)

# Big Data na Prática 4 - Customer Churn Analytics

# A rotatividade (churn) de clientes ocorre quando clientes ou assinantes param de fazer negócios
# com uma empresa ou serviço. Também é conhecido como perda de clientes ou taxa de cancelamento.

# Um setor no qual saber e prever as taxas de cancelamento é particularmente útil é o setor de telecomunicações,
# porque a maioria dos clientes tem várias opções de escolha dentro de uma localização geográfica.

# Neste projeto, vamos prever a rotatividade (churn) de clientes usando um conjunto de dados de telecomunicações.
# Usaremos a regressão logística, a árvore de decisão e a floresta aleatória como modelos de Machine Learning.

# Usaremos um dataset oferecido gratuitamente no portal IBM Sample Data Sets.
# Cada linha representa um cliente e cada coluna contém os atributos desse cliente.

# https://www.ibm.com/communities/analytics/watson-analytics-blog/guide-to-sample-datasets/

# Definindo o diretório de trabalho
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap06")
getwd()


# Carregando os pacotes
# OBS: Instale aqueles pacotes que precisarem. Para isso, é possível usar o proprio terminal para execução dos
comandos.
library(plyr)
library(corrplot)
library(ggplot2)
library(gridExtra)
library(ggthemes)
library(caret)
library(MASS)
library(randomForest)
library(party)
```

```
##### Carregando e Limpando os Dados #####

# Os dados brutos contém 7043 linhas (clientes) e 21 colunas (recursos).
# A coluna "Churn" é o nosso alvo.
churn <- read.csv('Telco-Customer-Churn.csv')
View(churn)
str(churn)


# Usamos sapply para verificar o número de valores ausentes (missing) em cada coluna.
# Descobrimos que há 11 valores ausentes nas colunas "TotalCharges".
# Então, vamos remover todas as linhas com valores ausentes.
sapply(churn, function(x) sum(is.na(x)))
churn <- churn[complete.cases(churn), ]


# Olhe para as variáveis, podemos ver que temos algumas limpezas e ajustes para fazer.
# 1. Vamos mudar "No internet service" para "No" por seis colunas, que são:
# "OnlineSecurity", "OnlineBackup", "DeviceProtection", "TechSupport", "streamingTV",
# "streamingMovies".
str(churn)
cols_recode1 <- c(10:15)
for(i in 1:ncol(churn[,cols_recode1])) {
  churn[,cols_recode1][,i] <- as.factor(mapvalues
                                (churn[,cols_recode1][,i], from =c("No internet service"),to=c("No")))
}


# 2. Vamos mudar "No phone service" para "No" para a coluna “MultipleLines”
churn$MultipleLines <- as.factor(mapvalues(churn$MultipleLines,
                                  from=c("No phone service"),
                                  to=c("No")))


# 3. Como a permanência mínima é de 1 mês e a permanência máxima é de 72 meses,
# podemos agrupá-los em cinco grupos de posse (tenure):
# “0-12 Mês”, “12–24 Mês”, “24–48 Meses”, “48–60 Mês” Mês ”,“> 60 Mês”
min(churn$tenure); max(churn$tenure)

group_tenure <- function(tenure){
  if (tenure >= 0 & tenure <= 12){
    return('0-12 Month')
  }else if(tenure > 12 & tenure <= 24){
    return('12-24 Month')
  }else if (tenure > 24 & tenure <= 48){
    return('24-48 Month')
  }else if (tenure > 48 & tenure <=60){
    return('48-60 Month')
  }else if (tenure > 60){
    return('> 60 Month')
  }
}

churn$tenure_group <- sapply(churn$tenure,group_tenure)
churn$tenure_group <- as.factor(churn$tenure_group)


# 4. Alteramos os valores na coluna “SeniorCitizen” de 0 ou 1 para “No” ou “Yes”.
churn$SeniorCitizen <- as.factor(mapvalues(churn$SeniorCitizen,
                                  from=c("0","1"),
                                  to=c("No", "Yes")))
```

```
# 5. Removemos as colunas que não precisamos para a análise.
churn$customerID <- NULL
churn$tenure <- NULL
View(churn)


##### Análise exploratória de dados e seleção de recursos #####

# Correlação entre variáveis numéricas
numeric.var <- sapply(churn, is.numeric)
corr.matrix <- cor(churn[,numeric.var])
corrplot(corr.matrix, main="\n\nGráfico de Correlação para Variáveis Numéricas", method="number")


# Os encargos mensais e os encargos totais estão correlacionados.
# Então, um deles será removido do modelo. Nós removemos Total Charges.
churn$TotalCharges <- NULL


# Gráficos de barra de variáveis categóricas
p1 <- ggplot(churn, aes(x=gender)) + ggtitle("Gender") + xlab("Sexo") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentual") + coord_flip() + theme_minimal()
p2 <- ggplot(churn, aes(x=SeniorCitizen)) + ggtitle("Senior Citizen") + xlab("Senior Citizen") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentual") + coord_flip() + theme_minimal()
p3 <- ggplot(churn, aes(x=Partner)) + ggtitle("Partner") + xlab("Parceiros") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentual") + coord_flip() + theme_minimal()
p4 <- ggplot(churn, aes(x=Dependents)) + ggtitle("Dependents") + xlab("Dependentes") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentual") + coord_flip() + theme_minimal()
grid.arrange(p1, p2, p3, p4, ncol=2)


p5 <- ggplot(churn, aes(x=PhoneService)) + ggtitle("Phone Service") + xlab("Telefonia") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentual") + coord_flip() + theme_minimal()
p6 <- ggplot(churn, aes(x=MultipleLines)) + ggtitle("Multiple Lines") + xlab("Múltiplas Linhas") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentual") + coord_flip() + theme_minimal()
p7 <- ggplot(churn, aes(x=InternetService)) + ggtitle("Internet Service") + xlab("Internet Service") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentual") + coord_flip() + theme_minimal()
p8 <- ggplot(churn, aes(x=OnlineSecurity)) + ggtitle("Online Security") + xlab("Online Security") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentual") + coord_flip() + theme_minimal()
grid.arrange(p5, p6, p7, p8, ncol=2)


p9 <- ggplot(churn, aes(x=OnlineBackup)) + ggtitle("Online Backup") + xlab("Online Backup") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentual") + coord_flip() + theme_minimal()
p10 <- ggplot(churn, aes(x=DeviceProtection)) + ggtitle("Device Protection") + xlab("Device Protection") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentual") + coord_flip() + theme_minimal()
p11 <- ggplot(churn, aes(x=TechSupport)) + ggtitle("Tech Support") + xlab("Tech Support") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentual") + coord_flip() + theme_minimal()
p12 <- ggplot(churn, aes(x=StreamingTV)) + ggtitle("Streaming TV") + xlab("Streaming TV") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentual") + coord_flip() + theme_minimal()
grid.arrange(p9, p10, p11, p12, ncol=2)


p13 <- ggplot(churn, aes(x=StreamingMovies)) + ggtitle("Streaming Movies") + xlab("Streaming Movies") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentual") + coord_flip() + theme_minimal()
p14 <- ggplot(churn, aes(x=Contract)) + ggtitle("Contract") + xlab("Contract") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentual") + coord_flip() + theme_minimal()
p15 <- ggplot(churn, aes(x=PaperlessBilling)) + ggtitle("Paperless Billing") + xlab("Paperless Billing") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentual") + coord_flip() + theme_minimal()
p16 <- ggplot(churn, aes(x=PaymentMethod)) + ggtitle("Payment Method") + xlab("Payment Method") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentual") + coord_flip() + theme_minimal()
p17 <- ggplot(churn, aes(x=tenure_group)) + ggtitle("Tenure Group") + xlab("Tenure Group") +
```

```
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentual") + coord_flip() + theme_minimal()
grid.arrange(p13, p14, p15, p16, p17, ncol=2)


# Todas as variáveis categóricas parecem ter uma distribuição razoavelmente ampla,
# portanto, todas elas serão mantidas para análise posterior.


##### Modelagem Preditiva #####

# Regressão Logística

# Primeiro, dividimos os dados em conjuntos de treinamento e testes
intrain <- createDataPartition(churn$Churn,p=0.7,list=FALSE)
set.seed(2017)
training <- churn[intrain,]
testing <- churn[-intrain,]


# Confirme se a divisão está correta
dim(training); dim(testing)


# Treinando o modelo de regressão logística - Fitting do Modelo:
?glm
LogModel <- glm(Churn ~ ., family=binomial(link="logit"), data=training)
# Explicando o comando:
# Churn é a variável alvo da previsão; "~" faz parte da sintaxe do comando; "." significa todas as outras variáveis do
conjunto de dados
# family=binomial(link="logit") são os parâmetros que indicam a regressão logística
# data=training é a seleção dos dados de treinamento do modelo de regressão
print(summary(LogModel))


# Análise de Variância - ANOVA

# Os três principais recursos mais relevantes incluem
# Contract, tenure_group e PaperlessBilling.
?anova
anova(LogModel, test="Chisq")


# Analisando a tabela de variância, podemos ver a queda no desvio ao adicionar cada variável
# uma de cada vez. Adicionar InternetService, Contract e tenure_group reduz
# significativamente o desvio residual.
# As outras variáveis, como PaymentMethod e Dependents, parecem melhorar menos o modelo,
# embora todos tenham valores p baixos.
testing$Churn <- as.character(testing$Churn)
testing$Churn[testing$Churn=="No"] <- "0"
testing$Churn[testing$Churn=="Yes"] <- "1"
fitted.results <- predict(LogModel,newdata=testing,type='response')
fitted.results <- ifelse(fitted.results > 0.5,1,0)
misClasificError <- mean(fitted.results != testing$Churn)
print(paste('Logistic Regression Accuracy',1-misClasificError))


# Matriz de Confusão de Regressão Logística
print("Confusion Matrix Para Logistic Regression"); table(testing$Churn, fitted.results > 0.5)


# Odds Ratio

# Uma das medidas de desempenho interessantes na regressão logística é Odds Ratio.
```

```
# Basicamente, odds ratio é a chance de um evento acontecer.
exp(cbind(OR=coef(LogModel), confint(LogModel)))

# Para cada aumento de unidade no encargo mensal (Monthly Charge),
# há uma redução de 2,5% na probabilidade do cliente cancelar a assinatura.


# Árvore de Decisão

# Visualização da Árvore de Decisão
# Para fins de ilustração, vamos usar apenas três variáveis para plotar
# árvores de decisão, elas são “Contrato”, “tenure_group” e “PaperlessBilling”.
?ctree
tree <- ctree(Churn ~ Contract+tenure_group+PaperlessBilling, training)
plot(tree, type='simple')

# 1. Das três variáveis que usamos, o Contrato é a variável mais importante
# para prever a rotatividade de clientes ou não.
# 2. Se um cliente em um contrato de um ano ou de dois anos,
# não importa se ele (ela) tem ou não a PapelessBilling, ele (ela) é menos propenso
# a se cancelar a assinatura.
# 3. Por outro lado, se um cliente estiver em um contrato mensal,
# e no grupo de posse de 0 a 12 meses, e usando o PaperlessBilling,
# esse cliente terá mais chances de cancelar a assinatura.


# Matriz de Confusão da Árvore de Decisão
# Estamos usando todas as variáveis para tabela de matriz de confusão de produto e fazer previsões.
pred_tree <- predict(tree, testing)
print("Confusion Matrix Para Decision Tree"); table(Predicted = pred_tree, Actual = testing$Churn)


# Precisão da árvore de decisão
p1 <- predict(tree, training)
tab1 <- table(Predicted = p1, Actual = training$Churn)
tab2 <- table(Predicted = pred_tree, Actual = testing$Churn)
print(paste('Decision Tree Accuracy',sum(diag(tab2))/sum(tab2)))


##### Random Forest #####

set.seed(2017)
?randomForest
rfModel <- randomForest(Churn ~ ., data = training)
print(rfModel)
plot(rfModel)

# A previsão é muito boa ao prever "Não".
# A taxa de erros é muito maior quando se prevê "sim".

# Prevendo valores com dados de teste
pred_rf <- predict(rfModel, testing)

# Confusion Matrix
print("Confusion Matrix Para Random Forest"); table(testing$Churn, pred_rf)

# Recursos mais importantes
?varImpPlot # função usada para saber a importancia das variaveis no modelo
varImpPlot(rfModel, sort=T, n.var = 10, main = 'Top 10 Feature Importance')
```

Capítulo 07:

```
# Solução Lista de Exercícios - Capítulo 6

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap07")
getwd()


# Exercicio 1 - Instale e carregue os pacotes necessários para trabalhar com SQLite e R
install.packages(c("dbplyr", "RSQLite"))
library(RSQLite)
library(dbplyr)
library(dplyr)


# Exercicio 2 - Crie a conexão para o arquivo mamiferos.sqlite em anexo a este script
mamiferos <- dbConnect(SQLite(), "mamiferos.sqlite")


# Exercicio 3 - Qual a versão do SQLite usada no banco de dados?
# Dica: Consulte o help da função src_dbi()
?src_dbi
src_dbi(mamiferos) # traz versão do banco de dados e tabelas do banco de dados


# Exercicio 4 - Execute a query abaixo no banco de dados e grave em um objeto em R:
# SELECT year, species_id, plot_id FROM surveys
?tbl
dados <- tbl(mamiferos, sql("SELECT year, species_id, plot_id FROM surveys"))


# Exercicio 5 - Qual o tipo de objeto criado no item anterior?
class(dados)


# Exercicio 6 - Já carregamos a tabela abaixo para você.
# Selecione as colunas species_id, sex e weight com a seguinte condição:
# Condição: weight < 5
# OBS: é uma forma moderna de manipular dados em R.
pesquisas <- tbl(mamiferos, "surveys")

pesquisas %>%
  filter(weight < 5) %>%
  select(species_id, sex, weight)


# Exercicio 7 - Grave o resultado do item anterior em um objeto R.
# O objeto final deve ser um dataframe
dados2 <- pesquisas %>%
  filter(weight < 5) %>%
  select(species_id, sex, weight)

class(dados2) # formato tbl
dados3 <- as.data.frame(dados2) # convertendo tbl para dataframe
class(dados3)
View(dados3)
```

```
# Exercicio 8 - Liste as tabelas do banco de dados carregado
dbListTables(mamiferos)


# Exercicio 9 - A partir do dataframe criado no item 7, crie uma tabela no banco de dados
dbWriteTable(mamiferos, "dados3", dados3)
# criando tabela com dbWriteTable, onde:
# mamiferos é a conexao, "dados3" é o nome da tabela e dados3 é o dataframe

dbRemoveTable(mamiferos, "dados3") # caso a tabela já exista, basta remover
dbListTables(mamiferos)


# Exercicio 10 - Imprima os dados da tabela criada no item anterior
dbReadTable(mamiferos, "dados3")

# Limpeza, Formatação e Manipulação de Dados em R
# dplyr - Transformação de Dados

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap07")
getwd()


# Instalando os pacotes
install.packages("readr") # para leitura de conjunto de dados
install.packages("dplyr")
library(readr)
library(dplyr)


# Carregando o dataset
sono_df <- read_csv("sono.csv")
View(sono_df)
head(sono_df)
class(sono_df) # tabela dataframe ou tbl df
str(sono_df) # resumo completo com todos os tipos de dados


# Função glimpse() pode ser usada no lugar da função str()
glimpse(sono_df)


# Aplicando mutate
glimpse(mutate(sono_df, peso_libras = sono_total / 0.45359237))
# mutate pega o dataframe sono_df e gera uma nova coluna peso_libras
# na verdade, com o glimpse, apenas se está verificando como seria a nova coluna


# Contagem e histograma
count(sono_df, cidade) # contagem agrupada por cidade
hist(sono_df$sono_total) # distribuição de frequência de determinada variável


# Amostragem
sample_n(sono_df, size = 10)
```

```
# select()
sleepData <- select(sono_df, nome, sono_total)
# sintaxe do select: (dados, coluna1, coluna2, ...)
# ou range de colunar com: select(dados, coluna_Inicio:coluna_Fim)
head(sleepData)
class(sleepData)
select(sono_df, nome)
select(sono_df, nome:cidade)
select(sono_df, nome:pais)


# filter()
filter(sono_df, sono_total >= 16)
filter(sono_df, sono_total >= 16, peso >= 80)
filter(sono_df, cidade %in% c("Recife", "Curitiba"))


# arrange() - ordenação ou sort
sono_df %>% arrange(cidade) %>% head

sono_df %>%
  select(nome, cidade, sono_total) %>%
  arrange(cidade, sono_total) %>%
  head

sono_df %>%
  select(nome, cidade, sono_total) %>%
  arrange(cidade, sono_total) %>%
  filter(sono_total >= 16)

sono_df %>%
  select(nome, cidade, sono_total) %>%
  arrange(cidade, desc(sono_total)) %>% # desc - ordena de forma decrescente
  filter(sono_total >= 16)


# mutate()
head(sono_df)
sono_df %>%
  mutate(novo_indice = sono_total / peso) %>%
  head

# percebe-se que o dataframe continua original,
# pois o mutate foi aplicado apenas em tempo de execução
head(sono_df)


# gerando mais de um cálculo de maneira simultânea:
sono_df %>%
  mutate(novo_indice = sono_total / peso,
         peso_libras = peso / 0.45359237) %>%
  head


# summarize()
sono_df %>%
  summarise(media_sono = mean(sono_total))
# a coluna media_sono não existe e está sendo calculada durante sumarização

sono_df %>%
  summarise(media_sono = mean(sono_total),
```

```
            min_sono = min(sono_total),
            max_sono = max(sono_total),
            total = n())


# group_by()
sono_df %>%
  group_by(cidade) %>%
  summarise(avg_sono = mean(sono_total),
            min_sono = min(sono_total),
            max_sono = max(sono_total),
            total = n())
# normalmente, para usar o sumarise, usa-se o group_by. Dessa forma,
# tem-se a sumarização agrupada por variável cidade.


# Operador: %>%
head(select(sono_df, nome, sono_total))

sono_df %>%
  select(nome, sono_total) %>%
  head


# Utilizando várias funções simultâneas do dplyr:
sono_df %>%
  mutate(novo_indice = round(sono_total * peso)) %>%
  arrange(desc(novo_indice)) %>%
  select(cidade, novo_indice)


sono_df
View(sono_df) # dataset se mantém original


# gerando novo dataframe com o resultado das funções:
sono_df2 <- sono_df %>%
  mutate(novo_indice = round(sono_total * peso)) %>%
  arrange(desc(novo_indice)) %>%
  select(cidade, novo_indice)

View(sono_df2)

# Estudo de Caso - Limpando, Transformando e Manipulando Dados de Voos

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap07")
getwd()


# Instalando pacote hflights (Dados de voos de Houston)
install.packages("hflights")
library(hflights)
library(dplyr)
?hflights


# Criando um objeto tbl
```

```
?tbl_df
flights <- tbl_df(hflights)
flights
View(flights)


# Resumindo os dados
# IMPORTANTE: a primeira coisa a se fazer após carregar um data set
# é visualizar os tipos de dados!
str(hflights)
glimpse(hflights)


# Visualizando como dataframe
data.frame(head(flights))


# Filtrando e fatiando os dados com slice
flights[1,1]
flights[flights$Month == 1 & flights$DayofMonth == 1, ]
# Sintaxe: tabela[condição,] - trazendo as linhas pois a condição está no campo coluna


# Aplicando filter do pacote dplyr para substituir o slice:
filter(flights, Month == 1, DayofMonth == 1)
filter(flights, UniqueCarrier == "AA" | UniqueCarrier == "UA")
filter(flights, UniqueCarrier %in% c("AA", "UA"))


# Select
select(flights, Year:DayofMonth, contains("Taxi"), contains("Delay"))
# traz um range de colunas que contenham em seu nome os caracteres Taxi ou Delay


# Organizando os dados
flights %>%
  select(UniqueCarrier, DepDelay) %>%
  arrange(DepDelay)

flights %>%
  select(Distance, AirTime) %>%
  mutate(Speed = Distance/AirTime*60)

head(with(flights, tapply(ArrDelay, Dest, mean, na.rm = TRUE)))
head(aggregate(ArrDelay ~ Dest, flights, mean))

flights %>%
  group_by(Month, DayofMonth) %>%
  tally(sort = TRUE)

# Remodelagem de Dados com tidyr - mudando o shape

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap07")
getwd()


# Instalando os pacotes
```

```
install.packages("tidyr")
library(tidyr)
library(ggplot2)

# Dados de notas em disciplinas
dados <- data.frame(
  Nome = c("Geografia", "Literatura", "Biologia"),
  Regiao_A = c(97, 80, 84),
  Regiao_B = c(86, 90, 91)
)
dados

# Transforma as colunas em linhas, gerando 2 novas colunas, separando as variáveis
dados %>%
  gather(Regiao, NotaFinal, Regiao_A:Regiao_B)

# Criando dados
set.seed(10)
df2 <- data.frame(
  id = 1:4,
  acao = sample(rep(c('controle', 'tratamento'), each = 2)),
  work.T1 = runif(4), # runif gera valores aleatórios
  home.T1 = runif(4),
  work.T2 = runif(4),
  home.T2 = runif(4)
)

df2

# Reshape 1
df2_organizado1 <- df2 %>%
  gather(key, time, -id, -acao)
# pega todas as colunas, menos id e acao, e cria 2 novas variaveis/colunas

df2_organizado1 %>% head(8)

# Reshape 2
df2_organizado2 <- df2_organizado1 %>%
  separate(key, into = c("localidade", "tempo"), sep = "\\.")
# separa key em 2 novas variaveis/colunas, separando os conteúdos pelo ponto

df2_organizado2 %>% head(8)

# Mais um exemplo
set.seed(1)
df3 <- data.frame(
  participante = c("p1", "p2", "p3", "p4", "p5", "p6"),
  info = c("g1m", "g1m", "g1f", "g2m", "g2m", "g2m"),
  day1score = rnorm(n = 6, mean = 80, sd = 15),
  day2score = rnorm(n = 6, mean = 88, sd = 8)
)

print(df3)

# Reshape dos dados
df3 %>%
  gather(day, score, c(day1score, day2score))


df3 %>%
  gather(day, score, c(day1score, day2score)) %>%
  spread(day, score)
# spread faz o oposto do gather. Nesse caso, volta o df ao formato original
```

```
df3 %>%
  gather(day, score, c(day1score, day2score)) %>%
  separate(col = info, into = c("group", "gender"), sep = 2)


df3 %>%
  gather(day, score, c(day1score, day2score)) %>%
  separate(col = info, into = c("group", "gender"), sep = 2) %>%
  unite(infoAgain, group, gender)


df3 %>%
  gather(day, score, c(day1score, day2score)) %>%
  separate(col = info, into = c("group", "gender"), sep = 2) %>%
  ggplot(aes(x = day, y = score)) +
  geom_point() +
  facet_wrap(~ group) +
  geom_smooth(method = "lm", aes(group = 1), se = F)

# Remodelando os Dados com Reshape

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap07")
getwd()

# Pivot (transposta da Matriz)
mtcars
t(mtcars) # t - transposta de matriz

# Reshape
head(iris)
str(iris)
library(lattice)
View(iris)

# Distribuindo os dados verticalmente (long)
?reshape # faz parte do pacote stats (estatística padrão da linguagem R)
iris_modif <- reshape(iris,
                      varying = 1:4,
                      v.names = "Medidas",
                      timevar = "Dimensoes",
                      times = names(iris)[1:4],
                      idvar = "ID",
                      direction = "long")

head(iris_modif)
View(iris_modif)

bwplot(Medidas ~ Species | Dimensoes, data = iris_modif)
# Relaciona os atributos Medidas e Species e, com o resultado, relaciona Dimensoes


iris_modif_sp <- reshape(iris,
                         varying = list(c(1,3),c(2,4)),
                         v.names = c("Comprimento", "Largura"),
                         timevar = "Parte",
```

```
                  times = c("Sepal", "Petal"),
                  idvar = "ID",
                  direction = "long")

head(iris_modif_sp)
View(iris_modif_sp)

xyplot(Comprimento ~  Largura | Species, groups = Parte,
     data = iris_modif_sp, auto.key = list(space="right"))

xyplot(Comprimento ~  Largura | Parte, groups = Species,
     data = iris_modif_sp, auto.key = list(space="right"))


# Reshape2 - modelagem dos dados
install.packages("reshape2")
library(reshape2)

# Criando um dataframe
df = data.frame(nome = c("Zico", "Pele"),
           chuteira = c(40, 42),
           idade = c(34,NA),
           peso = c(93, NA),
           altura = c(175, 178))
df

# "Derretendo" o dataframe - Função melt()
df_wide = melt(df, id = c("nome", "chuteira"))
df_wide

# Removendo valores NA
df_wide = melt(df, id = c("nome", "chuteira"), na.rm = TRUE)
df_wide

# "Esticando" o dataframe - Função dcast()
dcast(df_wide, formula = chuteira + nome ~ variable)
dcast(df_wide, formula = nome + chuteira  ~ variable)
dcast(df_wide, formula = nome  ~ variable)
dcast(df_wide, formula = ...  ~ variable)
# ... indica todas as variáveis se relacionando com variable


# Aplicando o reshape2
names(airquality) <- tolower(names(airquality))
head(airquality)
dim(airquality) # especifica as dimensões

# Função melt() - wide
?melt
df_wide <- melt(airquality)
class(df_wide)
df_wide
head(df_wide)
tail(df_wide)

# Inserindo mais duas variáveis
df_wide <- melt(airquality, id.vars = c("month", "day")) # c() - vetor de variáveis
head(df_wide)

df_wide <- melt(airquality, id.vars = c("month", "day"),
          variable.name = "climate_variable",
          value.name = "climate_value")
```

```
head(df_wide)

# Função dcast() - long
df_wide <- melt(airquality, id.vars = c("month", "day"))
View(df_wide)
df_long <- dcast(df_wide, month + day ~ variable)
View(df_long)
head(airquality)

# Split-Apply-Combine - plyr

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap07")
getwd()

# Instalando os pacotes
install.packages("plyr")
install.packages("gapminder") # pacote de dados sobre a população mundial
# https://www.gapminder.org

library(plyr)
library(gapminder)
?gapminder

# Split-Apply-Combine
?ddply # split df, apply function, return results in a df
df <- ddply(gapminder, ~ continent,
            summarize,
            max_le = max(lifeExp))
str(df)
head(df)
View(df)
levels(df$continent)


# Split-Apply-Combine
ddply(gapminder, ~ continent,
      summarize,
      n_uniq_countries = length(unique(country)))

# Função criada para fazer a mesmo sumarização anterior
ddply(gapminder, ~ continent,
      function(x) c(n_uniq_countries = length(unique(x$country))))


ddply(gapminder, ~ continent,
      summarize,
      min = min(lifeExp),
      max = max(lifeExp),
      median = median(gdpPercap))


# Usando um dataset do ggplot
library(ggplot2)
data(mpg)
str(mpg)
?mpg # data set do ggplot2 com dados de economia de combustível por carros
```

```
# Trabalhando com um subset do dataset
data <- mpg[,c(1,7:9)] # todas as linhas e as colunas 1, 7 a 9
str(data)
View(data)


# Sumarizando e Agregando Dados
ddply(data, .(manufacturer),
      summarize,
      avgcty = mean(cty))


# Várias funções em uma única chamada
ddply(data, .(manufacturer),
      summarize,
      avgcty = mean(cty),
      sdcty = sd(cty),
      maxhwy = max(hwy))


# Sumarizando os dados pela combinação de variáveis/fatores
ddply(data, .(manufacturer, drv),
      summarize,
      avgcty = mean(cty),
      sdcty = sd(cty),
      maxhwy = max(hwy))

# Data.table

# criado para melhorar performance de processamento da linguage R
# Substitua o dataframe convencional pelo data.table p/ grande volume de dados

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap07")
getwd()

# Instalando os pacotes
install.packages("data.table")
library(data.table)


# Criando 2 vetores
vec1 <- c(1, 2, 3, 4)
vec2 <- c('Vermelho', 'Verde', 'Marrom', 'Laranja')


# Criando um data.table
?data.table
dt1 <- data.table(vec1, vec2)
dt1
class(dt1)

# Slicing do data.table
dt2 <- data.table(A = 1:9, B = c("Z", "W", "Q"), C = rnorm(9), D = TRUE)
dt2
class(dt2)
dt2[1,1]
```

```
dt2[3:5,] # linhas de 3 a 5, todas as colunas
dt2[, .(B, C)]

# Aplicando função ao data.table
dt2[, .(Total = sum(A), Mean = mean(C))]
dt2[, plot(A, C)]
dt2[, .(MySum = sum(A)), by = .(Grp = A%%2)]

# Definindo valores por grupos
dt3 <- data.table(B = c("a", "b", "c", "d", "e", "a", "b", "c", "d", "e"),
                  val = as.integer(c(6:10, 1:5)))
dt3

# Operações com data.tables
dt4 <- data.table(A = rep(letters[2:1], each = 4L),
                  B = rep(1:4, each = 2L),
                  C = sample(8))

dt4
new_dt4 <- dt4[, sum(C), by = A] # todas as linhas, soma da coluna C agrupado por A
new_dt4
class(new_dt4)
new_dt4[order(A)] # ordenando datatable pela coluna A

new_dt4 <- dt4[, sum(B), by = A][order(A)]
new_dt4

# Iris
dt5 <- as.data.table(iris)
dt5
dt5[, .(Sepal.Length = median(Sepal.Length),
        Sepal.Width = median(Sepal.Width),
        Petal.Length = median(Petal.Length),
        Petal.Width = median(Petal.Width)),
    by = Species]

# O parâmetro .SD significa Subset Data e um subset é criado considerando a coluna Species e depois
# é calculda a mediana. O resultado deve ser igual ao comando anterior.
# O .SD faz parte da notação do pacote data.table.
dt5[, lapply(.SD, median), by = Species]

# lapply - aplica um loop
# Tradução do comando: mediana para um subset de dados, agrupados por species

# Subsetting

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap07")
getwd()


# Muitas das técnicas abaixo podem ser realizadas com a aplicação das funções:
# subset(), merge(), plyr::arrange()
# Mas conhecer estas notações é fundamental para compreeder como
# gerar subconjuntos de dados


## Vetores
```

```
x <- c("A", "E", "D", "B", "C")
x[]  # retorna todos os elementos
x


# Índices Positivos - Elementos em posições específicas
x[c(1, 3)] # passando um vetor com os indices das colunas para selecao
x[c(1, 1)]
x[order(x)] # ordenando


# Índices Negativos - Ignora elementos em posições específicas
x[-c(1, 3)] # retorne tudo menos as colunas 1 e 3
x[-c(1, 4)]


# Vetor Lógico para gerar subsetting
x[c(TRUE, FALSE)] # x possui 5 elementos, logo esse filtro será: v f v f v
x
x[c(TRUE, FALSE, TRUE, FALSE)]


# Vetor de caracteres
x <- c("A", "B", "C", "D")
y <- setNames(x, letters[1:4])
y
y[c("d", "c", "a")] # nome das colunas (letras minusculas)
y[c("a", "a", "a")]


## Matrizes
mat <- matrix(1:9, nrow = 3)
colnames(mat) <- c("A", "B", "C")
mat
mat[1:2, ]
mat[1:2, 2:3]


# Função outer() permite que uma Matriz se comporte como vetores individuais
?outer # cria uma matriz, mas com formato diferente (elementos sao indexados sequencialmente)
vals <- outer(1:5, 1:5, FUN = "paste", sep = ",")
# concatenou o 1 elem do 1 vetor com o 1 elem do 2 vetor, e assim por diante
# usando como separador a virgula
vals
vals[c(4, 15)]


## Dataframes
df <- data.frame(x = 1:3, y = 3:1, z = letters[1:3])
df
df$x # retorna apenas x
df[df$x == 2, ] # retorna todas as colunas, onde as linhas de x são iguais a 2
df[c(1, 3), ]
df[c("x", "z")]
df[, c("x", "z")]
str(df["x"])
str(df[, "x"])


# Removendo colunas de dataframes
df <- data.frame(x = 1:3, y = 3:1, z = letters[1:3])
df
df$z <- NULL # atribuindo nulo para toda a coluna, ele some do df
```

```
df


# Operadores [], [[]] e $
a <- list(x = 1:3, y = 4:5)
a
a[1] # retorna titulo e elementos do 1o elemento da lista
a[[1]] # retorna apenas os elementos do 1o elemento da lista
a[[1]][[1]] # retorna apenas o 1o elemento do 1o elemento
a[["x"]] # retorna os elementos do elemento x

b <- list(a = list(b = list(c = list(d = 1))))
b
b[[c("a", "b", "c", "d")]]
b[["a"]][["b"]][["c"]][["d"]]


# x$y é equivalente a x[["y", exact = FALSE]]
var <- "cyl"
mtcars$var # aqui n traz nada, pq nao existe coluna var
View(mtcars)
mtcars[[var]] # aqui traz, pois o nome da coluna é o conteudo de var (cyl)

x <- list(abc = 1)
x
x$a # aqui retorna, pois possui a em abc
x[["a"]]
x[["abc"]]


# Subsetting e atribuição
x <- 1:5
x
x[c(1, 2)] <- 2:3
x

x[-1] <- 4:1
x

# Isso é subsetting
head(mtcars)
mtcars[] <- lapply(mtcars, as.integer) # atribuicao nos indices
head(mtcars)


# Isso não é subsetting
mtcars <- lapply(mtcars, as.integer) # atribuicao no dataset inteiro
head(mtcars)


# Lookup tables
x <- c("m", "f", "u", "f", "f", "m", "m")
lookup <- c(m = "Male", f = "Female", u = NA)
lookup[x]
unname(lookup[x]) # sem os titulos das colunas


# Usando operadores lógicos
x1 <- 1:10 %% 2 == 0 # valores de 1 a 10, v para pares e f para impares
x1
which(x1)
x2 <- which(x1) # aqui se movimenta os valores de fato
x2
```

```
y1 <- 1:10 %% 5 == 0
y2 <- which(y1)
y2
intersect(x2, y2) # interseção de valores (pertence a ambos)
x1 & y1
union(x2, y2)
setdiff(x2, y2) # diferença entre os dados

# Estudo de Caso - Extraindo Dados da Web com Web Scraping em R

# Web Crawling - "rastejar" por uma web page ou site buscando dados
# Web Scraping - "raspar" os dados de uma web page


# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap07")
getwd()


# Pacotes R para Web Scraping
# RCurl
# httr
# XML
# rvest

# Pacote rvest - útil para quem não conhece HTML e CSS
install.packages('rvest')
library(rvest)

library(stringr)
library(dplyr)
library(lubridate) # tratamento de datas
library(readr)

# Leitura da web page - Retorna um documento xml (semiestruturado)
webpage <- read_html("https://www.nytimes.com/interactive/2017/06/23/opinion/trumps-lies.html")
webpage


# Extraindo os registros
# Cada elemento na web page acima tem o seguinte formato em html:
# <span class="short-desc"><strong> DATE </strong> LIE <span class="short-truth"><a href="URL">
EXPLANATION </a></span></span>
?html_nodes # função do pacote rvest
results <- webpage %>% html_nodes(".short-desc")
results


# Construindo o dataset
# criando lista vazia com o comprimento de results (que é a lista de tags short-desc)
records <- vector("list", length = length(results))
records

for (i in seq_along(results)) { # de i até comprimento de results
  # buscando a data:
  date <- str_c(results[i] %>% # concatenando
                html_nodes("strong") %>% # o que existe em strong
                html_text(trim = TRUE), ', 2017') # recupera o texto dentro de strong
```

```
                                    # retirando espaços com o trim
                                    # adicionando '2017' no texto

  # buscando a mentira (da página mentiras do Trump):
  lie <- str_sub(xml_contents(results[i])[2] %>%    # str_sub extrai uma string
                                    # converte results em xml_contents
                html_text(trim = TRUE), 2, -2)

  # buscando a explicação dentro de short-truth
  explanation <- str_sub(results[i] %>%
                         html_nodes(".short-truth") %>%
                         html_text(trim = TRUE), 2, -2)

  # buscando o link:
  url <- results[i] %>%
         html_nodes("a") %>% # busca o node a
         html_attr("href")   # dentro de a o atributo href

  # gravando tudo como um dataframe:
  records[[i]] <- data_frame(date = date, lie = lie, explanation = explanation, url = url)
}


# Dataset final
df <- bind_rows(records) # juntando todas as linhas do objeto records, que é uma lista


# Transformando o campo data para o formato Date em R
df$date <- mdy(df$date) # tratando date movimentando para o próprio date


# Exportando para CSV
write_csv(df, "mentiras_trump.csv")


# Lendo os dados
df <- read_csv("mentiras_trump.csv")
View(df)

# Importando Dados de Softwares Estatísticos (SAS, STATA, SPSS)


# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap07")
getwd()


# Instalando o pacote
install.packages("haven")
library(haven)


# SAS - Importando dataset:
vendas <- read_sas("vendas.sas") # arquivo extensão .sas
?read_sas
class(vendas) # tipo tbl_df
print(vendas)
str(vendas)
```

```
# Stata - Importando dataset:
df_stata <- read_dta("mov.dta") # arquivo extensão .dta
class(df_stata) # tipo tbl_df
str(df_stata)
head(df_stata)


# Pacote Foreign

install.packages("foreign")
library(foreign)

# Stata - Importando dataset:
florida <- read.dta("florida.dta")
tail(florida)
class(florida) # tipo data frame


# SPSS - Importando dataset:

# Para extração de arquivos SPSS (.sav):
# http://cw.routledge.com/textbooks/9780415372985/resources/datasets.asp

dados <- read.spss("international.sav")
class(dados) # tipo lista
head(dados)
df <- data.frame(dados) # convertendo lista para data frame
df
head(df)


# Criando um boxplot
boxplot(df$gdp)


# Se você estiver familiarizado com estatística, você vai ter ouvido falar de Correlação.
# É uma medida para avaliar a dependência linear entre duas variáveis.
# Ela pode variar entre -1 e 1;
# Se próximo de 1, significa que há uma forte associação positiva entre as variáveis.
# Se próximo de -1, existe uma forte associação negativa:
# Quando a correlação entre duas variáveis é 0, essas variáveis são possivelmente independentes:
# Ou seja, não há nenhuma associação entre X e Y.

# Coeficiente de Correlação. Indica uma associação negativa entre GDP e alfabetização feminina
cor(df$gdp, df$f_illit)


# **** Importante ****
# Correlação não implica causalidade
# A correlação, isto é, a ligação entre dois eventos, não implica
# necessariamente uma relação de causalidade, ou seja, que um dos
# eventos tenha causado a ocorrência do outro. A correlação pode
# no entanto indicar possíveis causas ou áreas para um estudo mais
# aprofundado, ou por outras palavras, a correlação pode ser uma
# pista.

# Big Data na Prática - Análise de Séries Temporais no Mercado Financeiro

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding
```

```
# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap07")
getwd()

# Pacote para modelagem financeira quantitativa e um framework de trading:
# http://www.quantmod.com


# Instalar e carregar os pacotes
install.packages("quantmod")
install.packages("xts") # pacote para séries temporais
install.packages("moments") # pacote para séries temporais
library(quantmod)
library(xts)
library(moments)


# Seleção do período de análise
startDate = as.Date("2018-01-21")
endDate = as.Date("2018-06-21")


# Download dos dados do período
# Obs: O Yahoo Finance está passando por mudanças
# e o serviço de cotações online pode estar instável
?getSymbols # pacote do quantmod que permite carregar e gerenciar dados de múltiplas fontes

# buscando cotação (nome do papel), do yahoo (poderia ser google), no range de datas:
getSymbols("PETR4.SA", src = "yahoo", from = startDate, to = endDate, auto.assign = T)
# PETR4.SA = readRDS("PETR4.SA.rds") # essa linha só é necessária caso ocorra erro no comando anterior


# Checando o tipo de dado retornado
class(PETR4.SA) # xts zoo (um dos formatos de séries temporais em R)
is.xts(PETR4.SA) # perguntando se é xts - sequencia de dados ao longo do tempo (série temporal)


# Mostra os primeiros registros para as ações da Petrobras
head(PETR4.SA)
View(PETR4.SA)


# Analisando os dados de fechamento
PETR4.SA.Close <- PETR4.SA[, "PETR4.SA.Close"]
is.xts(PETR4.SA.Close) # como o índice sempre é coletado e nesse caso é a data, trata-se de uma série temporal
?Cl # função do quantmod que permite extrair e transformar colunas de séries temporais
head(Cl(PETR4.SA),5) # basicamente a mesma ação do slice realizado acima


# Agora, vamos plotar o gráfico da Petrobras
# Gráfico de candlestick da Petrobras
?candleChart # função do quantmod que cria gráfico de velas (candle)
candleChart(PETR4.SA)


# Plot dos valores de fechamento:
plot(PETR4.SA.Close, main = "Fechamento Diário Ações Petrobras",
     col = "red", xlab = "Data", ylab = "Preço", major.ticks = 'months',
     minor.ticks = FALSE)
```

```
# Adicionado as bandas de bollinger ao gráfico, com média de 20 períodos e 2 desvios
# Bollinger Band
# Como o desvio padrão é uma medida de volatilidade,
# Bollinger Bands ajustam-se às condições de mercado. Mercados mais voláteis,
# possuem as bandas mais distantes da média, enquanto mercados menos voláteis possuem as
# bandas mais próximas da média
?addBBands # função do quantmod que adiciona bandas de bollinger ao gráfico
addBBands(n = 20, sd = 2)


# Adicionando o indicador ADX, média 11 do tipo exponencial
?addADX # função do quantmod que adiciona indicador ADX ao gráfico
addADX(n = 11, maType = "EMA")


# Calculando logs diários
?log # função da base que calcula logarítmos e exponenciais
# nesse caso, utiliza-se calculo de logarítmo para alteração da escala dos dados
PETR4.SA.ret <- diff(log(PETR4.SA.Close), lag = 1)


# Remove valores NA na prosição 1
PETR4.SA.ret <- PETR4.SA.ret[-1]


# Plotar a taxa de retorno
plot(PETR4.SA.ret, main = "Fechamento Diário das Ações da Petrobras",
     col = "red", xlab = "Data", ylab = "Retorno", major.ticks = 'months',
     minor.ticks = FALSE)


# Calculando algumas medidas estatísticas
statNames <- c("Mean", "Standard Deviation", "Skewness", "Kurtosis")
PETR4.SA.stats <- c(mean(PETR4.SA.ret), sd(PETR4.SA.ret), skewness(PETR4.SA.ret), kurtosis(PETR4.SA.ret))
names(PETR4.SA.stats) <- statNames
PETR4.SA.stats


# Salvando os dados em um arquivo .rds (arquivo em formato binário do R)
# getSymbols("PETR4.SA", src = 'yahoo')
saveRDS(PETR4.SA, file = "PETR4.SA.rds") # Salva os dados em formato binário
Ptr = readRDS("PETR4.SA.rds")
dir()
head(Ptr)
```

Capítulo 08:

```
# Solução Lista de Exercícios - Capítulo 7

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap08")
getwd()

# Formatando os dados de uma página web
library(rvest)
library(stringr)
library(tidyr)

# Exercício 1 - Faça a leitura da url abaixo e grave no objeto pagina
# http://forecast.weather.gov/MapClick.php?lat=42.31674913306716&lon=-
71.42487878862437&site=all&smap=1#.VRsEpZPF84I
pagina <- read_html("http://forecast.weather.gov/MapClick.php?lat=42.31674913306716&lon=-
71.42487878862437&site=all&smap=1#.VRsEpZPF84I")


# Exercício 2 - Da página coletada no item anterior, extraia o texto que contenha as tags:
# "#detailed-forecast-body b  e .forecast-text"
# o comando html_nodes gera um arquivo xml
previsao <- html_nodes(pagina, "#detailed-forecast-body b , .forecast-text")

# Exercício 3 - Transforme o item anterior em texto
texto <- html_text(previsao) # transformando num texto (string unica)
paste(texto, collapse = " ") # colocando espaços


# Exercício 4 - Extraímos a página web abaixo para você. Agora faça a coleta da tag "table"
url <- 'http://espn.go.com/nfl/superbowl/history/winners'
pagina <- read_html(url)
tabela <- html_nodes(pagina, 'table')
class(tabela) # tipo xml_nodeset


# Exercício 5 - Converta o item anterior em um dataframe
tab <- html_table(tabela)[[1]]
class(tab)
head(tab)
View(tab)


# Exercício 6 - Remova as duas primeiras linhas e adicione nomes as colunas
tab <- tab[-(1:2), ]
head(tab)
names(tab) <- c("number", "date", "site", "result")
head(tab)
View(tab)


# Exercício 7 - Converta de algarismos romanos para números inteiros
tab$number <- 1:52
tab$date <- as.Date(tab$date, "%B. %d, %Y")
head(tab)
View(tab)
```

```
# Exercício 8 - Divida as colunas em 2 colunas com vencedores e perdedores
tab <- separate(tab, result, c('winner', 'loser'), sep = ', ', remove = TRUE) # remove o separador
head(tab)
View(tab)


# Exercício 9 - Inclua mais 2 colunas com o score dos vencedores e perdedores
# Dica: Você deve fazer mais uma divisão nas colunas
pattern <- " \\d+$" # padrao com digito p/ indicar quebra de coluna
tab$winnerScore <- as.numeric(str_extract(tab$winner, pattern))
tab$loserScore <- as.numeric(str_extract(tab$loser, pattern))
View(tab) # até aqui o padrao ainda continua, apenas foram criadas as novas colunas
tab$winner <- gsub(pattern, "", tab$winner) # substituido padrao por vazio
tab$loser <- gsub(pattern, "", tab$loser)
head(tab)
View(tab)

# Exercício 10 - Grave o resultado em um arquivo csv
write.csv(tab, 'superbowl.csv', row.names = F)
dir()
```

Capítulo 09:

```
# Medidas de Tendência Central - Média e Mediana

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap09")
getwd()


##### Medidas de Tendência Central #####

# As medidas de tendência central são valores representativos da distribuição
# em torno da qual as outras medidas se distribuem.

# Duas medidas são as mais utilizadas: a média aritmética e a mediana.


##### Média #####

# A média aritmética de um conjunto de n valores, como o próprio nome indica, é obtida somando-se
# todas as medidas e dividindo-se a soma por n.

# Representamos cada valor individual por uma letra (x, y, z, etc.) seguida por um sub-índice, ou seja,
# representamos os n valores da amostra por x1, x2, x3, …, xn, onde x1 é a primeira observação,
# x2 é a segunda e assim por diante.

# Exemplo: A lista abaixo possui as notas de 10 alunos de um curso de graduação no exame final. Calcule a média.
notas = c(6.4, 7.3, 9.8, 7.3, 7.9, 8.2, 9.1, 5.6, 8.5, 6.8)
notas
?mean # funcao do pacote base usada para calcular media aritimetica
mean(notas)
print(mean(notas))


##### Mediana #####

# A mediana é uma medida alternativa à média aritmética para representar o centro da distribuição,
# muito usada em estatística descritiva.

# A mediana de um conjunto de medidas (x1, x2, x3, …, xn) é um valor M tal que pelo menos 50% das medidas
# são menores ou iguais a M e pelo menos 50% das medidas são maiores ou iguais a M.

# Em outras palavras, 50% das medidas ficam abaixo da mediana e 50% acima.

# Se o número de elementos for ímpar, a mediana é o elemento do meio: n / 2
# Se o número de elementos for par, a mediana ainda é o elemento do meio, mas calculado assim: (n + 1) / 2

# Obs: a mediana não precisa ser um valor da lista, mas um valor que divide ao meio a lista

# Exemplo: Os dados da lista abaixo são tempos de vida (em dias) de 8 lâmpadas. Calcule a média e a mediana.
tempos = c(400, 350, 510, 550, 690, 720, 750, 2000)
mean(tempos)
?median # funcao do pacote base stats para calcular mediana
median(tempos)

# Medidas de Tendência Central - Moda, Valores Máximo e Mínimo e Amplitude
```

```
# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap09")
getwd()


##### Moda #####

# A moda de uma distribuição é o valor que ocorre mais frequentemente,
# ou o valor que corresponde ao intervalo de classe com a maior frequência.

# A moda, da mesma forma que a mediana, não é afetada por valores extremos.
# Já a media, é afetada por valores extremos (outliers).

# Uma distribuição de frequência que apresenta apenas uma moda é chamada de unimodal.
# Se a distribuição apresenta dois pontos de alta concentração ela é chamada de bimodal.

# Distribuições bimodais ou multimodais podem indicar que na realidade a distribuição de frequência
# se refere a duas populações cujas medidas foram misturadas.

# Por exemplo, suponha que um lote de caixas de leite longa vida é amostrado e em cada caixa da
# amostra é medido o volume envasado. Se o lote é formado pela produção de duas máquinas de envase
# que estão calibradas em valores diferentes, é possível que o histograma apresente duas modas,
# uma para cada valor de calibração.

# Exemplo: Uma loja de calçados quer saber qual o tamanho mais comprado em um dia de vendas.
# A partir dos dados coletados a seguir, determine o tamanho mais pedido.
tamanhos = c(38, 38, 36, 37, 36, 36, 40, 39, 36, 35, 36)
mean(tamanhos)
median(tamanhos)

# formula para calcular moda:
moda = function(dados) {
  vetor = table(as.vector(dados))
  names(vetor)[vetor == max(vetor)]
}

moda(tamanhos)


##### Valores Máximo e Mínimo #####

# Representam os valores máximos e mínimos da distribuição de dados

# Exemplo: Quais são os valores máximo e mínimo dos tamanhos de sapatos do item anterior.
tamanhos = c(38, 38, 36, 37, 36, 36, 40, 39, 36, 35, 36)
max(tamanhos) # funcao para calculo de valor maximo
min(tamanhos) # funcao para calculo de valor minimo


##### Amplitude #####

# A amplitude é a diferença entre o maior e menor valor de um conjunto de dados qualquer.

# Exemplo: Bob quer aprender a voar com asa delta, e ele quer saber qual a amplitude máxima que um voo pode ter.
# A partir dos dados de outros praticantes de voo livre, determine qual a amplitude.
dados = c(28, 31, 45, 58, 22, 33, 42, 68, 24, 37)
range(dados) # funcao que define os valores minimos e maximos (range)
diff(range(dados)) # funcao que calcula a diferenca entre valores
```

Quartil e Percentil

Obs: Caso tenha problemas com a acentuação, consulte este link:
https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap09")
getwd()

O que são os Quartis e Percentis?

Se o número de observações é grande, é interessante calcular algumas outras medidas de posição.
Essas medidas são uma extensão do conceito de mediana.

Suponha que estamos conduzindo um experimento com animais.
Eles recebem uma droga e medimos o tempo de vida (em dias)
após a ingestão dessa droga. Poderíamos fazer a seguinte pergunta:

Qual é o tempo em que 50% dos animais ainda estão vivos? Obviamente esse valor será a mediana.

Poderíamos estar interessados em saber qual é o tempo em que 75% dos animais estão vivos. Ou 25%.

Esses valores são chamados de Quartis da distribuição (dividem a distribuição em quartas partes)
e são representados por Q1 (1º quartil – 25%) e Q3 (3º quartil – 75%).
O segundo quartil, Q2, que corresponde a 50%, é a mediana.

Esse conceito pode ser estendido um pouco mais, e em lugar de 25%, 50% e 75%,
podemos querer calcular percentis (5%, 10%, 90%).

Quartis

Quartis são valores que dividem um conjunto de dados em quatro partes iguais.
O primeiro quartil será o termo que terá 25% dos dados antes dele,
o segundo quartil também seguirá a mesma lógica e irá coincidir com a mediana,
o terceiro quartil será o termo com 75% dos valores do conjunto de dados antes dele
e o quarto quartil será o último termo do conjunto com 100% dos dados antes dele.

# Exemplo: O horário de funcionamento de um banco já está se esgotando, para adiantar o atendimento dos clientes o
# gerente decide para de chamar individualmente e passa a chamar em grupos de 1/4 da quantidade total de clientes na fila.
# A partir dos números das fichas dos clientes, determine os grupos das 4 chamadas.
num_fichas = c(54, 55, 56, 57, 58, 59, 60, 61, 62, 63)
?quantile # funcao do pacote stats que calcula os quartis
quantile(num_fichas)

Ou seja, a primeira chamada contemplará os clientes com as fichas de 54 até 56,
a segunda de 57 até 58, a terceira de 59 até 60 e a quarta de 61 até 63.

Percentis

Os percentis são os valores que separam um conjunto de dados em 100 partes iguais.
O percentil 10 representa o décimo percentil e terá 10% dos dados antes dele,
a lógica se seguirá para todo percentil.

# Exemplo: Consoderando os dados do exemplos anterior, calcule o percentil 10, 80 e 98.
num_fichas = c(54, 55, 56, 57, 58, 59, 60, 61, 62, 63)
quantile(num_fichas, c(.10)) # funcao para calcular percentis

```
quantile(num_fichas, c(.80)) # o ponto antes do numero informa percentual de numero
quantile(num_fichas, c(.98)) # todos os calculos poderiam ter sido feitos de uma vez, separando os valores  por virgula

# Ou seja, o cliente que está com pouco mais do que 98% da fila a frente dele terá a ficha 63,
# o que está com pouco menos do que 80% da fila a frente dele terá a ficha 61
# e o que está com pouco mais do que 10% da fila a frente dele terá a ficha 55.

# Medidas de Dispersão - Desvio Padrão e Variância

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap09")
getwd()


##### Desvio Padrão #####

# O desvio padrão indica o grau de variação de um conjunto de dados, este conjunto pode ser amostral ou populacional.

# Para um conjunto amostral o desvio padrão é dado pelo somatório da raiz quadrada do quadrado da diferença entre
# o valor do dado coletado (xi) e o valor médio (x), dividido pelo tamanho amostral menos um (n−1).

# Exemplo: Um engenheiro precisa decidir entre três modelos de máquinas de corte de alta precisão,
# para isso ele usa como critério o desvio padrão. A máquina que tiver menor desvio será a escolhida por ele.
# A partir dos dados de medida de corte das 3 máquinas, determine qual deve ser a escolhida pelo engenheiro.
# Máquina 1 (mm) = (181.9, 180.8, 181.9, 180.2, 181.4).
# Máquina 2 (mm) = (180.1, 181.8, 181.5, 181.2, 182.4).
# Máquina 3 (mm) = (182.1, 183.7, 182.1, 180.2, 181.9).

Maq1 = c(181.9, 180.8, 181.9, 180.2, 181.4)
Maq2 = c(180.1, 181.8, 181.5, 181.2, 182.4)
Maq3 = c(182.1, 183.7, 182.1, 180.2, 181.9)

mean(Maq1)
mean(Maq2)
mean(Maq3)

sd(Maq1) # funcao do pacote stats para calcular o desvio padrao
sd(Maq2) # desvio padrao informa quão distante os dados estão da média
sd(Maq3)

# O desvio padrão é usado para medir a variabilidade entre os números em um conjunto
# de dados. Assim como o nome sugere, o desvio padrão é um padrão de desvio (distância)
# da média.
# Em termos bem simples, o desvio padrão é a distância média, da média.

# O desvio padrão é uma estatística importante, mas frequentemente é omitida quando
# os resultados são relatados. Sem ele, você está recebendo apenas uma parte da
# história sobre os dados.
# Por exemplo, a média de salários de uma empresa podem não representar verdadeiramente
# o que está se passando se os salários forem extremamente discrepantes.

##### Variância #####

# A variância também é um importante indicador de variabilidade dos dados.
# Como a soma dos desvios sempre somarão zero, é necessário usar uma medida de variabilidade que torne
# os desvios negativos em valores não negativos, para que a soma dos desvios represente um valor
# de variabilidade do conjunto de dados diferente de zero.
```

```
# A variância então é o quadrado do desvio padrão.

# As medidas amostrais tem n−1 graus de liberdade.

# Graus de liberdade é a diferença entre
# a dimensão da amostra (n) e a quantidade de parâmetros a serem avaliados na população.

# No caso da variância amostral, é usada como referência a média da amostra e isso tornaria o
# valor da variância amostral menor do que o da variância populacional, devido aos valores
# coletados estarem mais próximos da média amostral.

# Para corrigir isso o divisor perde 1 grau de liberdade e quando se trata das medidas
# amostrais utilizamos o n−1.

# Cálculo da variância para o exemplo anterior.

var(Maq1) # funcao do pacote stats para calcular a variância
var(Maq2)
var(Maq3)

# Medidas de Dispersão - Coeficiente de Variação

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap09")
getwd()


##### Coeficiente de Variação #####

# O coeficiente de variação indica a quantidade de variação de um conjunto de dados em relação a média.
# O valor é dado por uma relação direta do quociente entre o desvio com a média da amostra.
# Obs: quociente = divisão

# O coeficiente de variação (CV), mede o desvio padrão em termos de percentual da média.
# Um CV alto, indica alta variabilidade dos dados, ou seja, menos consistência dos dados.
# Um CV menor, indica mais consistência dentro do conjunto de dados.

# Quando comparamos a consistência entre 2 conjuntos de dados em relação a suas médias,
# é melhor feito quando utilizamos coeficiente de variação.


# Exemplo: Imagine que um investidor está decidindo se compra ações da Nike ou Adidas na bolsa de valores.
# O valor médio da ação de cada empresa e o desvio padrão, são dados a seguir.
# Qual deve ser a escolha do investidor?

# Nike ==> Valor médio da ação = $55.62 / desvio padrão = $5.10
# Adidas ==> Valor médio da ação = $24.86 / desvio padrão = $3.60

# CV = (desvio/media) * 100
CV_Nike   = (5.10/55.62) * 100 # formula do coeficiente de variação
CV_Adidas = (3.60/24.86) * 100

print(CV_Nike)
print(CV_Adidas)

# Conclusão

# Um investidor se sentiria mais seguro em adquirir ações da Nike, pois o preço das ações
```

```
# teria uma variação menor, podendo assim evitar perdas e permitindo ao investidor ter
# um investimento mais seguro.

# Interpretando um Histograma

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap09")
getwd()


##### Histograma #####

# Um histograma é um modelo de gráfico que representa uma distribuição de frequências
# através de um agrupamento de classes, de forma que se pode contabilizar as ocorrências dos
# dados em cada classe.

# Possibilita visualizar a distribuição de medidas, a dispersão, simetria dos dados e tendências centrais.

# Os conceitos de Frequência Absoluta e Frequência Relativa são importantes na construção de um histograma.

# Por frequência absoluta, entende-se o número de observações correspondente a cada classe.
# Obs: observações são as linhas de uma tabela.

# A frequência relativa, por sua vez, diz respeito ao quociente entre a frequência absoluta da classe
# correspondente e a soma das frequências absolutas.

# A soma das áreas de todos os retângulos do histograma deve ser igual a 1.

# Como fazer um histograma manualmente?

# 1- Ordenar os valores
# 2- Encontrar a amplitude total: A = xmax − xmin. Assim, os intervalos devem cobrir uma faixa de, no mínimo, o valor da amplitude.
# 3- Estimar o número de classes: 2k ≤ n. Sendo que n é igual a raiz quadrada do número total de observações.
# 4- Estimar o tamanho de cada intervalo de classe: C = A/k
# 5- Contar o número de observações que caem em cada intervalo de classe (subintervalo), frequência.
# 6- Determinar a frequência relativa do intervalo: Frequência relativa = frequência/total de observações.
# 7- Construir o gráfico.

# Exemplo: Os seguintes dados representam o número de acidentes diários em um complexo industrial
# (colocados em ordem crescente), durante o período de 50 dias. Represente o histograma desses dados.

dados = c(18, 20, 20, 21, 22, 24, 25, 25, 26, 27, 29, 29,
          30, 30, 31, 31, 32, 33, 34, 35, 36, 36, 37, 37,
          37, 37, 38, 38, 38, 40, 41, 43, 44, 44, 45, 45,
          45, 46, 47, 48, 49, 50, 51, 53, 54, 54, 56, 58, 62, 65)

hist(dados, main = "Número de Acidentes Diários", xlab = "Acidentes", ylab = "Frequência")
hist(dados, main = "Número de Acidentes Diários", xlab = "Acidentes", ylab = "Frequência", breaks = 6) # breaks é o número de classes
hist(dados, main = "Número de Acidentes Diários", xlab = "Acidentes", ylab = "Frequência", breaks = 5) # o interpretador da linguagem R fez o ajuste, pois para esses dados 5 não é o número de classes ideal
?hist # funcao para criar histograma

# Coeficiente de Assimetria (Skewness)

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding
```

```
# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap09")
getwd()


##### Coeficiente de Assimetria #####

# O coeficiente de assimetria é o que permite dizer se uma determinada distribuição é assimétrica ou não.

# Exemplo: Os seguintes dados representam o número de acidentes diários em um complexo industrial
# (colocados em ordem crescente), durante o período de 50 dias. Represente o histograma desses dados.

dados = c(18, 20, 20, 21, 22, 24, 25, 25, 26, 27, 29, 29,
          30, 30, 31, 31, 32, 33, 34, 35, 36, 36, 37, 37,
          37, 37, 38, 38, 38, 40, 41, 43, 44, 44, 45, 45,
          45, 46, 47, 48, 49, 50, 51, 53, 54, 54, 56, 58, 62, 65)

hist(dados, main = "Número de Acidentes Diários", xlab = "Acidentes", ylab = "Frequência")

mean(dados)
sd(dados)
median(dados)

library(moments) # carregando pacote moments (já vem na linguagem R)
?skewness
SK = skewness(dados) # função para calcular coeficiente de assimetria
print(SK)

# Sk ≈ 0: dados simétricos. Tanto a cauda do lado direito quanto a do lado esquerdo da função densidade de probabilidade são iguais.
# Sk < 0: assimetria negativa. A cauda do lado esquerdo da função densidade de probabilidade é maior que a do lado direito.
# Sk > 0: assimetria positiva. A cauda do lado direito da função densidade de probabilidade é maior que a do lado esquerdo.

# O coeficiente de assimetria é 0.2549279.
# Como o coeficiente de assimetria é maior que 0, diz-se que a curva apresenta assimetria positiva
# e a cauda do lado direito da função densidade de probabilidade é maior que no lado esquerdo.
# Ao observar também o Histograma, percebe-se que há maior densidade de dados do lado direito.

# Obs: Antes de aplicar modelo de machine learning é importante tratar os dados
# para que os mesmos fiquem simétricos.

# Outro exemplo
set.seed(1234)
x = rnorm(1000)
hist(x)
skewness(x)

# Coeficiente de Curtose (Kurtosis)

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap09")
getwd()
```

```
##### Coeficiente de Assimetria #####

# O coeficiente de curtose é uma medida que caracteriza o achatamento da curva da função de distribuição.

# Exemplo: Os seguintes dados representam o número de acidentes diários em um complexo industrial
# (colocados em ordem crescente), durante o período de 50 dias. Represente o histograma desses dados.

dados = c(18, 20, 20, 21, 22, 24, 25, 25, 26, 27, 29, 29,
          30, 30, 31, 31, 32, 33, 34, 35, 36, 36, 37, 37,
          37, 37, 38, 38, 38, 40, 41, 43, 44, 44, 45, 45,
          45, 46, 47, 48, 49, 50, 51, 53, 54, 54, 56, 58, 62, 65)

hist(dados, main = "Número de Acidentes Diários", xlab = "Acidentes", ylab = "Frequência")

mean(dados)
sd(dados)
median(dados)

library(moments)
?kurtosis
CK = kurtosis(dados) # função para calcular coeficiente de curtose
print(CK)

# CK ≈ 0: Distribuição normal. Chamada de Curtose Mesocúrtica.
# CK < 0: Cauda mais leve que a normal. Para um coeficiente de Curtose negativo, tem-se uma Curtose Platicúrtica.
# CK > 0: Cauda mais pesada que a normal. Para um coeficiente de Curtose positivo, tem-se uma Curtose Leptocúrtica.

# O coeficiente de curtose é igual a 2.37652. Logo, como o valor de CK é maior que 0, a curva é Leptocúrtica.

# Outro exemplo
n.sample <- rnorm(n = 10000, mean = 55, sd = 4.5)

# Skewness e Kurtosis
library(moments)
skewness(n.sample)
kurtosis(n.sample)

# Histograma
library(ggplot2)
datasim <- data.frame(n.sample)
ggplot(datasim, aes(x = n.sample), binwidth = 2) +
  geom_histogram(aes(y = ..density..), fill = 'red', alpha = 0.5) +
  geom_density(colour = 'blue') + xlab(expression(bold('Dados'))) +
  ylab(expression(bold('Densidade')))

# Interpretando um BoxPlot

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap09")
getwd()


##### Boxplot #####

# Box-plot, ou diagrama de caixa, é possível obter informações sobre vários aspectos dos dados simultaneamente como,
# outliers, dispersão, tendências centrais, erros padrão e simetria.
```

```
# Utilizado para avaliar a distribuição empírica dos dados, é formado pelo primeiro e terceiro quartis,
# juntamente com a mediana.

dados = c(18, 20, 20, 21, 22, 24, 25, 25, 26, 27, 29, 29,
          30, 30, 31, 31, 32, 33, 34, 35, 36, 36, 37, 37,
          37, 37, 38, 38, 38, 40, 41, 43, 44, 44, 45, 45,
          45, 46, 47, 48, 49, 50, 51, 53, 54, 54, 56, 58, 62, 65)

mean(dados)
sd(dados)
median(dados)
range(dados)
quantile(dados)

boxplot(dados, main = "Número de Acidentes Diários")

# Covariância e Coeficiente de Correlação

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap09")
getwd()


##### Covariância #####

# A covariância entre duas variáveis (X, Y) é uma medida de variabilidade conjunta dessas duas variáveis aleatórias.
# Quando a covariâncias entre essas variáveis é positiva os dados apresentam tendência positiva na dispersão.
# Quando o valor da covariância é negativo, o comportamento é análogo, no entanto, os dados apresentam tendências
negativas.

# Covariância é uma medida de como as alterações em uma variável estão associadas a mudanças em uma segunda
variável.
# Especificamente, a covariância mede o grau em que duas variáveis estão linearmente associadas.
# No entanto, também é frequentemente usado informalmente como uma medida geral de como duas variáveis são
# monotonicamente relacionadas.


##### Coeficiente de Correlação #####

# Correlação é uma versão em escala de covariância que assume valores em [−1,1]
# com uma correlação de ± 1 indicando associação linear perfeita e 0 indicando nenhuma relação linear.
# Esse escalonamento torna a correlação invariante às mudanças na escala das variáveis originais
# A constante de escala é o produto dos desvios padrão das duas variáveis.

# Portanto, o Coeficiente de Correlação p mede o grau de correlação entre duas variáveis.

# Para p = 1, tem-se uma correlação perfeita entre as duas variáveis.
# Se p = - 1, há uma correlação perfeita entre as variáveis, no entanto, essa correlação é negativa.
# Caso p = 0, as duas variáveis não dependem linearmente uma da outra.

# Para p = -1 indica uma forte correlação negativa: isso significa que toda vez que x aumenta, y diminui
# Para p = 0 significa que não há associação entre as duas variáveis (x e y)
# Para p = 1 indica uma forte correlação positiva: isso significa que y aumenta com x

# Exemplo: Analisar a covariância e correlação entre as variáveis milhas/galão e peso do veículo no dataset mtcars.

my_data <- mtcars # mtcars é um dataset que vem na linguagem R
View(my_data)
```

```
install.packages("ggpubr")
library("ggpubr") # se der erro, instale os pacotes listados. ex: ggplot2 ou magrittr

# funcao para criar grafico de dispersao (pontinhos) e calcular coeficiente de correlacao
ggscatter(my_data, x = "mpg", y = "wt", # x é milhas por galão e y o peso
          add = "reg.line", conf.int = TRUE,
          cor.coef = TRUE, cor.method = "pearson",
          xlab = "Autonomia", ylab = "Peso do Veículo") # labels para x e y

# ### Interpretando o gráfico: ###
# À medida que diminui o peso, aumenta a autonomia. Exemplo de correlação negativa;
#

# Definindo x e y
x = my_data$mpg
y = my_data$wt

# Covariância
?cov # funcao para calcular covariancia
cov(x, y)

# Correlação - mais fácil de interpretar, pois vai de -1 a 1
?cor # funcao para calcular correlacao
cor(x, y)


# Obs: O fato de calcular correlação não implica, necessariamente, causalidade.
# Ou seja, no exemplo, não posso afirmar que a perda da autonomia está condicionada
# ao aumento do peso do veículo, mas sim que existe uma correlação entre as variáveis.

# Distribuição Binomial

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap09")
getwd()


# Uma variável aleatória tem Distribuição Binomial quando o experimento ao
# qual está relacionada apresenta apenas 2 resultados: sucesso e fracasso.

# Vamos imaginar que nosso experimento seja contar quantos clientes que entram em uma loja de celulares,
# adquirem um plano pós-pago.

# Para este experimento, temos 2 possibilidades para cada observação: adquirir ou não adquirir o plano.

# Como podemos ter apenas 2 resultados possíveis, este é chamado um Experimento Binomial.

# Vamos imaginar agora, que historicamente, 10% dos clientes que entram na loja, adquirem um plano pós-pago.
# Portanto, a probabilidade de sucesso (que vamos chamar de p) para cada observação é 0.10.
# E a probabilidade de falha (que vamos chamar de q) para cada observação é 0.90.

# Ou seja: p = 1 – q

# p = probabilidade de sucesso
# q = probabilidade de fracasso

# A palavra sucesso não tem relação com resultado positivo do experimento, assim como a palavra fracasso
```

```
# não tem conotação negativa.

# Outra característica da distribuição binomial, é que cada observação é independente das outras.

# Dessa forma, um Experimento Binomial consiste de um número fixo de observações,
# indicado por n e contamos o número de sucessos, indicado por x.

# A Média de uma Distribuição Binomial, representa a média de longo prazo de sucessos esperados,
# baseado no número de observações.

# A Variância de uma Distribuição Binomial, representa a variação que existe no número de sucessos (p)
# sobre um número (n) de observações.



# Exemplo: A probabilidade de um paciente com um ataque cardíaco morrer do ataque é de 0.04
# (ou seja, 4 de 100 morrem do ataque). Suponha que tenhamos 5 pacientes que sofrem um ataque cardíaco,
# qual é a probabilidade de que todos sobrevivam?

# Para este exemplo, vamos chamar um sucesso um ataque fatal (p = 0.04).
# Temos n = 5 pacientes e queremos saber a probabilidade de que todos sobrevivam ou, em outras palavras,
# que nenhum seja fatal (0 sucessos).

# X = Número de sobreviventes ao ataque
# p = 0.04
# n = 5
# dbinom(X, n, p)
?dbinom # funcao para criacao de uma distribuicao binomial
a <- dbinom(0, 5, 0.04)
print(a)

# Desenhando a distribuição de probabilidades
graph <- function(n,p){
  x <- dbinom(0:n, size = n, prob = p)
  barplot(x,ylim=c(0,1),names.arg=0:n,
          main=sprintf(paste('Distribuição Binomial (n,p) ',n,p,sep=',')))
}
graph(5,0.04)


# Criando o gráfico de uma distribuição binomial
x <- seq(0,50,by = 1)
y <- dbinom(x,50,0.5)
png(file = "dbinom.png")
plot(x,y)
dev.off()

# Distribuição Poisson

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap09")
getwd()


# A Distribuição Poisson é muito útil para calcular a probabilidade de um certo número de eventos
# que ocorrerá em um específico intervalo de tempo ou espaço.

# Nós poderíamos usar este tipo de distribuição para determinar a probabilidade de 10 clientes
```

```
# entrarem em uma loja nos próximos 30 minutos ou a probabilidade de 2 acidentes de carro ocorrerem
# em um determinado cruzamento no próximo mês.

# A Distribuição Poisson é um modelo para o número de eventos observados numa unidade de tempo ou de espaço,
# dado que a taxa de eventos por unidade é constante e os eventos ocorrem de modo independente.

# O único parâmetro da Poisson é λ (lambda), que representa a taxa de eventos por unidade.

# Se um certo número de objetos está distribuído ao acaso por uma área, e esta área é dividida em quadrículas de
# mesmo tamanho, o número de objetos por quadrículas pode ser descrito por uma Distribuição Poisson. Neste caso,
# o parâmetro λ será o total de objetos dividido pelo total de quadrículas.

# Distribuição Binomial, o número de sucessos observados é limitado ao número de possibilidades.
# Distribuição Poisson, o número de resultados pode ser qualquer um.

# A Distribuição Poisson não conta o número de sucessos, como na distribuição binomial.
# A Distribuição Poisson conta o número de ocorrências de um evento particular sobre um intervalo
# específico de tempo ou espaço.


# Exemplo: Considere um processo que têm uma taxa de 0,5 defeitos por unidade.
# Qual a probabilidade de uma unidade apresentar dois defeitos? E nenhum defeito?
?dpois # funcao para calcular distribuicao poisson
dpois(2, 0.5)
dpois(0, 0.5)

# Distribuição Normal

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap09")
getwd()


# A Distribuição Normal, ou Gaussiana, é a mais importante distribuição contínua.
# Isso por conta de vários fatores, entre eles, o teorema central do limite, o qual é um resultado essencial
# em aplicações práticas e teóricas, pois garante que, mesmo que os dados não sejam distribuídos segundo uma normal,
# a média dos dados converge para uma distribuição normal conforme o número de dados aumenta.

# O R inclui funcionalidades para operações com distribuições de probabilidades.
# Para cada distribuição há 4 operações básicas indicadas pelas letras:

# d: calcula a densidade de probabilidade f(x) no ponto
# p: calcula a função de probabilidade acumulada F(x) no ponto
# q: calcula o quantil correspondente a uma dada probabilidade
# r: retira uma amostra da distribuição

# Para utlizar as funções combina-se uma das letras acima com uma abreviatura do nome da distribuição.
# Por exemplo, para calcular probabilidades usamos: pnorm para normal, pexp para exponencial,
# pbinom para binomial, ppois para Poisson e assim por diante.

# x <- rnorm(n, mean, sd)
# Onde n é o tamanho da amostra e mean e sd são parâmetros opcionais relacionados à média e desvio padrão,
# respectivamente.

# Distribuição Normal
?rnorm # funcao para criacao de uma distribuicao normal
x <- rnorm(100) # passando apenas o numero de elementos
hist(x)
```

```
# Densidade - é a curva de distribuição normal
# # Observe que o gráfico gerado assemelha-se a uma Gaussiana e não apresenta assimentria.
# Quando o gráfico da distribuição possui tal forma, há grandes chances de se tratar de uma distribuição normal.
x <- seq(-6, 6, by=0.01)
y <- dnorm(x)
plot(x, y, type="l")
```

Capítulo 10:

```
# Solução Lista de Exercícios - Capítulo 9

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap10")
getwd()


# Exercício 1 - Gere 1000 números de uma distribuição normal com média 3 e sd = .25 e grave no objeto chamado x.
?rnorm
x <- rnorm(1000, 3, .25)


# Exercício 2 - Crie o histograma dos dados gerados no item anterior e adicione uma camada com a curva da normal.
?hist
hist(x, # dados
     prob=TRUE, # probabilidade para ser usada depois na função curve
     ylim=c(0,1.80), # limites de y ou altura do grafico
     breaks=20, # quantidade de retangulos do histograma
     main = "Histograma de x") # titulo
curve(
  dnorm( # densidade das probabilidades
    x, # dados
    3, # media
    0.25), # desvio padrao
    add=TRUE, # para adicionar as probabilidades
    col="red",
    lwd=1) # espessura da linha


# Exercício 3 - Suponha que 80% dos adultos com alergias relatem alívio sintomático com uma medicação específica.
# Se o medicamento é dado a 10 novos pacientes com alergias, qual é a probabilidade de que ele seja
# eficaz em exatamente sete?
?dbinom # distribuicao binomial
dbinom(7, 10, 0.8)

# Calculando a probabilidade para todas as possibilidades:
graph <- function(n,p){
  x <- dbinom(0:n,size=n,prob=p)
  barplot(x,ylim=c(0,0.4),names.arg=0:n,
          main=sprintf(paste('Binomial Distribution(n,p) ',n,p,sep=',')))
}
graph(10,0.8)



# Exercício 4 - Suponha que os resultados dos testes de um vestibular se ajustem a uma distribuição normal.
# Além disso, a pontuação média do teste é de 72 e o desvio padrão é de 15,2.
# Qual é a porcentagem de alunos que pontuaram 84 ou mais no exame?

# Aplicamos a função pnorm da distribuição normal com média 72 e desvio padrão 15.2.
# Como estamos procurando a porcentagem de alunos com pontuação superior a 84,
# estamos interessados na cauda superior da distribuição normal.
?pnorm
pnorm(84,
      mean=72, # media
```

```
    sd=15.2, # desvio padrao
    lower.tail=FALSE) # cauda inferior igual a faldo traz apenas a cauda superior


# Exercício 5 - Suponha que o tempo médio de check-out de um caixa de supermercado seja de três minutos.
# Encontre a probabilidade de um check-out do cliente ser concluído pelo caixa em menos de dois minutos.

# A taxa de processamento de checkout é igual a uma dividida pelo tempo médio de conclusão da finalização.
# Por isso, a taxa de processamento é de 1/3 de checkouts por minuto.
# Em seguida, aplicamos a função pexp da distribuição exponencial com taxa = 1/3.
pexp(2, rate=1/3)


# Exercício 6 - Selecione dez números aleatórios entre um e três.
# Aplicamos a função de geração runif da distribuição uniforme para gerar dez números aleatórios entre um e três.
runif(10, min=1, max=3)


# Exercício 7 - Se houver 12 carros atravessando uma ponte por minuto, em média,
# encontre a probabilidade de ter 15 ou mais carros cruzando a ponte em um determinado minuto.
# A probabilidade de ter 14 ou menos carros atravessando a ponte em um determinado minuto é dada pela função ppois.
?ppois # distribuicao poisson
ppois(14, lambda=12)   # lower tail - 14 ou menos

# Assim, a probabilidade de ter 15 ou mais carros cruzando a ponte em um minuto está na
# cauda superior da função de densidade de probabilidade.
ppois(15, lambda=12, lower=FALSE)   # upper tail - 15 ou mais


# Exercício 8 - Suponha que haja 12 questões de múltipla escolha em um questionário de inglês.
# Cada questão tem cinco respostas possíveis e apenas uma delas está correta.
# Encontre a probabilidade de ter quatro ou menos respostas corretas se um aluno tentar
# responder a cada pergunta aleatoriamente.

# Como apenas uma das cinco respostas possíveis está correta, a probabilidade de responder corretamente
# a uma pergunta aleatória é 1/5 = 0,2. Podemos encontrar a probabilidade de ter exatamente 4 respostas
# corretas por tentativas aleatórias como segue.
dbinom(4, size=12, prob=0.2)

# Para encontrar a probabilidade de ter quatro ou menos respostas corretas por tentativas aleatórias,
# aplicamos a função dbinom com x = 0,…, 4.
?dbinom # realiza calculos de maneira individual
dbinom(0, size=12, prob=0.2) +
  dbinom(1, size=12, prob=0.2) +
  dbinom(2, size=12, prob=0.2) +
  dbinom(3, size=12, prob=0.2) +
  dbinom(4, size=12, prob=0.2)

# Ou então:
?pbinom
pbinom(4, size=12, prob=0.2) # realiza somando todas as probabilidades
```

Capítulo 11:

```
# Solução Lista de Exercícios - Capítulo 10

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap11")
getwd()


# Pacotes
install.packages("dplyr") # para manipulação de dados
install.packages('nycflights13') # dados sobre voos em 3 aeroportos americanos
library('ggplot2')
library('dplyr')
library('nycflights13')
View(flights) # dataset
?flights

# Definindo o Problema de Negócio
# Crie um teste de hipótese para verificar se os voos da Delta Airlines (DL)
# atrasam mais do que os voos da UA (United Airlines)


##### ATENÇÃO #####
# Você vai precisar do conhecimento adquirido em outros capítulos do curso
# estudados até aqui para resolver esta lista de exercícios!


###--- INICIO DE ATIVIDADE DE AMOSTRAGEM ---###

# Exercício 1 - Construa o dataset pop_data com os dados de voos das
# companhias aéreas UA (United Airlines) e DL (Delta Airlines).
# O dataset deve conter apenas duas colunas, nome da companhia e atraso nos voos de chegada.
# Os dados devem ser extraídos do dataset flights para construir o dataset pop_data
# Vamos considerar este dataset como sendo nossa população de voos
pop_data = na.omit(flights) %>%    ### omite tudo o que for not avaliable
  filter(carrier == 'UA' | carrier == 'DL', arr_delay >= 0) %>% ### filtro
  select(carrier, arr_delay) %>%  ### selecionando apenas as 2 colunas solicitadas
  group_by(carrier) %>%
  sample_n(17000) %>%  ### trazendo apenas uma amostra de 17k registros
  ungroup()

View(pop_data)


# Exercício 2  - Crie duas amostras de 1000 observações cada uma a partir do
# dataset pop_data apenas com dados da companhia DL para amostra 1 e apenas dados
# da companhia UA na amostra 2

# Dica: inclua uma coluna chamada sample_id preenchida com número 1 para a primeira
# amostra e 2 para a segunda amostra
amostra1 = na.omit(pop_data) %>%
  select(carrier, arr_delay) %>%
  filter(carrier == 'DL') %>%
  mutate(sample_id = '1') %>%  ### adiciona uma coluna nova com valor fixo
  sample_n(1000)
```

```
View(amostra1)

amostra2 = na.omit(pop_data) %>%
  select(carrier, arr_delay) %>%
  filter(carrier == 'UA') %>%
  mutate(sample_id = '2') %>%
  sample_n(1000)

View(amostra2)


# Exercício 3 - Crie um dataset contendo os dados das 2 amostras criadas no item anterior.
samples = rbind(amostra1,amostra2)  # rbind pra fazer ligação por linha
View(samples)

###--- FIM DE ATIVIDADE DE AMOSTRAGEM ---###


# Exercício 4 - Calcule o intervalo de confiança (95%) da amostra1

# Usamos a fórmula: erro_padrao_amostra1 = sd(amostra1$arr_delay) / sqrt(nrow(amostra1))

# Esta fórmula é usada para calcular o desvio padrão de uma distribuição da média amostral
# (de um grande número de amostras de uma população). Em outras palavras, só é aplicável
# quando você está procurando o desvio padrão de médias calculadas a partir de uma amostra de
# tamanho n*n*, tirada de uma população.

# Digamos que você obtenha 10000 amostras de uma população qualquer com um tamanho de amostra de n = 2.
# Então calculamos as médias de cada uma dessas amostras (teremos 10000 médias calculadas).
# A equação acima informa que, com um número de amostras grande o suficiente, o desvio padrão das médias
# da amostra pode ser aproximado usando esta fórmula: sd(amostra) / sqrt(nrow(amostra))

# Deve ser intuitivo que o seu desvio padrão das médias da amostra será muito pequeno,
# ou em outras palavras, as médias de cada amostra terão muito pouca variação.

# Com determinadas condições de inferência (nossa amostra é aleatória, normal, independente),
# podemos realmente usar esse cálculo de desvio padrão para estimar o desvio padrão de nossa população.
# Como isso é apenas uma estimativa, é chamado de erro padrão. A condição para usar isso como
# uma estimativa é que o tamanho da amostra n é maior que 30 (dado pelo teorema do limite central)
# e atende a condição de independência n <= 10% do tamanho da população.

# Erro padrão
erro_padrao_amostra1 = sd(amostra1$arr_delay) / sqrt(nrow(amostra1))

# Limites inferior e superior
# 1.96 é o valor de z score para 95% de confiança (consultar tabela de escore z)
lower = mean(amostra1$arr_delay) - 1.96 * erro_padrao_amostra1
upper = mean(amostra1$arr_delay) + 1.96 * erro_padrao_amostra1

# Intervalo de confiança
ic_1 = c(lower,upper)
mean(amostra1$arr_delay)
ic_1


# Exercício 5 - Calcule o intervalo de confiança (95%) da amostra2
erro_padrao_amostra2 = sd(amostra2$arr_delay) / sqrt(nrow(amostra2))
lower = mean(amostra2$arr_delay) - 1.96 * erro_padrao_amostra2
upper = mean(amostra2$arr_delay) + 1.96 * erro_padrao_amostra2
ic_2 = c(lower,upper)
mean(amostra2$arr_delay)
ic_2
```

```
# Exercício 6 - Crie um plot Visualizando os intervalos de confiança criados nos itens anteriores
# Dica: Use o geom_point() e geom_errorbar() do pacote ggplot2
toPlot = summarise(group_by(samples, sample_id), mean = mean(arr_delay))
toPlot = mutate(toPlot, lower = ifelse(toPlot$sample_id == 1,ic_1[1],ic_2[1]))
toPlot = mutate(toPlot, upper = ifelse(toPlot$sample_id == 1,ic_1[2],ic_2[2]))
ggplot(toPlot, aes(x = sample_id, y=mean, colour = sample_id )) +
  geom_point() +
  geom_errorbar(aes(ymin=lower, ymax=upper), width=.1)


# Exercício 7 - Podemos dizer que muito provavelmente, as amostras vieram da mesma população?
# Por que?
# Sim. A maior parte dos dados reside no mesmo intervalo de confiança nas duas amostras.


# Exercício 8 - Crie um teste de hipótese para verificar se os voos da Delta Airlines (DL)
# atrasam mais do que os voos da UA (United Airlines)

# H0 e H1 devem ser mutuamente exclusivas.

# H0 = Não há diferença significativa entre os atrasos da DL e UA (diff da média de atrasos = 0).
# H1 = Delta atrasa mais (diff das médias > 0).


# Cria as amostras
dl <- sample_n(filter(pop_data, carrier == "DL", arr_delay > 0), 1000)
ua <- sample_n(filter(pop_data, carrier == "UA", arr_delay > 0), 1000)

# Calcula erro padrão e média
se = sd(dl$arr_delay) / sqrt(nrow(dl))
mean(dl$arr_delay)

# Limites inferior e superior
lower = mean(dl$arr_delay) - 1.96 * se
upper = mean(dl$arr_delay) + 1.96 * se
ic_dl = c(lower,upper)
ic_dl

# Repete o processo para a outra companhia
se = sd(ua$arr_delay) / sqrt(nrow(ua))
mean(ua$arr_delay)

lower = mean(ua$arr_delay) - 1.96 * se
upper = mean(ua$arr_delay) + 1.96 * se
ic_ua = c(lower,upper)
ic_ua

# Teste t
# O teste t (de Student) foi desenvolvido por Willian Sealy Gosset em 1908 que usou o
# pseudônimo “Student” em função da confidencialidade requerida por seu empregador
# (cervejaria Guiness) que considerava o uso de estatística na manutenção da qualidade como
# uma vantagem competitiva.
# O teste t de Student tem diversas variações de aplicação, e pode ser usado na comparação
# de duas (e somente duas) médias e as variações dizem respeito às hipóteses que são testadas

t.test(dl$arr_delay, ua$arr_delay, alternative="greater")


# Valor p
# O valor-p é uma quantificação da probabilidade de se errar ao rejeitar H0 e a mesma
# decorre da distribuição estatística adotada.
# Se o valor-p é menor que o nível de significância, conclui-se que o correto é rejeitar a
```

```
# hipótese de nulidade.

# Valor p é a probabiblidade de que a estatística do teste assuma um valor extremo em relação
# ao valor observado quando H0 é verdadeira.

# Estamos trabalhando com alfa igual a 0.05 (95% de confiança)

# Regra
# Baixo valor p: forte evidência empírica contra h0
# Alto valor p: pouca ou nenhuma evidência empírica contra h0

# Falhamos em rejeitar a hipótese nula, pois p-valor é maior que o nível de significância
# Isso que dizer que há uma probabilidade alta de não haver diferença significativa entre os atrasos.
# Para os nossos dados, não há evidência estatística de que a DL atrase mais que a UA.

# Análise Exploratória de Dados (Template para Ponto de Partida)

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap11")
getwd()


# Carregando o pacote readr
library(readr)

# Carregando o dataset
carros <- read_csv("carros-usados.csv")

# Resumo dos dados
View(carros)
str(carros) # resumo das variáveis

# Medidas de Tendência Central
summary(carros$ano)
summary(carros[c('preco', 'kilometragem')])


##### Análise Exploratória de Dados Para Variáveis Numéricas #####

# Usando as funções
mean(carros$preco)
median(carros$preco)
quantile(carros$preco)
quantile(carros$preco, probs = c(0.01, 0.99))
quantile(carros$preco, seq( from = 0, to = 1, by = 0.20))
IQR(carros$preco) #Diferença entre Q3 e Q1
range(carros$preco)
diff(range(carros$preco))

# Plot

# Boxplot
# Leitura de Baixo para Cima - Q1, Q2 (Mediana) e Q3
boxplot(carros$preco, main = "Boxplot para os Preços de Carros Usados", ylab = "Preço (R$)")
boxplot(carros$kilometragem, main = "Boxplot para a Km de Carros Usados", ylab = "Kilometragem (R$)")

# Histograma
# Indicam a frequência de valores dentro de cada bin (classe de valores)
```

```
hist(carros$preco, main = "Histograma para os Preços Carros Usados", xlab = "Preço (R$)")
hist(carros$kilometragem, main = "Histograma para a Km de Carros Usados", ylab = "Kilometragem (R$)")
hist(carros$kilometragem, main = "Histograma para a Km de Carros Usados", breaks = 5, ylab = "Kilometragem (R$)")

# Scatterplot Preço x Km
# Usando o preço como variável dependente (y)
plot(x = carros$kilometragem, y = carros$preco,
     main = "Scatterplot - Preço x Km",
     xlab = "Kilometragem",
     ylab = "Preço (R$)")

# Medidas de Dispersão
# Ao interpretar a variância, números maiores indicam que
# os dados estão espalhados mais amplamente em torno da
# média. O desvio padrão indica, em média, a quantidade
# de cada valor diferente da média.
var(carros$preco)
sd(carros$preco)
var(carros$kilometragem)
sd(carros$kilometragem)


##### Análise Exploratória de Dados Para Variáveis Categóricas #####

# Criando tabelas de contingência - representam uma única variável categórica
# Lista as categorias das variáveis nominais
?table
str(carros)
table(carros$cor) # calcula as ocorrencias para cada variavel categorica
table(carros$modelo)
str(carros)

# Calculando a proporção de cada categoria
model_table <- table(carros$modelo) # monta a tabela de contingência
prop.table(model_table) # calcula a proporção com a função prop

# Arrendondando os valores
model_table <- table(carros$modelo)
model_table <- prop.table(model_table) * 100
round(model_table, digits = 1)

# Criando uma nova variável indicando cores conservadoras
# (que as pessoas compram com mais frequência)
head(carros)
carros$conserv <- carros$cor %in% c("Preto", "Cinza", "Prata", "Branco") # criando uma nova coluna com o retorno da
funcao (true ou false)
head(carros)

# Checando a variável
table(carros$conserv)

# Verificando o relacionamento entre 2 variáveis categóricas
# Criando uma crosstable
# Tabelas de contingência fornecem uma maneira de exibir
# as frequências e frequências relativas de observações
# (lembra do capítulo de Estatística?), que são classificados
# de acordo com duas variáveis categóricas. Os elementos de
# uma categoria são exibidas através das colunas;
# os elementos de outra categoria são exibidas sobre as linhas.
install.packages("gmodels")
library(gmodels)
?CrossTable
CrossTable(x = carros$modelo, y = carros$conserv)
```

Teste do Qui-quadrado

Qui Quadrado, simbolizado por χ2 é um teste de
hipóteses que se destina a encontrar um valor da
dispersão para duas variáveis nominais, avaliando a
associação existente entre variáveis qualitativas.

É um teste não paramétrico, ou seja, não depende dos
parâmetros populacionais, como média e variância.

O princípio básico deste método é comparar proporções,
isto é, as possíveis divergências entre as frequências
observadas e esperadas para um certo evento.
Evidentemente, pode-se dizer que dois grupos se
comportam de forma semelhante se as diferenças entre
as frequências observadas e as esperadas em cada
categoria forem muito pequenas, próximas a zero.

Ou seja, Se a probabilidade é muito baixa, ele fornece
fortes evidências de que as duas variáveis estão
associadas.

CrossTable(x = carros$modelo, y = carros$conserv, chisq = TRUE) # chisq é o qui-quadrado
chisq.test(x = carros$modelo, y = carros$conserv)

Trabalhamos com 2 hipóteses:
Hipótese nula: As frequências observadas não são diferentes das frequências esperadas.
Não existe diferença entre as frequências (contagens) dos grupos.
Portanto, não há associação entre os grupos

Hipótese alternativa: As frequências observadas são diferentes das frequências esperadas,
portanto existe diferença entre as frequências.
Portanto, há associação entre os grupos.

Neste caso, o valor do Chi = 0.15
E graus de liberdade (df) = 2
Portanto, não há associação entre os grupos
O valor alto do p-value confirma esta conclusão.

Prevendo a Ocorrência de Câncer

Obs: Caso tenha problemas com a acentuação, consulte este link:
https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap11/Classificacao")
getwd()

Definição do Problema de Negócio: Previsão de Ocorrência de Câncer de Mama
http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

Etapa 1 - Coletando os Dados

Os dados do câncer da mama incluem 569 observações de biópsias de câncer,
cada um com 32 características (variáveis). Uma característica é um número de
identificação (ID), outro é o diagnóstico de câncer, e 30 são medidas laboratoriais
numéricas. O diagnóstico é codificado como "M" para indicar maligno ou "B" para

```
# indicar benigno.
dados <- read.csv("dataset.csv", stringsAsFactors = FALSE) # nao considera variavel numerica como tipo fator
str(dados)
View(dados)


## Etapa 2 - Pré-Processamento

# Excluindo a coluna ID
# Independentemente do método de aprendizagem de máquina, deve sempre ser excluídas
# variáveis de ID. Caso contrário, isso pode levar a resultados errados porque o ID
# pode ser usado para unicamente "prever" cada exemplo. Por conseguinte, um modelo
# que inclui um identificador pode sofrer de superajuste (overfitting),
# e será muito difícil usá-lo para generalizar outros dados.
dados$id = NULL

# Ajustando o label da variável alvo
dados$diagnosis = sapply(dados$diagnosis, function(x){ifelse(x=='M', 'Maligno', 'Benigno')})

# Muitos classificadores requerem que as variáveis sejam do tipo Fator
table(dados$diagnosis)
dados$diagnosis <- factor(dados$diagnosis, levels = c("Benigno", "Maligno"), labels = c("Benigno", "Maligno"))
str(dados$diagnosis)

# Verificando a proporção
round(prop.table(table(dados$diagnosis)) * 100, digits = 1)

# Medidas de Tendência Central
# Detectamos um problema de escala entre os dados, que então precisam ser normalizados
# O cálculo de distância feito pelo kNN é dependente das medidas de escala nos dados de entrada.
summary(dados[c("radius_mean", "area_mean", "smoothness_mean")])

# Criando um função de normalização - colocando variáveis numéricas na mesma escala
normalizar <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

# Testando a função de normalização - os resultados devem ser idênticos
normalizar(c(1, 2, 3, 4, 5))
normalizar(c(10, 20, 30, 40, 50))

# Normalizando os dados
dados_norm <- as.data.frame(lapply(dados[2:31], normalizar)) # coluna 1 é a target e é categórica
View(dados_norm)


## Etapa 3: Treinando o modelo com KNN - vizinho mais próximo

# Carregando o pacote library
# install.packages("class")
library(class)
?knn

# Criando dados de treino e dados de teste
dados_treino <- dados_norm[1:469, ]
dados_teste <- dados_norm[470:569, ]

# Criando os labels para os dados de treino e de teste
dados_treino_labels <- dados[1:469, 1]
dados_teste_labels <- dados[470:569, 1]
length(dados_treino_labels)
length(dados_teste_labels)
```

```
# Criando o modelo
modelo_knn_v1 <- knn(train = dados_treino,
                     test = dados_teste,
                     cl = dados_treino_labels,
                     k = 21) # vai olhar os 21 dados mais próximos para cada dado - distância euclidiana

# A função knn() retorna um objeto do tipo fator com as previsões para cada exemplo no dataset de teste
summary(modelo_knn_v1)


## Etapa 4: Avaliando e Interpretando o Modelo

# Carregando o gmodels
library(gmodels)

# Criando uma tabela cruzada dos dados previstos x dados atuais
# Usaremos amostra com 100 observações: length(dados_teste_labels)
CrossTable(x = dados_teste_labels, y = modelo_knn_v1, prop.chisq = FALSE)

# Interpretando os Resultados
# A tabela cruzada mostra 4 possíveis valores, que representam os falso/verdadeiro positivo e negativo
# Temos duas colunas listando os labels originais nos dados observados
# Temos duas linhas listando os labels dos dados de teste

# Temos:
# Cenário 1: Célula Benigno (Observado) x Benigno (Previsto) - 61 casos - true positive
# Cenário 2: Célula Maligno (Observado) x Benigno (Previsto) - 00 casos - false positive (o modelo errou)
# Cenário 3: Célula Benigno (Observado) x Maligno (Previsto) - 02 casos - false negative (o modelo errou)
# Cenário 4: Célula Maligno (Observado) x Maligno (Previsto) - 37 casos - true negative

# Lendo a Confusion Matrix (Perspectiva de ter ou não a doença):

# True Negative  = nosso modelo previu que a pessoa NÃO tinha a doença e os dados mostraram que realmente a pessoa NÃO tinha a doença
# False Positive = nosso modelo previu que a pessoa tinha a doença e os dados mostraram que NÃO, a pessoa tinha a doença
# False Negative = nosso modelo previu que a pessoa NÃO tinha a doença e os dados mostraram que SIM, a pessoa tinha a doença
# True Positive = nosso modelo previu que a pessoa tinha a doença e os dados mostraram que SIM, a pessoa tinha a doença

# Falso Positivo - Erro Tipo I
# Falso Negativo - Erro Tipo II

# Taxa de acerto do Modelo: 98% (acertou 98 em 100)


## Etapa 5: Otimizando a Performance do Modelo

# Usando a função scale() para padronizar o z-score
?scale()
dados_z <- as.data.frame(scale(dados[-1])) # -1 retira a coluna da variável alvo
# embora o nome seja escala, trata-se de uma normalização

# Confirmando transformação realizada com sucesso
summary(dados_z$area_mean)

# Criando novos datasets de treino e de teste
dados_treino <- dados_z[1:469, ]
dados_teste <- dados_z[470:569, ]

dados_treino_labels <- dados[ 1: 469, 1]
dados_teste_labels <- dados[ 470: 569, 1]
```

```
# Reclassificando
modelo_knn_v2 <- knn(train = dados_treino,
                     test = dados_teste,
                     cl = dados_treino_labels,
                     k = 21)

# Criando uma tabela cruzada dos dados previstos x dados atuais
CrossTable(x = dados_teste_labels, y = modelo_knn_v2, prop.chisq = FALSE)

# Experimente diferentes valores para k


# Etapa 6: Construindo um Modelo com Algoritmo Support Vector Machine (SVM)

# Definindo a semente para resultados reproduzíveis
set.seed(40)

# Prepara o dataset
dados <- read.csv("dataset.csv", stringsAsFactors = FALSE)
dados$id = NULL
dados['index'] <- ifelse(runif(nrow(dados)) < 0.8,1,0) # criação randômica de índices
View(dados)

# Dados de treino e teste
trainset <- dados[dados$index==1,]
testset <- dados[dados$index==0,]

# Obter o índice
trainColNum <- grep('index', names(trainset))

# Remover o índice dos datasets
trainset <- trainset[,-trainColNum]
testset <- testset[,-trainColNum]

# Obter índice de coluna da variável target no conjunto de dados
typeColNum <- grep('diag',names(dados))

# Cria o modelo
# Nós ajustamos o kernel para radial, já que este conjunto de dados não tem um
# plano linear que pode ser desenhado
library(e1071)
?svm
modelo_svm_v1 <- svm(diagnosis ~ ., # var target ~ todas as outras
                     data = trainset,
                     type = 'C-classification',
                     kernel = 'radial')


# Previsões

# Previsões nos dados de treino
pred_train <- predict(modelo_svm_v1, trainset)

# Percentual de previsões corretas com dataset de treino
mean(pred_train == trainset$diagnosis)


# Previsões nos dados de teste
pred_test <- predict(modelo_svm_v1, testset)

# Percentual de previsões corretas com dataset de teste
mean(pred_test == testset$diagnosis)
```

```
# Confusion Matrix
table(pred_test, testset$diagnosis)


# Etapa 7: Construindo um Modelo com Algoritmo Random Forest

# Criando o modelo
library(rpart)
modelo_rf_v1 = rpart(diagnosis ~ ., data = trainset, control = rpart.control(cp = .0005))

# Previsões nos dados de teste
tree_pred = predict(modelo_rf_v1, testset, type='class')

# Percentual de previsões corretas com dataset de teste
mean(tree_pred==testset$diagnosis)

# Confusion Matrix
table(tree_pred, testset$diagnosis)

# Machine Learning - Regressão
# Prevendo Despesas Hospitalares

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap11/
Regressao")
getwd()

# Problema de Negócio: Previsão de Despesas Hospitalares

# Para esta análise, vamos usar um conjunto de dados simulando despesas médicas hipotéticas
# para um conjunto de pacientes espalhados por 4 regiões do Brasil.
# Esse dataset possui 1.338 observações e 7 variáveis.

# Etapa 1 - Coletando os dados
despesas <- read.csv("despesas.csv")
View(despesas)

# Etapa 2: Explorando e Preparando os Dados
# Visualizando as variáveis e seus tipos
str(despesas)

# Medias de Tendência Central da variável gastos (variável target)
summary(despesas$gastos)
# quando media e mediana são diferentes, muito provável NÃO se tratar de uma
# distribuição normal.

# Construindo um histograma
hist(despesas$gastos, main = 'Histograma', xlab = 'Gastos')

# Tabela de contingência das regiões
table(despesas$regiao)

# Explorando relacionamento entre as variáveis: Matriz de Correlação
cor(despesas[c("idade", "bmi", "filhos", "gastos")]) # coeficiente de correlação

# Nenhuma das correlações na matriz é considerada forte, mas existem algumas associações interessantes.
# Por exemplo, a idade e o bmi (IMC) parecem ter uma correlação positiva fraca, o que significa que
# com o aumento da idade, a massa corporal tende a aumentar. Há também uma correlação positiva
```

```
# moderada entre a idade e os gastos, além do número de filhos e os gastos. Estas associações implicam
# que, à media que idade, massa corporal e número de filhos aumenta, o custo esperado do seguro saúde sobe.

# Visualizando relacionamento entre as variáveis: Scatterplot
# Perceba que não existe um claro relacionamento entre as variáveis
pairs(despesas[c("idade", "bmi", "filhos", "gastos")])
# pair - conjunto de graficos scatterplot (de dispersão), na mesma área de plotagem.

# Scatterplot Matrix
install.packages("psych")
library(psych)

# Este gráfico fornece mais informações sobre o relacionamento entre as variáveis
pairs.panels(despesas[c("idade", "bmi", "filhos", "gastos")])

# Etapa 3: Treinando o Modelo (usando os dados de treino)
?lm # função para construção de um modelo linear
modelo <- lm(gastos ~ idade + filhos + bmi + sexo + fumante + regiao, data = despesas)
# do lado esquerdo do "~", temos a variável target. do direito, as preditoras

# Similar ao item anterior
modelo <- lm(gastos ~ ., data = despesas)
# já o "." significa utilizar todas as variáveis como preditoras (menos a target)

# Visualizando os coeficientes
modelo

# Prevendo despesas médicas
?predict

# Aqui verificamos os gastos previstos pelo modelo que devem ser iguais aos dados de treino
previsao1 <- predict(modelo) # previsão usando o modelo de machine learning
View(previsao1)

# Prevendo os gastos com Dados de teste
despesasteste <- read.csv("despesas-teste.csv")
View(despesasteste)
previsao2 <- predict(modelo, despesasteste)
View(previsao2)



# Etapa 4: Avaliando a Performance do Modelo
# Mais detalhes sobre o modelo
summary(modelo)


# ************************************************
# *** Estas informações abaixo é que farão de você ***
# *** um verdadeiro conhecedor de Machine Learning ***
# ************************************************

# Equação de Regressão
# y = a + bx (simples)
# y = a + b0x0 + b1x1 (múltipla)

# Resíduos (Residuals)
# Diferença entre os valores observados de uma variável e seus valores previstos
# Seus resíduos devem se parecer com uma distribuição normal, o que indica
# que a média entre os valores previstos e os valores observados é próximo de 0 (o que é bom)

# Coeficiente (Coefficients) - Intercept - a (alfa)(ponto onde a reta corta o eixo y)
# Valor de a na equação de regressão
```

Coeficientes - Nomes das variáveis - b (beta)
Valor de b na equação de regressão

Obs: A questão é que lm() ou summary() têm diferentes convenções de
rotulagem para cada variável explicativa.
Em vez de escrever slope_1, slope_2, ...
Eles simplesmente usam o nome da variável em qualquer saída para
indicar quais coeficientes pertencem a qual variável.

Erro Padrão (Std. Error)
Medida de variabilidade na estimativa do coeficiente a (alfa). O ideal é que este valor
seja menor que o valor do coeficiente, mas nem sempre isso irá ocorrer.

Asteriscos
Os asteriscos representam os níveis de significância de acordo com o p-value.
Quanto mais estrelas, maior a significância.
Atenção --> Muitos astericos indicam que é improvável que não exista
relacionamento entre as variáveis.

Valor t (t value)
Define se coeficiente da variável é significativo ou não para o modelo.
Ele é usado para calcular o p-value e os níveis de significância.

p-value
O p-value representa a probabilidade que a variável não seja relevante.
Deve ser o menor valor possível.
Se este valor for realmente pequeno, o R irá mostrar o valor
como notação científica (ex. 2e-16)

Significância (Signif. code)
São aquelas legendas próximas as suas variáveis
Espaço em branco - ruim
Pontos - razoável
Asteriscos - bom
Muitos asteriscos - muito bom

Residual Standard Error (desvio padrão)
Este valor representa o desvio padrão dos resíduos

Degrees of Freedom (grau de liberdade)
É a diferença entre o número de observações na amostra de treinamento
e o número de variáveis no seu modelo

R-squared (coeficiente de determinação - R^2 ou R ao quadrado)
Ajuda a avaliar o nível de precisão do nosso modelo.
Quanto maior, melhor, sendo 1 o valor ideal.

F-statistics (estatística F)
É o teste F do modelo. Esse teste obtém os parâmetros do nosso modelo
e compara com um modelo que tenha menos parâmetros.
Em teoria, um modelo com mais parâmetros tem um desempenho melhor.

Se o seu modelo com mais parâmetros NÃO tiver perfomance
melhor que um modelo com menos parâmetros, o valor do p-value será bem alto.

Se o modelo com mais parâmetros tiver performance
melhor que um modelo com menos parâmetros, o valor do p-value será mais baixo.

Lembre-se que correlação não implica causalidade

```
# Etapa 5: Otimizando a Performance do Modelo

# Adicionando uma variável com o dobro do valor das idades
despesas$idade2 <- despesas$idade ^ 2

# Adicionando um indicador para BMI >= 30
despesas$bmi30 <- ifelse(despesas$bmi >= 30, 1, 0)

View(despesas)

# Criando o modelo final
modelo_v2 <- lm(gastos ~ idade + idade2 + filhos + bmi + sexo +
                bmi30 * fumante + regiao, data = despesas)

summary(modelo_v2)

# Dados de teste
despesasteste <- read.csv("despesas-teste.csv")
View(despesasteste)
previsao <- predict(modelo, despesasteste)
class(previsao)
View(previsao)
```

Capítulo 12:

```
# Solução Lista de Exercícios Parte 1 - Capítulo 11

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap12")
getwd()


## Exercício 1 - Massa de dados aleatória

# Criando a massa de dados (apesar de aleatória, y possui
# uma relação com os dados de x)
x <- seq(0, 100)
y <- 2 * x + 35

# Imprimindo as variáveis
x
y

# Gerando uma distribuição normal
y1 <- y + rnorm(101, 0, 50)
y1
hist(y1)

# Crie um plot do relacionamento de x e y1
plot(x, y1, pch = 19, xlab = 'X', ylab = 'Y')

# Crie um modelo de regressão para as duas variáveis x e y1
modelo <- lm(y1 ~ x)
modelo
class(modelo)

# Capture os coeficentes
a <- modelo$coefficients[1]
b <- modelo$coefficients[2]

# Fórmula de Regressão
y2 <- a + b*x

# Visualize a linha de regressão
lines(x, y2, lwd = 2)

# Simulando outras possíveis linhas de regressão
y3 <- (y2[51]-50*(b-1))+(b-1)*x
y4 <- (y2[51]-50*(b+1))+(b+1)*x
y5 <- (y2[51]-50*(b+2))+(b+2)*x
lines(x,y3,lty=3)
lines(x,y4,lty=3)
lines(x,y5,lty=3)


## Exercício 2 - Pesquisa sobre idade e tempo de reação

# Criando os dados
Idade <- c(9,13,14,21,15,18,20,8,14,23,16,21,10,12,20,
           9,13,5,15,21)
```

```
Tempo <- c(17.87,13.75,12.72,6.98,11.01,10.48,10.19,19.11,
           12.72,0.45,10.67,1.59,14.91,14.14,9.40,16.23,
           12.74,20.64,12.34,6.44)

# Crie um Gráfico de Dispersão (ScatterPlot)
plot(Idade, Tempo,
     xlab = 'Idade',
     ylab = 'Tempo de Reação')

# Crie um modelo de regressão
modelo <- lm(Tempo ~ Idade)
modelo

# Calcule a reta de regressão
# y <- a + b*x
reta <- 25.8134 - 0.9491 * Idade # usa-se subtração por que essa correlação é negativa

# Crie o gráfico da reta
lines(Idade,reta)


# Exercício 3 - Relação entre altura e peso

# Criando os dados
alturas = c(176, 154, 138, 196, 132, 176, 181, 169, 150, 175)
pesos = c(82, 49, 53, 112, 47, 69, 77, 71, 62, 78)

plot(alturas, pesos, pch = 16, cex = 1.3, col = "blue",
     main = "Altura x Peso",
     ylab = "Peso Corporal (kg)",
     xlab = "Altura (cm)")

# Crie o modelo de regressão
modelo <- lm(pesos ~ alturas)

# Visualizando o modelo
modelo
summary(modelo)

# Gere a linha de regressão
abline(-70.4627, 0.8528) # esses coeficentes (a e b) estão visíveis em modelo

# Faça as previsões de pesos com base na nova lista de alturas
alturas2 = data.frame(c(179, 152, 134, 197, 131, 178, 185, 162, 155, 172))
previsao <- predict(modelo, alturas2)
previsao

# Plot
plot(alturas, pesos, pch = 16, cex = 1.3,
     col = "blue",
     main = "Altura x Peso",
     ylab = "Peso (kg)",
     xlab = "Altura (cm)")

# Construindo a linha de regressão - passando a montagem do modelo como parâmetro
abline(lm(pesos ~ alturas))

# Obtendo o tamanho de uma das amostras de dados
num <- length(alturas)
num

# Gerando um gráfico com os valores residuais (erros do modelo)
```

```
for (k in 1: num)
  lines(c(alturas[k], alturas[k]),
        c(pesos[k], pesos[k]))

# Gerando gráficos com a distribuição dos resíduos
par(mfrow = c(2,2)) # dividindo a área de plotagem numa matriz 2x2
plot(modelo)

# Solução Lista de Exercícios Parte 2 - Capítulo 11

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap12")
getwd()


# Regressão Linear
# Definição do Problema: Prever as notas dos alunos com base em diversas métricas
# https://archive.ics.uci.edu/ml/datasets/Student+Performance
# Dataset com dados de estudantes
# Vamos prever a nota final (grade) dos alunos

# Carregando o dataset
df <- read.csv2('estudantes.csv')
View(df)

# Explorando os dados
summary(df)
str(df) # tipos de dados
any(is.na(df)) # tem valor missing?

# install.packages("ggplot2")
# install.packages("ggthemes")
# install.packages("dplyr")
library(ggplot2)
library(ggthemes)
library(dplyr)

# Obtendo apenas as colunas numéricas
colunas_numericas <- sapply(df, is.numeric)
colunas_numericas

# Filtrando as colunas numéricas para correlação
data_cor <- cor(df[,colunas_numericas])
data_cor
head(data_cor)

# Pacotes para visualizar a análise de correlação
# install.packages('corrgram')
# install.packages('corrplot')
library(corrplot)
library(corrgram)

# Criando um corrplot
corrplot(data_cor, method = 'color')

# Criando um corrgram (uma evolução do corrplot)
corrgram(df)
corrgram(df, order = TRUE, lower.panel = panel.shade,
```

```
        upper.panel = panel.pie, text.panel = panel.txt)

# Criando um histograma
ggplot(df, aes(x = G3)) +
  geom_histogram(bins = 20,
                 alpha = 0.5, fill = 'blue') +
  theme_minimal()


# Treinando e Interpretando o Modelo
# Import Library
install.packages("caTools")
library(caTools)

# Criando as amostras de forma randômica
set.seed(101)
?sample.split # divide os dados em treino e teste
amostra <- sample.split(df$age, SplitRatio = 0.70) # usando age como um índice

# ***** Treinamos nosso modelo nos dados de treino *****
# *****   Fazemos as predições nos dados de teste  *****

# Criando dados de treino - 70% dos dados
treino = subset(df, amostra == TRUE)

# Criando dados de teste - 30% dos dados
teste = subset(df, amostra == FALSE)

# Gerando o Modelo (Usando todos os atributos)
modelo_v1 <- lm(G3 ~ ., treino)
modelo_v2 <- lm(G3 ~ G2 + G1, treino)
modelo_v3 <- lm(G3 ~ absences, treino)
modelo_v4 <- lm(G3 ~ Medu, treino)

# Interpretando o Modelo
summary(modelo_v1) # 0.86
summary(modelo_v2) # 0.82
summary(modelo_v3) # 0.0002675
summary(modelo_v4) # 0.06442


# Visualizando o Modelo e Fazendo Previsões

# Obtendo os resíduos
res <- residuals(modelo_v1)

# Convertendo o objeto para um dataframe
res <- as.data.frame(res)
head(res)

# Histograma dos resíduos
ggplot(res, aes(res)) +
  geom_histogram(fill = 'blue',
                 alpha = 0.5,
                 binwidth = 1)

# Plot do Modelo
plot(modelo_v1)

# Fazendo as predições
modelo_v1 <- lm(G3 ~ ., treino) # recriando o modelo (não há necessidade)
prevendo_G3 <- predict(modelo_v1, teste)
prevendo_G3
```

```
# Visualizando os valores previstos e observados (previsão e real)
resultados <- cbind(prevendo_G3, teste$G3)
colnames(resultados) <- c('Previsto','Real')
resultados <- as.data.frame(resultados)
resultados
min(resultados)

# Tratando os valores negativos
trata_zero <- function(x){
  if  (x < 0){
    return(0)
  }else{
    return(x)
  }
}

# Aplicando a função para tratar valores negativos em nossa previsão
resultados$Previsto <- sapply(resultados$Previsto, trata_zero)
resultados$Previsto

# Calculando o erro médio
# Quão distantes seus valores previstos estão dos valores observados
# MSE
mse <- mean((resultados$Real - resultados$Previsto)^2)
print(mse)

# RMSE
rmse <- mse^0.5
rmse

# Calculando R Squared
SSE = sum((resultados$Previsto - resultados$Real)^2)
SST = sum((mean(df$G3) - resultados$Real)^2)

# R-Squared
# Ajuda a avaliar o nível de precisão do nosso modelo. Quanto maior, melhor, sendo 1 o valor ideal.
R2 = 1 - (SSE/SST)
R2

# Solução Lista de Exercícios Parte 3 - Capítulo 11

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap12")
getwd()


# Definindo o Problema: Analisando dados das casas de Boston, nos EUA e fazendo previsoes.

# The Boston Housing Dataset
# http://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html

# Seu modelo deve prever a MEDV (Valor da Mediana de ocupação das casas).
# Utilize um modelo de rede neural!

# Carregando o pacote MASS
library(MASS)
```

```
# Importando os dados do dataset Boston
set.seed(101)
dados <- Boston
View(dados)

# Resumo dos dados
str(dados)
summary(dados)
any(is.na(dados))

# Carregando o pacote para Redes Neurais
install.packages("neuralnet")
library(neuralnet)

# Como primeiro passo, vamos abordar o pré-processamento de dados.
# É uma boa pratica normalizar seus dados antes de treinar uma rede neural.
# Dependendo do seu conjunto de dados, evitando a normalizacao pode levar a
# resultados inuteis ou a um processo de treinamento muito dificil
# (na maioria das vezes o algoritmo não ira convergir antes do numero de iteracoes
# maximo permitido). Voce pode escolher diferentes metodos para dimensionar os
# dados (normalizacao-z, escala min-max, etc ...).
# Normalmente escala nos intervalos [0,1] ou [1,1] tende a dar melhores resultados.

# Normalizacao
maxs <- apply(dados, 2, max)
mins <- apply(dados, 2, min)

# Imprimindo os valores
maxs
mins

# Normalizando
dados_normalizados <- as.data.frame(scale(dados, center = mins, scale = maxs - mins))
head(dados_normalizados)

# Criando os dados de treino e de teste
install.packages("caTools")
library(caTools)
split = sample.split(dados_normalizados$medv, SplitRatio = 0.70)

treino = subset(dados_normalizados, split == TRUE)
teste = subset(dados_normalizados, split == FALSE)

# Obtendo o nome das colunas
coluna_nomes <- names(treino)
coluna_nomes

# Agregando
formula <- as.formula(paste("medv ~", paste(coluna_nomes[!coluna_nomes %in% "medv"], collapse = " + ")))
# aqui foi usado programação, adicionando à string o nome das colunas preditoras
formula

# Treinando o Modelo
rede_neural <- neuralnet(formula, data = treino, hidden = c(5,3), linear.output = TRUE)

# Plot
plot(rede_neural)

# Fazendo previsoes com os dados de teste
rede_neural_prev <- compute(rede_neural, teste[1:13])
rede_neural_prev # previsão com dados normalizados

# O retorno da previsao da Rede Neural é uma lista
```

```
str(rede_neural_prev)

# Convertendo os dados de teste (desnormalizando)
previsoes <- rede_neural_prev$net.result * (max(dados$medv) - min(dados$medv)) + min(dados$medv)
teste_convert <- (teste$medv) * (max(dados$medv) - min(dados$medv)) + min(dados$medv)
teste_convert

# Calculando o Mean Squared Error (medida de erro para regressões)
MSE.nn <- sum((teste_convert - previsoes)^2)/nrow(teste)
MSE.nn

# Obtendo os erros de previsao
error.df <- data.frame(teste_convert, previsoes)
head(error.df)

# Plot dos erros
library(ggplot2)
ggplot(error.df, aes(x = teste_convert,y = previsoes)) +
  geom_point() + stat_smooth()

# Solução Lista de Exercícios Parte 4 - Capítulo 11

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap12")
getwd()


# Definindo o Problema: OCR - Optical Character Recognition
# Seu modelo deve prever o caracter a partir do dataset fornecido. Use um modelo SVM

## Explorando e preparando os dados
letters <- read.csv("letterdata.csv")
View(letters)
str(letters)

# Criando dados de treino e dados de teste
letters_treino <- letters[1:16000, ]
letters_teste  <- letters[16001:20000, ]

## Treinando o Modelo
install.packages("kernlab")
library(kernlab) # pacote que possui a função ksvm

# Criando o modelo com o kernel vanilladot
letter_classifier <- ksvm(letter ~ ., data = letters_treino, kernel = "vanilladot")

# Visualizando resultado do modelo
letter_classifier

# Avaliando a performance do modelo
letter_predictions <- predict(letter_classifier, letters_teste)
head(letter_predictions)
table(letter_predictions, letters_teste$letter)

# Criando um vetor de TRUE/FALSE indicando previsoes corretas/incorretas
agreement <- letter_predictions == letters_teste$letter
table(agreement)
prop.table(table(agreement)) # colocando em proporção
```

```
## Otimizando o Modelo
set.seed(12345)

# Recriando o modelo com outro tipo de kernel
letter_classifier_rbf <- ksvm(letter ~ ., data = letters_treino, kernel = "rbfdot")

# Novas previsoes
letter_predictions_rbf <- predict(letter_classifier_rbf, letters_teste)

# Compare os resultados com a primeira versao do modelo
agreement_rbf <- letter_predictions_rbf == letters_teste$letter
table(agreement_rbf)
prop.table(table(agreement_rbf))

# Usando o Pacote Caret Para Criar Modelos de Machine Learning em R
# http://topepo.github.io/caret/index.html

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap12")
getwd()

# Instalando os pacotes
install.packages("caret")
install.packages("randomForest")

# Carregando os pacotes
library(caret)
library(randomForest)
library(datasets)

# Usando o dataset mtcars
View(mtcars)

# Funcao do Caret para divisao dos dados de treino e de teste
?createDataPartition
split <- createDataPartition(y = mtcars$mpg, p = 0.7, list = FALSE)

# Criando dados de treino e de teste
dados_treino <- mtcars[split,]
dados_teste <- mtcars[-split,]

# Treinando o modelo
?train
names(getModelInfo())

# Mostrando a importância das variáveis para a criação do modelo
?varImp

modelol_v1 <- train(mpg ~ ., data = dados_treino, method = "lm")
varImp(modelol_v1)

# Regressão linear
modelol_v1 <- train(mpg ~ wt + hp + qsec + drat, data = dados_treino, method = "lm")

# Random forest
modelol_v2 <- train(mpg ~ wt + hp + qsec + drat, data = dados_treino, method = "rf")
```

```
# Resumo do modelo
summary(modelol_v1)
summary(modelol_v2)

# Ajustando o modelo
?expand.grid
?trainControl
controle1 <- trainControl(method = "cv", number = 10)

modelol_v3 <- train(mpg ~ wt + hp + qsec + drat,
                    data = dados_treino,
                    method = "lm",
                    trControl = controle1,
                    metric = "Rsquared")

# Resumo do modelo
summary(modelol_v3)

# Coletando os residuos (erros de previsão do modelo)
residuals <- resid(modelol_v3)
residuals

# Previsoes
?predict
predictedValues <- predict(modelol_v1, dados_teste)
predictedValues
plot(dados_teste$mpg, predictedValues)

# Plot das variáveis mais relevantes no modelo
plot(varImp(modelol_v1))
```

Capítulo 13:

```
# Solução Lista de Exercícios - Capítulo 12

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap13")
getwd()

# Existem diversos pacotes para árvores de decisão em R. Usaremos aqui o rpart.
install.packages('rpart')
library(rpart)

# Vamos utilizar um dataset que é disponibilizado junto com o pacote rpart
str(kyphosis)
head(kyphosis)
View(kyphosis)
?kyphosis

# Exercício 1 - Depois de explorar o dataset, crie um modelo de árvore de decisão
?rpart # rpart pode ser utilizado tanto para modelos de regressão quanto classificação
arvore <- rpart(Kyphosis ~ . , method = 'class', data = kyphosis)
class(arvore)
arvore

# Para examinar o resultado de uma árvore de decisao, existem diversas funcões, mas você pode usar printcp()
printcp(arvore)

# Visualizando a ávore (execute uma função para o plot e outra para o texto no plot)
# Utilize o zoom para visualizar melhor o gráfico
plot(arvore, uniform = TRUE, main = "Arvore de Decisao em R")
text(arvore, use.n = TRUE, all = TRUE)

# Este outro pacote faz a visualizaco ficar mais legivel
install.packages('rpart.plot')
library(rpart.plot)
prp(arvore)

# Solução Lista de Exercícios - Capítulo 12

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap13")
getwd()

# Usaremos o dataset iris neste exemplo
# O dataset iris possui observações de 3 espécies de flores (Iris setosa, Iris virginica e Iris versicolor)
# Para cada flor, 4 medidas sao usadas:
# comprimento (length) e largura (width) do caule (sepal) e comprimento e largura da petala (petal)
library(datasets)
head(iris)
View(iris)

# Análise exploratória de dados com ggplot2
```

```
library(ggplot2)

# Veja que os dados claramente possui grupos com característcas similares
ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) +
  geom_point(size = 3) # gráfico de pontos de tamanho 3

# Agora usarmeos o K-Means para tentar agrupar os dados em clusters
set.seed(101)
help(kmeans)

# Exercício 1 - Usando a função kmeans(), crie um modelo de clustering (aprendizagem não supervisionada).
# Use a documentação, para fazer sua pesquisa.
# Neste caso, ja sabemos quantos grupos (clusters) existem em nossos dados (3)
# Perceba que o dataset iris possui 5 colunas, mas estamos usando as 4 primeiras
irisCluster <- kmeans(iris[, 1:4], 3, nstart = 20)
# 3 é a quantidade de clusters
# nstart é a quantidade de conjunto de dados randomicos para treinamento
irisCluster

# Obtendo informação sobre os clusters
# Foram criados 3 clusters: cluster 1, 2 e 3
# Perceba que apesar o algoritmo ter feito a divisão dos dados em clusters, houve problema em dividir alguns dos
dados,
# que apesar de terem caracteristicas diferentes, ficaram no mesmo cluster
table(irisCluster$cluster, iris$Species)
irisCluster

# Visualizando os clusters
install.packages("cluster")
library(cluster)
help(clusplot)

# Plot
clusplot(iris, irisCluster$cluster, color = TRUE, shade = TRUE, labels = 0, lines = 0 )

# Este código contém comandos para filtrar e plotar os dados de aluguel de bikes, dados que estão em nosso dataset
# Este código foi criado para executar tanto no Azure, quanto no RStudio. (RStudio ou Azure)
# Para executar no Azure, altere o valor da variavel Azure para TRUE. Se o valor for FALSE, o código sera executado
no RStudio

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap13")
getwd()

# Variável que controla a execução do script
Azure <- FALSE

# Execução de acordo com o valor da variável Azure
if(Azure){
  source("src/Tools.R")
  Bikes <- maml.mapInputPort(1) # porta 1 de entrada do módulo Execute R Script
  Bikes$dteday <- set.asPOSIXct(Bikes)
}else{
  source("Tools.R")
  Bikes <- read.csv("datasets/bikes.csv", sep = ",", header = T, stringsAsFactors = F )
  Bikes$dteday <- char.toPOSIXct(Bikes)
}
```

```
require(dplyr)
print("Dimensões do dataframe antes das operações de transformação:")
print(dim(Bikes))

# Filtrando o dataframe
Bikes <- Bikes %>% filter(cnt > 100)

print("Dimensões do dataframe após as operações de transformação:")
print(dim(Bikes))

# ggplot2
require(ggplot2)
qplot(dteday, cnt, data = subset(Bikes, hr == 9), geom = "line")

# Resultado
if(Azure) maml.mapOutputPort("Bikes")

# Este código contém comandos para filtrar e plotar os dados de aluguel de bikes,
# dados que estão em nosso dataset.
# Este código foi criado para executar tanto no Azure, quanto no RStudio.
# Para executar no Azure, altere o valor da variavel Azure para TRUE.
# Se o valor for FALSE, o codigo sera executado no RStudio.

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap13")
# getwd()

# Variável que controla a execução do script
Azure = FALSE

if(Azure){
  restaurantes <- maml.mapInputPort(1) # recupera dados da porta e cria um dataset
  ratings <- maml.mapInputPort(2)
}else{ # IMPORTANTE: antes, faça o download dos arquivos do azure no formato csv
  restaurantes  <- read.csv("datasets/Restaurant-features.csv", sep = ",", header = T, stringsAsFactors = F )
  ratings <- read.csv("datasets/Restaurant-ratings.csv", sep = ",", header = T, stringsAsFactors = F)
}

# Filtrando o dataset restaurantes por linha (código antes da vírgula nos colchetes)
restaurantes <- restaurantes[restaurantes$franchise == 'f' & restaurantes$alcohol != 'No_Alcohol_Served', ]
#

require(dplyr) # esse pacote já está disponível o azure ml

# Combinando os datasets com base em regras
df <- as.data.frame(restaurantes %>% # carrega dataset restaurantes
                    inner_join(ratings, by = 'placeID') %>% # inner_join (dplyr) une restaurantes e ratings pela coluna
placeID
                    select(name, rating) %>% # seleciona 2 colunas
                    group_by(name) %>% # agrupa pelo nome do restaurante
                    summarize(ave_Rating = mean(rating)) %>% # agrupa pela média do rating
                    arrange(desc(ave_Rating))) # imprime o resultado em ordem decrescente da média de avaliações
df

if(Azure) maml.mapOutputPort("df")

# Este código contém comandos para instalar pacotes no Azure ML
# Este codigo foi criado para executar tanto no Azure, quanto no RStudio.
```

```
# Para executar no Azure, altere o valor da variavel Azure para TRUE. Se o valor for FALSE, o codigo sera executado no RStudio.

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("C:/FCD/BigDataRAzure/Cap13")
# getwd()

# Variável que controla a execução do script
Azure <- TRUE

# IMPORTANTE: faça o download dos pacotes, conforme descrito em Instalando Pacotes R no Azure ML, no documento resumo do curso
if(Azure){
  # Instala o pacote tidyr e as dependências tibble e rlang a partir do arquivo zip
  install.packages("src/rlang_0.3.1.zip", lib = ".", repos = NULL, verbose = TRUE)
  install.packages("src/tibble_2.0.1.zip", lib = ".", repos = NULL, verbose = TRUE)
  install.packages("src/tidyr_0.8.2.zip", lib = ".", repos = NULL, verbose = TRUE)

  require(tibble, lib.loc = ".") # lib.loc = "." indica o diretório das bibliotecas da linguagem R no Azure
  require(rlang, lib.loc = ".")
  require(tidyr, lib.loc = ".")
}else{
  install.packages("tidyr")
  require(tidyr)
}

if(Azure) maml.mapOutputPort("dataset")

# Este código contém comandos para filtrar e plotar os dados de aluguel de bikes,
# dados que estao em nosso dataset
# Este codigo foi criado para executar tanto no Azure, quanto no RStudio.
# Para executar no Azure, altere o valor da variavel Azure para TRUE. Se o valor for FALSE, o codigo sera executado no RStudio.

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("C:/FCD/BigDataRAzure/Cap13")
# getwd()

# Variável que controla a execução do script
Azure <- FALSE

if(Azure){
  source("src/Tools.R") # arquivo de utilitários (deve ser transferido antes para o Azure)
  Bikes <- maml.mapInputPort(1) # carrega dataset na porta 1
  Bikes$dteday <- set.asPOSIXct(Bikes) # conversão de datas com função definida em Tools.R
}else{
  source("Tools.R")
  Bikes <- read.csv("datasets/bikes.csv", sep = ",", header = T, stringsAsFactors = F )
  Bikes$dteday <- char.toPOSIXct(Bikes)
}

require(dplyr)
Bikes <- Bikes %>% filter(hr == 9)
```

```
# ggplot2
require(ggplot2) # pacote já exite no Azure
ggplot(Bikes, aes(x = dteday, y = cnt)) +
  geom_line() + # gráfico de linha
  ylab("Numero de Bikes") +
  xlab("Linha do Tempo") +
  ggtitle("Demanda por Bikes as 09:00") +
  theme(text = element_text(size = 20))
```

Capítulo 14:

```
# Solução Lista de Exercícios - Capítulo 13

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
###setwd("C:/FCD/BigDataRAzure/Cap14")
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap14")
getwd()

# Para este script, vamos usar o mlbench (Machine Learning Benchmark Problems)
# https://cran.r-project.org/web/packages/mlbench/mlbench.pdf
# Este pacote contém diversos datasets e usaremos um com os dados
# de votação do congresso americano

# Seu trabalho é prever os votos em republicanos e democratas (variável Class)

# Import
install.packages("mlbench") # dataset HouseVotes84
library(mlbench)

# Carregando o dataset
?HouseVotes84
data("HouseVotes84")
View(HouseVotes84)

### ANALISE EXPLORATORIA DOS DADOS ###

# Analise exploratória de dados
plot(as.factor(HouseVotes84[,2])) # converter variavel em fator e colocar num plot
title(main = "Votes cast for issue", xlab = "vote", ylab = "# reps")
plot(as.factor(HouseVotes84[HouseVotes84$Class == 'republican', 2]))
title(main = "Republican votes cast for issue 1", xlab = "vote", ylab = "# reps")
plot(as.factor(HouseVotes84[HouseVotes84$Class == 'democrat',2]))
title(main = "Democrat votes cast for issue 1", xlab = "vote", ylab = "# reps")

### PRE-PROCESSAMENTO ###

# Funções usadas para imputation
# Função que retorna o numeros de NA's por voto e classe (democrat or republican)
na_by_col_class <- function (col,cls){return(sum(is.na(HouseVotes84[,col]) & HouseVotes84$Class==cls))}

p_y_col_class <- function(col,cls){
  sum_y <- sum(HouseVotes84[,col] == 'y' & HouseVotes84$Class == cls, na.rm = TRUE)
  sum_n <- sum(HouseVotes84[,col] == 'n' & HouseVotes84$Class == cls, na.rm = TRUE)
  return(sum_y/(sum_y+sum_n))}

# Testando a função
p_y_col_class(2,'democrat')
p_y_col_class(2,'republican')
na_by_col_class(2,'democrat') # verificando se ha valores missing
na_by_col_class(2,'republican')

# Impute missing values - uma de muitas tecnicas utilizadas para substituição de valores N/A
for (i in 2:ncol(HouseVotes84)) {
  if(sum(is.na(HouseVotes84[,i])>0)) {
    c1 <- which(is.na(HouseVotes84[,i]) & HouseVotes84$Class == 'democrat',arr.ind = TRUE)
    c2 <- which(is.na(HouseVotes84[,i]) & HouseVotes84$Class == 'republican',arr.ind = TRUE)
```

```
    HouseVotes84[c1,i] <- ifelse(runif(na_by_col_class(i,'democrat'))<p_y_col_class(i,'democrat'),'y','n')
    HouseVotes84[c2,i] <- ifelse(runif(na_by_col_class(i,'republican'))<p_y_col_class(i,'republican'),'y','n')}
}


### CONSTRUCAO DO MODELO ###

# Gerando dados de treino e dados de teste
HouseVotes84[,"train"] <- ifelse(runif(nrow(HouseVotes84)) < 0.80,1,0) # criando a regra para divisão dos dados
trainColNum <- grep("train",names(HouseVotes84)) # criando um índice para marcar dados de treino e teste

# Gerando os dados de treino e de teste a partir da coluna de treino
trainHouseVotes84 <- HouseVotes84[HouseVotes84$train == 1, -trainColNum]
testHouseVotes84 <- HouseVotes84[HouseVotes84$train == 0, -trainColNum]

# Invocando o método NaiveBayes
install.packages("e1071") # pacote necessário para o NaiveBayes
library(e1071)

### EXERCICIO ###

# Exercício 1 - Crie o modelo NaiveBayes

# Treine o modelo
?naiveBayes
nb_model <- naiveBayes(Class ~ ., data = trainHouseVotes84)
# Class é a variável target; . são todas as outras como preditoras

# Visualizando o resultado
nb_model
summary(nb_model)
str(nb_model)

# Faça as Previsões
nb_test_predict <- predict(nb_model, testHouseVotes84[,-1])

# Crie a Confusion matrix (tabela de comparação)
table(pred = nb_test_predict, true = testHouseVotes84$Class)

# Média (percentual de acerto do modelo)
mean(nb_test_predict == testHouseVotes84$Class)

### BONUS - TUDO NUM CÓDIGO SÓ: ###

# Função para executar o registrar todos os resultados do modelo
nb_multiple_runs <- function(train_fraction, n) {
  fraction_correct <- rep(NA,n)
  for (i in 1:n) {
    HouseVotes84[,"train"] <- ifelse(runif(nrow(HouseVotes84))<train_fraction,1,0)
    trainColNum <- grep("train", names(HouseVotes84))
    trainHouseVotes84 <- HouseVotes84[HouseVotes84$train == 1,-trainColNum]
    testHouseVotes84 <- HouseVotes84[HouseVotes84$train == 0,-trainColNum]
    nb_model <- naiveBayes(Class ~ ., data = trainHouseVotes84)
    nb_test_predict <- predict(nb_model, testHouseVotes84[,-1])
    fraction_correct[i] <- mean(nb_test_predict == testHouseVotes84$Class)
  }
  return(fraction_correct)
}

# Executando o modelo 20 vezes
fraction_correct_predictions <- nb_multiple_runs(0.8, 20)
fraction_correct_predictions
```

```
# Resumo dos resultados
summary(fraction_correct_predictions)

# Desvio padrão
sd(fraction_correct_predictions)


# Os resultados das execuções estão bem próximos, entre 0.87 e 0.95,
# com um desvio padrão de 0.02.
# O Naive Bayes fez um bom trabalho com este conjunto de dados

# Solução Lista de Exercícios - Capítulo 13

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap14")
getwd()


# Para este exemplo, usaremos o dataset Titanic do Kaggle
# Este dataset é famoso e usamos parte dele nas aulas de SQL.
# Ele normalmente é usado por aqueles que estão começando em Machine Learning.

# Vamos prever uma classificação - sobreviventes e não sobreviventes

# https://www.kaggle.com/c/titanic/data


# Comecamos carregando o dataset de dados_treino
dados_treino <- read.csv('datasets/titanic-train.csv')
View(dados_treino)

### ANALISE EXPLORATORIA DOS DADOS ###

# Vamos usar o pacote Amelia e suas funções para definir o volume de dados Missing
# Clique no zoom para visualizar o grafico
# Cerca de 20% dos dados sobre idade estão Missing (faltando)
install.packages("Amelia")
library(Amelia)

# missmap é do pacote Amelia e usado para visualizar dados missing num mapa/grafico
# boa opção para documentação dos valores missing
?missmap
missmap(dados_treino,
        main = "Titanic Training Data - Mapa de Dados Missing",
        col = c("yellow", "black"),
        legend = FALSE)

# Visualizando os dados
library(ggplot2)
ggplot(dados_treino,aes(Survived)) + geom_bar()
ggplot(dados_treino,aes(Pclass)) + geom_bar(aes(fill = factor(Pclass)), alpha = 0.5)
ggplot(dados_treino,aes(Sex)) + geom_bar(aes(fill = factor(Sex)), alpha = 0.5)
ggplot(dados_treino,aes(Age)) + geom_histogram(fill = 'blue', bins = 20, alpha = 0.5)
ggplot(dados_treino,aes(SibSp)) + geom_bar(fill = 'red', alpha = 0.5)
ggplot(dados_treino,aes(Fare)) + geom_histogram(fill = 'green', color = 'black', alpha = 0.5)

### PRE-PROCESSAMENTO - Limpeza dos dados e remoção de valores NA ###
```

```
# Para tratar os dados missing, usaremos o recurso de imputation.
# Essa técnica visa substituir os valores missing por outros valores,
# que podem ser a média da variável ou qualquer outro valor escolhido pelo Cientista de Dados

# Por exemplo, vamos verificar as idades por classe de passageiro (baixa, média, alta):
pl <- ggplot(dados_treino, aes(Pclass,Age)) + geom_boxplot(aes(group = Pclass, fill = factor(Pclass), alpha = 0.4))
pl + scale_y_continuous(breaks = seq(min(0), max(80), by = 2))

# Vimos que os passageiros mais ricos, nas classes mais altas, tendem a ser mais velhos.
# Usaremos esta média para imputar as idades Missing
impute_age <- function(age, class){ # recebe idade e classe como parametro
  out <- age # a variável idade retornara na variavel out
  for (i in 1:length(age)){

    if (is.na(age[i])){

      if (class[i] == 1){
        out[i] <- 37 # media da primeira classe

      }else if (class[i] == 2){
        out[i] <- 29 # da segunda classe

      }else{
        out[i] <- 24 # da terceira classe
      }
    }else{
      out[i]<-age[i] # retorna propria idade
    }
  }
  return(out)
}

fixed.ages <- impute_age(dados_treino$Age, dados_treino$Pclass)
dados_treino$Age <- fixed.ages

# Visualizando o mapa de valores missing (nao existem mais dados missing)
missmap(dados_treino,
        main = "Titanic Training Data - Mapa de Dados Missing",
        col = c("yellow", "black"),
        legend = FALSE)

### EXERCICIO ###

# Exercício 1 - Construindo o Modelo

### PRE-PROCESSAMENTO ###

# Primeiro, uma limpeza nos dados
str(dados_treino)
head(dados_treino, 3)
library(dplyr) # para uso do select abaixo
dados_treino <- select(dados_treino, -PassengerId, -Name, -Ticket, -Cabin) # removendo 4 variáveis do dataset
head(dados_treino, 3)
str(dados_treino)

### TREINAMENTO DO MODELO ###

# Treine o modelo e depois faça as previsões
log.model <- glm(formula = Survived ~ . , family = binomial(link = 'logit'), data = dados_treino)
# glm - regressão logística

# Podemos ver que as variáveis Sex, Age e Pclass sao as variaveis mais significantes
summary(log.model)
```

```
# Fazendo as previsoes nos dados de teste
library(caTools)
set.seed(101)

### SEPARANDO TREINO E TESTE ###

# Split dos dados
split = sample.split(dados_treino$Survived, SplitRatio = 0.70)

# Datasets de treino e de teste
dados_treino_final = subset(dados_treino, split == TRUE)
dados_teste_final = subset(dados_treino, split == FALSE)

### GERAÇÃO DO MODELO (separado treino e teste) ###
# Gerando o modelo com a versão final do dataset
final.log.model <- glm(formula = Survived ~ . , family = binomial(link='logit'), data = dados_treino_final)

# Resumo
summary(final.log.model)

# Prevendo a acurácia
fitted.probabilities <- predict(final.log.model, newdata = dados_teste_final, type = 'response')

# Calculando os valores
fitted.results <- ifelse(fitted.probabilities > 0.5, 1, 0)

# Conseguimos quase 80% de acurácia
misClasificError <- mean(fitted.results != dados_teste_final$Survived)
print(paste('Acuracia', 1-misClasificError))

# Criando a confusion matrix
table(dados_teste_final$Survived, fitted.probabilities > 0.5)

# Coleta e Transformação de Dados

# Este código contém comandos para filtrar e transformar os dados de aluguel de bikes,
# dados que estão em nosso dataset.

# Este código foi criado para executar tanto no Azure, quanto no RStudio.
# Para executar no Azure, altere o valor da variavel Azure para TRUE.
# Se o valor for FALSE, o código sera executado no RStudio

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap14/
Projeto")
# getwd()


# Variável que controla a execução do script
Azure <- FALSE

# Execução de acordo com o valor da variável Azure
if(Azure){
  source("src/Tools.R")
  bikes <- maml.mapInputPort(1)
  bikes$dteday <- set.asPOSIXct(bikes)
}else{
```

```
  source("src/Tools.R")
  bikes <- read.csv("bikes.csv", sep = ",", header = TRUE, stringsAsFactors = FALSE )

  # Selecionar as variáveis que serão usadas
  cols <- c("dteday", "mnth", "hr", "holiday",
            "workingday", "weathersit", "temp",
            "hum", "windspeed", "cnt")

  # Criando um subset dos dados
  bikes <- bikes[, cols]

  # Transformar o objeto de data
  bikes$dteday <- char.toPOSIXct(bikes)

  # Esta linha acima gera dois valores NA
  # Esta linha abaixo corrige
  bikes <- na.omit(bikes)

  # Normalizar as variaveis preditoras numericas
  cols <- c("temp", "hum", "windspeed")
  bikes[, cols] <- scale(bikes[, cols])
}

#?scale
str(bikes)
View(bikes)

# Criar uma nova variável para indicar dia da semana (workday)
bikes$isWorking <- ifelse(bikes$workingday & !bikes$holiday, 1, 0)

# Adicionar uma coluna com a quantidade de meses, o que vai ajudar a criar o modelo
bikes <- month.count(bikes)

# Criar um fator ordenado para o dia da semana, comecando por segunda-feira
# Neste fator eh convertido para ordenado numérico para ser compativel com os tipos de dados do Azure ML
bikes$dayWeek <- as.factor(weekdays(bikes$dteday))


############ ATENÇÃO ############

# ==> Analise o dataframe bikes.
# Se os nomes dos dias da semana estiverem em português na coluna bikes$dayWeek,
# execute o Bloco1 abaixo, caso contrátio, execute o Bloco2 com os nomes em inglês.
# Execute um bloco ou o outro.
str(bikes$dayWeek)

# Bloco1
# Se o seu sistema operacional estiver em portugês, execute o comando abaixo.
bikes$dayWeek <- as.numeric(ordered(bikes$dayWeek,
                            levels = c("segunda-feira",
                                       "terça-feira",
                                       "quarta-feira",
                                       "quinta-feira",
                                       "sexta-feira",
                                       "sábado",
                                       "domingo")))

# Bloco2
# Se o seu sistema operacional estiver em inglês, execute o comando abaixo.
bikes$dayWeek <- as.numeric(ordered(bikes$dayWeek,
                            levels = c("Monday",
                                       "Tuesday",
                                       "Wednesday",
```

```
                                  "Thursday",
                                  "Friday",
                                  "Saturday",
                                  "Sunday")))

# Agora os dias da semana devem estar como valores numéricos
# Se estiverem como valores NA, volte e verifique se você seguiu as instruções acima.
str(bikes$dayWeek)
str(bikes)

# Adiciona uma variável com valores únicos para o horário do dia em dias de semana e dias de fim de semana
# Com isso diferenciamos as horas dos dias de semana, das horas em dias de fim de semana
bikes$workTime <- ifelse(bikes$isWorking, bikes$hr, bikes$hr + 24)

# Transforma os valores de hora na madrugada, quando a demanda por bibicletas é praticamente nula
bikes$xformHr <- ifelse(bikes$hr > 4, bikes$hr - 5, bikes$hr + 19)

# Adiciona uma variável com valores únicos para o horário do dia para dias de semana e dias de fim de semana
# Considerando horas da madrugada
bikes$xformWorkHr <- ifelse(bikes$isWorking, bikes$xformHr, bikes$xformHr + 24)

# str(bikes)
# View(bikes)
# O trabalho que fizemos até aqui também é chamado de Feature Engineering ou
# Engenharia de Atributos

# Gera saída no Azure ML
if(Azure) maml.mapOutputPort('bikes')

# Análise de Correlação

# Este código contém comandos para análise de correlação.

# Este código foi criado para executar tanto no Azure, quanto no RStudio.
# Para executar no Azure, altere o valor da variavel Azure para TRUE.
# Se o valor for FALSE, o codigo será executado no RStudio

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap14/
Projeto")
# getwd()

# Variável que controla a execução do script
Azure <- FALSE

if(Azure){
  source("src/Tools.R")
  bikes <- maml.mapInputPort(1)
  bikes$dteday <- set.asPOSIXct(bikes)
}else{
  bikes <- bikes
}

View(bikes) # o dataset bikes foi criado no script 00-CargaTransformacao.R
# uma boa prática seria salvar esse arquivo como csv (no script R anterior)
# e carregá-lo, nesse script, com read.csv

# Definindo as colunas para a análise de correlação
```

```
cols <- c("mnth", "hr", "holiday", "workingday",
          "weathersit", "temp", "hum", "windspeed",
          "isWorking", "monthCount", "dayWeek",
          "workTime", "xformHr", "cnt")

# Métodos de Correlação
# Pearson - coeficiente usado para medir o grau de relacionamento entre duas variáveis com relação linear
# Spearman - teste não paramétrico, para medir o grau de relacionamento entre duas variaveis
# Kendall - teste não paramétrico, para medir a força de dependência entre duas variaveis

# Vetor com os métodos de correlação
metodos <- c("pearson", "spearman")

# Aplicando os métodos de correlação com a função cor()
cors <- lapply(metodos, function(method)
  (cor(bikes[, cols], method = method)))

head(cors)

# Preprando o plot
require(lattice)
plot.cors <- function(x, labs){
  diag(x) <- 0.0
  plot( levelplot(x,
                  main = paste("Plot de Correlação usando Método", labs),
                  scales = list(x = list(rot = 90), cex = 1.0)) )
}

# Mapa de Correlação
Map(plot.cors, cors, metodos)

# Gera saida no Azure ML
if(Azure) maml.mapOutputPort('bikes')

# Análise de Série Temporal

# Este código contém comandos para análise de série temporal

# Este código foi criado para executar tanto no Azure, quanto no RStudio.
# Para executar no Azure, altere o valor da variavel Azure para TRUE.
# Se o valor for FALSE, o codigo será executado no RStudio

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap14/
Projeto")
# getwd()

# Variável que controla a execução do script
Azure <- FALSE

if(Azure){
  source("src/Tools.R")
  Bikes <- maml.mapInputPort(1)
  Bikes$dteday <- set.asPOSIXct(Bikes)
}else{
  bikes <- bikes
}
```

```
### Descobrindo a demanda por hora do dia (aluguel/hora do dia)

# Avaliando a demanda por aluguel de bikes ao longo do tempo
# Construindo um time series plot para alguns determinados horários
# em dias úteis e dias de fim de semana.
times <- c(7, 9, 12, 15, 18, 20, 22)

# Time Series Plot - definindo a função (ainda não executou)
tms.plot <- function(times){
  ggplot(bikes[bikes$workTime == times, ], aes(x = dteday, y = cnt)) +
    geom_line() +
    ylab("Numero de Bikes") +
    labs(title = paste("Demanda de Bikes as ", as.character(times), ":00", sep = "")) +
    theme(text = element_text(size = 20))
}

require(ggplot2)
lapply(times, tms.plot) # executando a função com o vetor de horários

# Gera saida no Azure ML
if(Azure) maml.mapOutputPort('bikes')

# Analisando BoxPlots

# Este código contém comandos para análise de variáveis usando BoxPlots

# BoxPlots são úteis para verificar se há dados acima da média, se existem muitos
# outliers, como os dados estão relacionados, etc.

# Este código foi criado para executar tanto no Azure, quanto no RStudio.
# Para executar no Azure, altere o valor da variavel Azure para TRUE.
# Se o valor for FALSE, o codigo será executado no RStudio

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap14/
Projeto")
# getwd()

# Variável que controla a execução do script
Azure <- FALSE

if(Azure){
  source("src/Tools.R")
  Bikes <- maml.mapInputPort(1)
  Bikes$dteday <- set.asPOSIXct(Bikes)
}else{
  bikes <- bikes
}

# Convertendo a variável dayWeek para fator ordenado e plotando em ordem de tempo
bikes$dayWeek <- fact.conv(bikes$dayWeek)

# Demanda de bikes x potenciais variáveis preditoras (lista de labels)
labels <- list("Boxplots - Demanda de Bikes por Hora",
               "Boxplots - Demanda de Bikes por Estação",
               "Boxplots - Demanda de Bikes por Dia Útil",
               "Boxplots - Demanda de Bikes por Dia da Semana")
```

```
xAxis <- list("hr", "weathersit", "isWorking", "dayWeek")

# Função para criar os boxplots - automatiza criação de vários gráficos
plot.boxes  <- function(X, label){
  ggplot(bikes, aes_string(x = X, y = "cnt", group = X)) +
    geom_boxplot( ) + # geom_boxplot para uso do conceito de camadas dos gráficos
    ggtitle(label) +
    theme(text = element_text(size = 18))
  }

Map(plot.boxes, xAxis, labels)



# Gera saída no Azure ML
if(Azure) maml.mapOutputPort('bikes')

# Analisando Density Plots

# Este código contém comandos para análise de variáveis usando Density Plots

# Mostra a correlação entre as variáveis, de maneira gráfica

# Este código foi criado para executar tanto no Azure, quanto no RStudio.
# Para executar no Azure, altere o valor da variavel Azure para TRUE.
# Se o valor for FALSE, o codigo será executado no RStudio

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap14/
Projeto")
# getwd()

# Variável que controla a execução do script
Azure <- FALSE

if(Azure){
  source("src/Tools.R")
  Bikes <- maml.mapInputPort(1)
  Bikes$dteday <- set.asPOSIXct(Bikes)
}else{
  bikes <- bikes
}


# Visualizando o relacionamento entre as variáveis preditoras e demanda por bike
labels <- c("Demanda de Bikes vs Temperatura",
            "Demanda de Bikes vs Humidade",
            "Demanda de Bikes vs Velocidade do Vento",
            "Demanda de Bikes vs Hora")

xAxis <- c("temp", "hum", "windspeed", "hr")

# Função para os Density Plots
plot.scatter <- function(X, label){
  ggplot(bikes, aes_string(x = X, y = "cnt")) +
    geom_point(aes_string(colour = "cnt"), alpha = 0.1) +
    scale_colour_gradient(low = "green", high = "blue") +
    geom_smooth(method = "loess") + # linha que mostra uma evolução da relação e uma margem de erro
```

```
    ggtitle(label) +
    theme(text = element_text(size = 20))
  }

Map(plot.scatter, xAxis, labels)

# Explorando a interação entre tempo e dia, em dias da semana e fins de semana
# Criando outros boxplots
labels <- list("Box plots - Demanda por Bikes as 09:00 para \n dias da semana e fins de semana",
               "Box plots - Demanda por Bikes as 18:00  para \n dias da semana e fins de semana")

Times <- list(9, 18) # horários onde se detectou maior demanda pelo aluguel de bikes

plot.box2 <- function(time, label){
  ggplot(bikes[bikes$hr == time, ], aes(x = isWorking, y = cnt, group = isWorking)) +
    geom_boxplot( ) + ggtitle(label) +
    theme(text = element_text(size = 18)) }

Map(plot.box2, Times, labels)

# Gera saída no Azure ML
if(Azure) maml.mapOutputPort('bikes')

# Feature Selection

# Este código contém comandos para feature selection (seleção de variáveis, atributos ou subsets de variáveis)

# Em Machine Learning Feature Selection é o processo de selecionar um subset de variáveis
# que sejam relevantes para construção do modelo preditivo.

# Este código foi criado para executar tanto no Azure, quanto no RStudio.
# Para executar no Azure, altere o valor da variavel Azure para TRUE.
# Se o valor for FALSE, o codigo será executado no RStudio

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("C:/FCD/BigDataRAzure/Cap14/Projeto")
# getwd()

# Variável que controla a execução do script
Azure <- FALSE

if(Azure){
  source("src/Tools.R")
  bikes <- maml.mapInputPort(1)
  bikes$dteday <- set.asPOSIXct(bikes)
}else{
  bikes <- bikes
}

dim(bikes) # dimensão do conjunto de dados
any(is.na(bikes)) # verificando existência de valor NA

# Criando um modelo para identificar os atributos com maior importância para o modelo preditivo
require(randomForest) # caso necessário, instale antes

# Avalidando a importância de todas as variaveis
modelo <- randomForest(cnt ~ . , # target ~ preditoras
                       data = bikes,
```

```
                ntree = 100, # número de árvores
                nodesize = 10, # tamanho de cada nó
                importance = TRUE) # identifica as var de mais relevância para o modelo

# Plotando as variáveis por grau de importância
varImpPlot(modelo) # quanto mais pra direita no gráfico, mais importante é a variável

# Removendo variáveis colineares
modelo <- randomForest(cnt ~ . - mnth
                - hr
                - workingday
                - isWorking
                - dayWeek
                - xformHr
                - workTime
                - holiday
                - windspeed
                - monthCount
                - weathersit,
                data = bikes,
                ntree = 100,
                nodesize = 10,
                importance = TRUE)

# Plotando as variáveis por grau de importância
varImpPlot(modelo)

# Gravando o resultado
df_saida <- bikes[, c("cnt", rownames(modelo$importance))]


if(Azure) maml.mapOutputPort("df_saida ")

# Cria um modelo preditivo usando randomForest

# Este código foi criado para executar tanto no Azure, quanto no RStudio.
# Para executar no Azure, altere o valor da variavel Azure para TRUE.
# Se o valor for FALSE, o codigo será executado no RStudio

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("C:/FCD/BigDataRAzure/Cap14/Projeto")
# getwd()

# Função para tratar as datas
set.asPOSIXct <- function(inFrame) {
  dteday <- as.POSIXct(
    as.integer(inFrame$dteday),
    origin = "1970-01-01")

  as.POSIXct(strptime(
    paste(as.character(dteday),
          " ",
          as.character(inFrame$hr),
          ":00:00",
          sep = ""),
    "%Y-%m-%d %H:%M:%S"))
}
```

```
char.toPOSIXct <-   function(inFrame) {
  as.POSIXct(strptime(
    paste(inFrame$dteday, " ",
          as.character(inFrame$hr),
          ":00:00",
          sep = ""),
    "%Y-%m-%d %H:%M:%S")) }


# Variável que controla a execução do script
Azure <- FALSE

if(Azure){
  dataset$dteday <- set.asPOSIXct(dataset)
}else{
  bikes <- bikes
}

require(randomForest)
model <- randomForest(cnt ~ xformWorkHr + dteday + temp + hum,
                      data = bikes, # altere o nome do objeto data para "dataset" de estiver trabalhando no Azure ML
                      ntree = 40,
                      nodesize = 5)
print(model)

# Score do modelo preditivo com randomForest

# Este código foi criado para executar tanto no Azure, quanto no RStudio.
# Para executar no Azure, altere o valor da variavel Azure para TRUE.
# Se o valor for FALSE, o codigo será executado no RStudio

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("C:/FCD/BigDataRAzure/Cap14/Projeto")
# getwd()

# Função para tratar as datas
set.asPOSIXct <- function(inFrame) {
  dteday <- as.POSIXct(
    as.integer(inFrame$dteday),
    origin = "1970-01-01")

  as.POSIXct(strptime(
    paste(as.character(dteday),
          " ",
          as.character(inFrame$hr),
          ":00:00",
          sep = ""),
    "%Y-%m-%d %H:%M:%S"))
}

char.toPOSIXct <-   function(inFrame) {
  as.POSIXct(strptime(
    paste(inFrame$dteday, " ",
          as.character(inFrame$hr),
          ":00:00",
          sep = ""),
    "%Y-%m-%d %H:%M:%S")) }
```

```
# Variável que controla a execução do script
Azure <- FALSE


if(Azure){
  bikes <- dataset
  bikes$dteday <- set.asPOSIXct(bikes)
}else{
  bikes <- bikes
}

require(randomForest)
scores <- data.frame(actual = bikes$cnt,
                     prediction = predict(model, newdata = bikes))

# Avaliação do modelo

# Este código foi criado para executar tanto no Azure, quanto no RStudio.
# Para executar no Azure, altere o valor da variavel Azure para TRUE.
# Se o valor for FALSE, o codigo será executado no RStudio

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("C:/FCD/BigDataRAzure/Cap14/Projeto")
# getwd()


# Variável que controla a execução do script
Azure <- FALSE

if(Azure){
  source("src/Tools.R")
  inFrame <- maml.mapInputPort(1)
  refFrame <- maml.mapInputPort(2)
  refFrame$dteday <- set.asPOSIXct2(refFrame)
}else{
  source("src/Tools.R")
  inFrame <- scores[, c("actual", "prediction")]
  refFrame <- bikes
}

# Criando um dataframe
inFrame[, c("dteday", "monthCount", "hr", "xformWorkHr")] <- refFrame[, c("dteday", "monthCount", "hr",
"xformWorkHr")]

# Nomeando o dataframe
names(inFrame) <- c("cnt", "predicted", "dteday", "monthCount", "hr", "xformWorkHr")

#  Time series plot mostrando a diferença entre valores reais e valores previstos
library(ggplot2)
inFrame <- inFrame[order(inFrame$dteday),]
s <- c(7, 9, 12, 15, 18, 20, 22)

lapply(s, function(s){
  ggplot() +
    geom_line(data = inFrame[inFrame$hr == s, ],
              aes(x = dteday, y = cnt)) +
    geom_line(data = inFrame[inFrame$hr == s, ],
```

```
            aes(x = dteday, y = predicted), color = "red") +
    ylab("Numero de Bikes") +
    labs(title = paste("Demanda de Bikes as ",
                       as.character(s), ":00", spe ="")) +
    theme(text = element_text(size = 20))
})

# Computando os resíduos (valores previstos menos valores reais)
library(dplyr) # para uso da função mutate
inFrame <-  mutate(inFrame, resids = predicted - cnt)

# Plotando os resíduos
ggplot(inFrame, aes(x = resids)) +
  geom_histogram(binwidth = 1, fill = "white", color = "black")

qqnorm(inFrame$resids)
qqline(inFrame$resids)

# Plotando os resíduos com as horas transformadas
inFrame <- mutate(inFrame, fact.hr = as.factor(hr),
                  fact.xformWorkHr = as.factor(xformWorkHr))
facts <- c("fact.hr", "fact.xformWorkHr")
lapply(facts, function(x){
  ggplot(inFrame, aes_string(x = x, y = "resids")) +
    geom_boxplot( ) +
    ggtitle("Residuos - Demanda de Bikes por Hora - Atual vs Previsto")})


# Mediana dos resíduos por hora
evalFrame <- inFrame %>%
  group_by(hr) %>%
  summarise(medResidByHr = format(round(
    median(predicted - cnt), 2),
    nsmall = 2))

# Computando a mediana dos resíduos
tempFrame <- inFrame %>%
  group_by(monthCount) %>%
  summarise(medResid = median(predicted - cnt))

evalFrame$monthCount <- tempFrame$monthCount
evalFrame$medResidByMcnt <- format(round(
  tempFrame$medResid, 2),
  nsmall = 2)

print("Resumo dos residuos")
print(evalFrame)

# Output
outFrame <- data.frame(evalFrame)
if(Azure) maml.mapOutputPort('outFrame')

# Este script contém diversas funções utilitárias usadas em diversos scripts R.
# Este arquivo deve ser compactado em um arquivo zip e importado no Azure ML.
# Para usar as funções deste script, utilizamos a funcao source() para carregar o script.

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

set.asPOSIXct <- function(inFrame) {
  dteday <- as.POSIXct(
    as.integer(inFrame$dteday),
    origin = "1970-01-01")
```

```
  as.POSIXct(strptime(
    paste(as.character(dteday),
          " ",
          as.character(inFrame$hr),
          ":00:00",
          sep = ""),
    "%Y-%m-%d %H:%M:%S"))
}

char.toPOSIXct <-   function(inFrame) {
  as.POSIXct(strptime(
    paste(inFrame$dteday, " ",
          as.character(inFrame$hr),
          ":00:00",
          sep = ""),
    "%Y-%m-%d %H:%M:%S")) }


set.asPOSIXct2 <- function(inFrame) {
  dteday <- as.POSIXct(
    as.integer(inFrame$dteday),
    origin = "1970-01-01")
}


fact.conv <- function(inVec){
  outVec <- as.factor(inVec)
  levels(outVec) <- c("Segunda", "Terca", "Quarta",
                      "Quinta", "Sexta", "Sabado",
                      "Domingo")
  outVec
}

get.date <- function(Date){
  temp <- strftime(Date, format = "%Y-%m-%d %H:%M:%S")
  substr(unlist(temp), 1, 10)
}


POSIX.date <- function(Date,Hour){
  as.POSIXct(strptime(paste(Date, " ", as.character(Hour),
                            ":00:00", sep = ""),
                      "%Y-%m-%d %H:%M:%S"))
}

var.log <- function(inFrame, col){
  outVec <- ifelse(inFrame[, col] < 0.1, 1, inFrame[, col])
  log(outVec)
}

month.count <- function(inFrame){
  Dteday <- strftime(inFrame$dteday, format = "%Y-%m-%dT%H:%M:%S")
  yearCount <- as.numeric(unlist(lapply(strsplit(
    Dteday, "-"),
    function(x){x[1]}))) - 2011
  inFrame$monthCount <- 12 * yearCount + inFrame$mnth
  inFrame
}

serList <- function(serlist){

  messages  <- c("O input nao eh uma lista ou tem comprimento maior que 0",
```

```
              "Elementos nulos",
              "A serializacao falhou")

  if(!is.list(serlist) | is.null(serlist) |
     length(serlist) < 1) {
    warning(messages[2])
    return(data.frame(as.integer(serialize(
      list(numElements = 0, payload = NA),
      connection = NULL))))}

  nObj  <-  length(serlist)

  tryCatch(outframe <- data.frame(payload = as.integer(
    serialize(list(numElements = nObj,
                   payload = serlist),
              connection=NULL))),
    error = function(e){warning(messages[3])
      outframe <- data.frame(
        payload = as.integer(serialize(list(
          numElements = 0, payload = NA),
          connection=NULL)))}
  )
  outframe
}


unserList <- function(inlist){

  messages <- c("A coluna payload esta missing ou com tipo incorreto de dado",
                "Erro ao executar esta funcao",
                "A funcao gerou uma lista vazia")

  if(!is.integer(inlist$payload) | dim(inlist)[1] < 2 |
     is.null(inlist$payload)){
    warning(messages[1])
    return(NA)
  }

  tryCatch(outList <- unserialize(as.raw(inlist$payload)),
           error = function(e){warning(messages[2]); return(NA)})

  if(outList$numElements < 1 ) {warning(messages[3]);
    return(NA)}

  outList$payload
}
```

Capítulo 15:

```
# Script para checar as colunas do dataset

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("C:/FCD/BigDataRAzure/Cap15/Projeto")
setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap15")
getwd()

# Carrega o dataset antes da transformacao
df <- read.csv("credito.csv")
View(df)
str(df)


# Nome das variáveis
# CheckingAcctStat, Duration, CreditHistory, Purpose, CreditAmount, SavingsBonds, Employment,
InstallmentRatePecnt, SexAndStatus, OtherDetorsGuarantors, PresentResidenceTime, Property, Age, OtherInstallments,
Housing, ExistingCreditsAtBank, Job, NumberDependents, Telephone, ForeignWorker, CreditStatus

# Aplicando Engenharia de Atributos em Variáveis Numéricas

# Este código foi criado para executar tanto no Azure, quanto no RStudio.
# Para executar no Azure, altere o valor da variavel Azure para TRUE.
# Se o valor for FALSE, o código sera executado no RStudio

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("C:/FCD/BigDataRAzure/Cap15/Projeto")
# setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap15")
# getwd()

# Variável que controla a execução do script
Azure <- FALSE

if(Azure){
  source("src/ClassTools.R")
  Credit <- maml.mapInputPort(1)
}else{
  source("src/ClassTools.R")
  Credit <- read.csv("credito.csv", header = F, stringsAsFactors = F )
  metaFrame <- data.frame(colNames, isOrdered, I(factOrder))
  Credit <- fact.set(Credit, metaFrame)

  # Balancear o número de casos positivos e negativos
  Credit <- equ.Frame(Credit, 2)
}

# Transformando variáveis numéricas em variáveis categóricas (criando 3 novas e mantendo as antigas)
toFactors <- c("Duration", "CreditAmount", "Age") # vetor com as variáveis
maxVals <- c(100, 1000000, 100) # vetor com os valores máximos para cada variável
facNames <- unlist(lapply(toFactors, function(x) paste(x, "_f", sep = ""))) # criando uma função, aplicando a função
paste para toFactors
```

```
Credit[, facNames] <- Map(function(x, y) quantize.num(Credit[, x], maxval = y), toFactors, maxVals)
#quantize.num está no arquivo classTools.R

# str(Credit)

# Output
if(Azure) maml.mapOutputPort('Credit')

# Análise Exploratória de Dados

# Este codigo foi criado para executar tanto no Azure, quanto no RStudio.
# Para executar no Azure, altere o valor da variavel Azure para TRUE. Se o valor for FALSE, o codigo sera executado no RStudio

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("C:/FCD/BigDataRAzure/Cap15/Projeto")
# setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap15")
# getwd()

# Variável que controla a execução do script
Azure <- FALSE

if(Azure){
  source("src/ClassTools.R")
  Credit <- maml.mapInputPort(1)
}

# Plots usando ggplot2
library(ggplot2)
lapply(colNames2, function(x){ # colNames2 está no arquivo ClassTools.R
  if(is.factor(Credit[,x])) {
    ggplot(Credit, aes_string(x)) +
      geom_bar() +
      # facet_grid permite incluir vários gráficos numa mesma área de plotagem
      facet_grid(. ~ CreditStatus) + # comparando todas as variáveis à CreditStatus
      ggtitle(paste("Total de Credito Bom/Ruim por",x))}})

# Plots CreditStatus vs CheckingAcctStat
lapply(colNames2, function(x){
  if(is.factor(Credit[,x]) & x != "CheckingAcctStat") {
    ggplot(Credit, aes(CheckingAcctStat)) +
      geom_bar() +
      # paste cola(junta) strings. Nesse caso, será usado para juntar cada valor de x à variável CreditStatus
      facet_grid(paste(x, " ~ CreditStatus"))+ # comparando cada variável de maneira separada à CreditStatus
      ggtitle(paste("Total de Credito Bom/Ruim CheckingAcctStat e",x))
  }})

# Feature Selection (Seleção de Variáveis)

# Este código foi criado para executar tanto no Azure, quanto no RStudio.
# Para executar no Azure, altere o valor da variavel Azure para TRUE.
# Se o valor for FALSE, o código sera executado no RStudio

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
```

```
# Não use diretórios com espaço no nome
# setwd("C:/FCD/BigDataRAzure/Cap15/Projeto")
# getwd()

# Variavel que controla a execucao do script
Azure <- FALSE

if(Azure){
  source("src/ClassTools.R")
  Credit <- maml.mapInputPort(1)
}

# Modelo randomForest para criar um plot de importância das variáveis
library(randomForest)
modelo <- randomForest( CreditStatus ~ .
                        - Duration
                        - Age
                        - CreditAmount
                        - ForeignWorker
                        - NumberDependents
                        - Telephone
                        - ExistingCreditsAtBank
                        - PresentResidenceTime
                        - Job
                        - Housing
                        - SexAndStatus
                        - InstallmentRatePecnt
                        - OtherDetorsGuarantors
                        - Age_f
                        - OtherInstalments,
                        data = Credit,
                        ntree = 100, nodesize = 10, importance = T)

varImpPlot(modelo)

outFrame <- serList(list(credit.model = modelo))


## Output
if(Azure) maml.mapOutputPort("outFrame")

# Criando o Modelo Preditivo no R

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("C:/FCD/BigDataRAzure/Cap15/Projeto")
# setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap15")
# getwd()

# Criar um modelo de classificação baseado em randomForest
library(randomForest)

# Cross Tabulation - Tabela Cruzada
?table
table(Credit$CreditStatus) # CreditSatus é a variável target

# Funcao anônima (criada em tempo de execução) para gerar dados de treino e dados de teste
splitData <- function(dataframe, seed = NULL) {
  if (!is.null(seed)) set.seed(seed)
```

```
  index <- 1:nrow(dataframe)
  trainindex <- sample(index, trunc(length(index)/2)) # sample função para amostragem (50% treino/teste)
  trainset <- dataframe[trainindex, ]
  testset <- dataframe[-trainindex, ]
  list(trainset = trainset, testset = testset)
}

# Gerando dados de treino e de teste, aplicando a fução splitData
splits <- splitData(Credit, seed = 808)

# Separando os dados
dados_treino <- splits$trainset
dados_teste <- splits$testset

# Verificando o numero de linhas
nrow(dados_treino)
nrow(dados_teste)

# Construindo o modelo
modelo <- randomForest( CreditStatus ~ CheckingAcctStat
                  + Duration_f
                  + Purpose
                  + CreditHistory
                  + SavingsBonds
                  + Employment
                  + CreditAmount_f,
                  data = dados_treino,
                  ntree = 100,
                  nodesize = 10)

# Imprimondo o resultado
print(modelo)

# Fazendo Previsões - Score Model

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("C:/FCD/BigDataRAzure/Cap15/Projeto")
# setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap15")
# getwd()

# Previsões com um modelo de classificação baseado em randomForest
require(randomForest)

# Gerando previsões nos dados de teste
previsoes <- data.frame(observado = dados_teste$CreditStatus,
                        previsto = predict(modelo, newdata = dados_teste))

# modelo e dados_teste foram criados no script 04-CriaModelo.R


# Visualizando o resultado
View(previsoes)
View(dados_teste)

# Calculando a Confusion Matrix em R (existem outras formas)

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding
```

```
# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("C:/FCD/BigDataRAzure/Cap15/Projeto")
# setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap15")
# getwd()

# Label 1 - Credito Ruim
# Label 2 - Credito Bom

# Formulas
Accuracy <- function(x){
  (x[1,1] + x[2,2]) / (x[1,1] + x[1,2] + x[2,1] + x[2,2])
}

Recall <- function(x){
  x[1,1] / (x[1,1] + x[1,2])
}

Precision <- function(x){
  x[1,1] / (x[1,1] + x[2,1])
}

W_Accuracy  <- function(x){
  (x[1,1] + x[2,2]) / (x[1,1] + 5 * x[1,2] + x[2,1] + x[2,2])
}

F1 <- function(x){
  2 * x[1,1] / (2 * x[1,1] + x[1,2] + x[2,1])
}

# onde:
# x é o parâmetro recebido ou o dataframe previsoes (do script 05-ScoreModel.R)
# 1, 1 = valor observado e valor previsto da classe 1
# 2, 2 = valor observado e valor previsto da classe 2


# Criando a confusion matrix.
confMat <- matrix(unlist(Map(function(x, y){sum(ifelse(previsoes[, 1] == x & previsoes[, 2] == y, 1, 0) )},
                             c(2, 1, 2, 1), c(2, 2, 1, 1))), nrow = 2)

# onde:
# função Map recebe uma função, depois é feito o unlist e transforma numa matriz com matrix

# Criando um dataframe com as estatisticas dos testes
df_mat <- data.frame(Category = c("Credito Ruim", "Credito Bom"),
                     Classificado_como_ruim = c(confMat[1,1], confMat[2,1]),
                     Classificado_como_bom = c(confMat[1,2], confMat[2,2]),
                     Accuracy_Recall = c(Accuracy(confMat), Recall(confMat)),
                     Precision_WAcc = c(Precision(confMat), W_Accuracy(confMat)))

print(df_mat)

# Gerando uma curva ROC em R
install.packages("ROCR")
library("ROCR")

# Gerando as classes de dados
class1 <- predict(modelo, newdata = dados_teste, type = 'prob')
class2 <- dados_teste$CreditStatus

# onde:
```

```
# Class1 representa as previsões do modelo
# Class2 representa os dados originais do dataset de testes


# Gerando a curva ROC
?prediction # do pacote ROCR
?performance # do pacote ROCR
pred <- prediction(class1[,2], class2)
perf <- performance(pred, "tpr","fpr")
plot(perf, col = rainbow(10))

# Gerando Confusion Matrix com o Caret
library(caret)
?confusionMatrix
confusionMatrix(previsoes$observado, previsoes$previsto)

# Otimizando o Modelo preditivo - Tentando Aumentar a Acurácia do Modelo

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("C:/FCD/BigDataRAzure/Cap15/Projeto")
# setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap15")
# getwd()

# Modelo randomForest ponderado - atribuindo peso para os erros do modelo (penalizar os erros)
# O pacote C50 permite que você dê peso aos erros, construindo assim um resultado ponderado
install.packages("C50")
library(C50)

# Criando uma Cost Function - Matriz de pesos
# a matriz de erro terá um peso 0 para os acertos e de 1.5 para [2,1] e 1.0 para [1,2]
Cost_func <- matrix(c(0, 1.5, 1, 0), nrow = 2, dimnames = list(c("1", "2"), c("1", "2")))



# Criando o Modelo
?randomForest
?C5.0


# Cria o modelo
modelo_v2  <- C5.0(CreditStatus ~ CheckingAcctStat
                   + Purpose
                   + CreditHistory
                   + SavingsBonds
                   + Employment,
                   data = dados_treino,
                   trials = 100,
                   cost = Cost_func) # matriz de pesos

print(modelo_v2)


# Dataframes com valores observados e previstos
previsoes_v2 <- data.frame(observado = dados_teste$CreditStatus,
                           previsto = predict(object = modelo_v2, newdata = dados_teste))

# Calculando a Confusion Matrix em R (existem outras formas).
```

```
# Label 1 - Credito Ruim
# Label 2 - Credito Bom

# Formulas
Accuracy <- function(x){
  (x[1,1] + x[2,2]) / (x[1,1] + x[1,2] + x[2,1] + x[2,2])
}

Recall <- function(x){
  x[1,1] / (x[1,1] + x[1,2])
}

Precision <- function(x){
  x[1,1] / (x[1,1] + x[2,1])
}

W_Accuracy  <- function(x){
  (x[1,1] + x[2,2]) / (x[1,1] + 5 * x[1,2] + x[2,1] + x[2,2])
}

F1 <- function(x){
  2 * x[1,1] / (2 * x[1,1] + x[1,2] + x[2,1])
}

# Criando a confusion matrix.
confMat_v2 <- matrix(unlist(Map(function(x, y){sum(ifelse(previsoes_v2[, 1] == x & previsoes_v2[, 2] == y, 1, 0) )},
                        c(2, 1, 2, 1), c(2, 2, 1, 1))), nrow = 2)


# Criando um dataframe com as estatisticas dos testes
df_mat <- data.frame(Category = c("Credito Ruim", "Credito Bom"),
                     Classificado_como_ruim = c(confMat_v2[1,1], confMat_v2[2,1]),
                     Classificado_como_bom = c(confMat_v2[1,2], confMat_v2[2,2]),
                     Accuracy_Recall = c(Accuracy(confMat_v2), Recall(confMat_v2)),
                     Precision_WAcc = c(Precision(confMat_v2), W_Accuracy(confMat_v2)))

print(df_mat)


# Gerando Confusion Matrix com o Caret
library(caret)
confusionMatrix(previsoes_v2$observado, previsoes_v2$previsto)

# Ao final, essa tentativa de otimização não deu certo. A acurácia do modelo caiu!

# Analisando o resultado atraves de gráficos (bônus extra)

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("C:/FCD/BigDataRAzure/Cap15/Projeto")
# getwd()

Azure <- FALSE

# Alterando atribuição da variável compFrame
if(Azure){
  source("src/ClassTools.R")
  compFrame <- maml.mapInputPort(1)
} else {
```

```
  compFrame <- result_previsto_v2 #outFrame
}

# Usando o dplyr para filter linhas com classificação incorreta
require(dplyr)
creditTest <- cbind(dados_teste, scored = compFrame[ ,2] )
creditTest <- creditTest %>% filter(CreditStatus != scored)

# Plot dos residuos para os niveis de cada fator
require(ggplot2)
colNames <- c("CheckingAcctStat", "Duration_f", "Purpose",
              "CreditHistory", "SavingsBonds", "Employment",
              "CreditAmount_f", "Employment")

lapply(colNames, function(x){
  if(is.factor(creditTest[,x])) {
    ggplot(creditTest, aes_string(x)) +
      geom_bar() +
      facet_grid(. ~ CreditStatus) +
      ggtitle(paste("Numero de creditos ruim/bom por",x))}})


# Plot dos residuos condicionados nas variváveis CreditStatus vs CheckingAcctStat
lapply(colNames, function(x){
  if(is.factor(creditTest[,x]) & x != "CheckingAcctStat") {
    ggplot(creditTest, aes(CheckingAcctStat)) +
      geom_bar() +
      facet_grid(paste(x, " ~ CreditStatus"))+
      ggtitle(paste("Numero de creditos bom/ruim por CheckingAcctStat e ",x))
  }})

# Funções e utilitários para análise de dados do dataset German Credit Data.

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# Não use diretórios com espaço no nome
# setwd("C:/FCD/BigDataRAzure/Cap15/Projeto")
# setwd("D:/Desenvolvedor/CienciaDeDados/Estudos/DSA-Cursos/FCD/03_BDataAnalyticsR_MAzureML/Cap15")
# getwd()

# Função para converter variáveis numéricas para fator
quantize.num <- function(x, nlevs = 5, maxval = 1000,
                         minval = 0, ordered = TRUE){
  cuts <- seq(min(x), max(x), length.out = nlevs + 1)
  cuts[1] <- minval
  cuts[nlevs + 1] <- maxval
  print(cuts)
  x <- cut(x, breaks = cuts, order_result = ordered)
}

# ?cut # função que converte numérico para fator

# Nomeando as variáveis
colNames <- c("CheckingAcctStat",
              "Duration",
              "CreditHistory",
              "Purpose",
              "CreditAmount",
              "SavingsBonds",
              "Employment",
```

```
            "InstallmentRatePecnt",
            "SexAndStatus",
            "OtherDetorsGuarantors",
            "PresentResidenceTime",
            "Property",
            "Age",
            "OtherInstalments",
            "Housing",
            "ExistingCreditsAtBank",
            "Job",
            "NumberDependents",
            "Telephone",
            "ForeignWorker")

# Nomeando as colunas que serao transformadas
colNames2 <- append(colNames, c("Duration_f", "CreditAmount_f", "Age_f"))

# Função anônima para testar o tipo de dado de cada coluna no dataset
colTypes <- list(function(x) is.character(x), # CheckingAcctStat
                 function(x) is.numeric(x),   # Duration
                 function(x) is.character(x), # CreditHistory
                 function(x) is.character(x), # Purpose
                 function(x) is.numeric(x),   # CreditAmount
                 function(x) is.character(x), # SavingsBonds
                 function(x) is.character(x), # Employment
                 function(x) is.numeric(x),   # InstallmentRatePecnt
                 function(x) is.character(x), # SexAndStatus
                 function(x) is.character(x), # OtherDetorsGuarantors
                 function(x) is.numeric(x),   # PresentResidenceTime
                 function(x) is.character(x), # Property
                 function(x) is.numeric(x),   # Age
                 function(x) is.character(x), # OtherInstalments
                 function(x) is.character(x), # Housing
                 function(x) is.numeric(x),   # ExistingCreditsAtBank
                 function(x) is.character(x), # Job
                 function(x) is.numeric(x),   # NumberDependents
                 function(x) is.character(x), # Telephone
                 function(x) is.character(x), # ForeignWorker
                 function(x) is.numeric(x)    # CreditStatus
)


colTypes2 <- list(function(x) is.factor(x),  # CheckingAcctStat
                  function(x) is.numeric(x), # Duration
                  function(x) is.factor(x),  # CreditHistory
                  function(x) is.factor(x),  # Purpose
                  function(x) is.numeric(x), # CreditAmount
                  function(x) is.factor(x),  # SavingsBonds
                  function(x) is.factor(x),  # Employment
                  function(x) is.numeric(x), # InstallmentRatePecnt
                  function(x) is.factor(x),  # SexAndStatus
                  function(x) is.factor(x),  # OtherDetorsGuarantors
                  function(x) is.numeric(x), # PresentResidenceTime
                  function(x) is.factor(x),  # Property
                  function(x) is.numeric(x), # Age
                  function(x) is.factor(x),  # OtherInstalments
                  function(x) is.factor(x),  # Housing
                  function(x) is.numeric(x), # ExistingCreditsAtBank
                  function(x) is.factor(x),  # Job
                  function(x) is.numeric(x), # NumberDependents
                  function(x) is.factor(x),  # Telephone
                  function(x) is.factor(x),  # ForeignWorker
                  function(x) is.factor(x)   # CreditStatus
```

```
)

# Indicar se o fator esta ordenado
isOrdered  <- as.logical(c(T,
                           F,
                           T,
                           F,
                           F,
                           T,
                           T,
                           F,
                           F,
                           T,
                           F,
                           T,
                           F,
                           T,
                           T,
                           F,
                           T,
                           F,
                           T,
                           T))

# Ordem dos fatores
factOrder  <- list(list("A11", "A14", "A12", "A13"),
                   NA,
                   list("A34", "A33", "A30", "A32", "A31"),
                   NA,
                   NA,
                   list("A65", "A61", "A62", "A63", "A64"),
                   list("A71", "A72", "A73", "A74", "A75"),
                   NA,
                   NA,
                   list("A101", "A102", "A103"),
                   NA,
                   list("A124", "A123", "A122", "A121"),
                   NA,
                   list("A143", "A142", "A141"),
                   list("A153", "A151", "A152"),
                   NA,
                   list("A171", "A172", "A173", "A174"),
                   NA,
                   list("A191", "A192"),
                   list("A201", "A202"))


fact.set  <- function(inframe, metaframe){
  # Esta funcao transforma o dataset para garantir que os fatores estao definidos
  # e adiciona nomes as colunas.
  numcol <- ncol(inframe) - 1
  for(i in 1:numcol){
    if(!is.numeric(inframe[, i])){
      inframe[, i]  <- as.factor(inframe[, i])}
  }

  inframe[, 21] <- as.factor(inframe[, 21])
  colnames(inframe) <- c(as.character(metaframe[, 1]), "CreditStatus")
  inframe
}
```

```
equ.Frame <- function(in.frame, nrep){
  nrep <- nrep - 1
  # Cria o dataframe com numero igual de respostas positivas e negativas
  # e converte a coluna 21 para fator.
  if(nrep > 0){
    posFrame  <- in.frame[in.frame[, "CreditStatus"] == 2, ]
    posFrame <- posFrame[rep(seq_len(nrow(posFrame)), nrep), ]
    in.frame <- rbind(in.frame, posFrame)
    #    in.frame <- data.frame(Map(function(x,y){rbind(x, rep(y, nrep))},
    #                           in.frame, posFrame))
  }
  in.frame
}

# Serialização
serList <- function(serlist){

  ## Mensagens em caso de erro
  messages  <- c("Input nao eh uma lista ou tem comprimento maior que zero",
                 "Elementos de input da lista sao NULL ou comprimento maior que 1",
                 "A serializacao falhou")

  if(!is.list(serlist) | is.null(serlist) | length(serlist) < 1) {
    warning(messages[2])
    return(data.frame(as.integer(serialize(list(numElements = 0, payload = NA), connection = NULL))))}

  # Encontrando o numero de objetos
  nObj  <-  length(serlist)

  tryCatch(outframe <- data.frame(payload = as.integer(serialize(list(numElements = nObj, payload = serlist),
connection=NULL))),
           error = function(e){warning(messages[3])
             outframe <- data.frame(payload = as.integer(serialize(list(numElements = 0, payload = NA),
connection=NULL)))}
  )
  outframe
}

unserList <- function(inlist){

  # Mensagens em caso de erro
  messages <- c("A coluna esta faltando ou nao eh do tipo correto",
                "Serializacao falhou",
                "Funcao encontrou um erro")

  # Checando o tipo de dado de entrada
  if(!is.integer(inlist$payload) | dim(inlist)[1] < 2 |
     is.null(inlist$payload | inlist$numElements < 1)){
    warning(messages[1])
    return(NA)
  }


  tryCatch(outList <- unserialize(as.raw(inlist$payload)),
           error = function(e){warning(messages[2]); return(NA)})


  if(outList$numElements < 1 ) {warning(messages[3]); return(NA)}

  outList$payload
}
```

Capítulo 17:

```
# Mini-Projeto 1 - Análise de Sentimentos em Redes Sociais

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Projetos/Mini-Projeto01")
getwd()

## Etapa 1 - Pacotes e Autenticação

# Instalando e Carregando o Pacote twitteR
install.packages("twitteR")
install.packages("httr")
library(twitteR)
library(httr)

# Carregando a biblioteca com funções de limpeza
source('utils.R') # script auxiliar

# Chaves de autenticação no Twitter
key <- ""
secret <- ""
token <- ""
tokensecret <- ""

# Autenticação. Responda 1 quando perguntado sobre utilizar direct connection.
setup_twitter_oauth(key, secret, token, tokensecret)


## Etapa 2 - Conexão e captura dos tweets

# Verificando a timeline do usuário
userTimeline("dsacademybr") # dsacademybr é uma conta no twitter!!! Pode ser outra!

# Capturando os tweets
tema <- "BigData" # tema
qtd_tweets <- 100 # quantidade de tweets para coletar
lingua <- "pt" # linguagem
tweetdata = searchTwitter(tema, n = qtd_tweets, lang = lingua)
# searchTwitter função que procura por tweets

# Visualizando as primeiras linhas do objeto tweetdata
head(tweetdata)


## Etapa 3 - Tratamento dos dados coletados através de text mining

# Instalando o pacote para Text Mining.
install.packages("tm")
install.packages("SnowballC")
library(SnowballC)
library(tm)
options(warn=-1)

# Tratamento (limpeza, organização e transformação) dos dados coletados
tweetlist <- sapply(tweetdata, function(x) x$getText())
tweetlist <- iconv(tweetlist, to = "utf-8", sub="")
tweetlist <- limpaTweets(tweetlist) # funcao contida no utils.R
```

```
tweetcorpus <- Corpus(VectorSource(tweetlist))
tweetcorpus <- tm_map(tweetcorpus, removePunctuation)
tweetcorpus <- tm_map(tweetcorpus, content_transformer(tolower))
tweetcorpus <- tm_map(tweetcorpus, function(x)removeWords(x, stopwords()))

# Convertendo o objeto Corpus para texto plano
# tweetcorpus <- tm_map(tweetcorpus, PlainTextDocument)

# converte o texto numa matriz numérica
termo_por_documento = as.matrix(TermDocumentMatrix(tweetcorpus), control = list(stopwords =
c(stopwords("portuguese"))))



## Etapa 4 - Wordcloud, associação entre as palavras e dendograma

# Instalando o pacote wordcloud
install.packages("RColorBrewer")
install.packages("wordcloud")
library(RColorBrewer)
library(wordcloud)

# Gerando uma nuvem palavras
pal2 <- brewer.pal(8,"Dark2")

wordcloud(tweetcorpus,
          min.freq = 2,
          scale = c(5,1),
          random.color = F,
          max.word = 60,
          random.order = F,
          colors = pal2)

# Convertendo o objeto texto para o formato de matriz
tweettdm <- TermDocumentMatrix(tweetcorpus)
tweettdm

# Encontrando as palavras que aparecem com mais frequência
findFreqTerms(tweettdm, lowfreq = 11)

# Buscando associações
findAssocs(tweettdm, 'datascience', 0.60)

# Removendo termos esparsos (não utilizados frequentemente)
tweet2tdm <- removeSparseTerms(tweettdm, sparse = 0.9)

# Criando escala nos dados
tweet2tdmscale <- scale(tweet2tdm)

# Distance Matrix
tweetdist <- dist(tweet2tdmscale, method = "euclidean")

# Preparando o dendograma
tweetfit <- hclust(tweetdist)

# Criando o dendograma (verificando como as palvras se agrupam)
plot(tweetfit)

# Verificando os grupos
cutree(tweetfit, k = 4)

# Visualizando os grupos de palavras no dendograma
rect.hclust(tweetfit, k = 3, border = "red")
```

```
## Etapa 5 - Análise de Sentimento

# Criando uma função para avaliar o sentimento
install.packages("stringr")
install.packages("plyr")
library(stringr)
library(plyr)

### Função criada para calcular o score dos sentimentos:
sentimento.score = function(sentences, pos.words, neg.words, .progress = 'none')
{

  # Criando um array de scores com lapply
  scores = laply(sentences,
                 function(sentence, pos.words, neg.words)
                 {
                   sentence = gsub("[[:punct:]]", "", sentence)
                   sentence = gsub("[[:cntrl:]]", "", sentence)
                   sentence =gsub('\\d+', '', sentence)
                   tryTolower = function(x)
                   {
                     y = NA
                     try_error = tryCatch(tolower(x), error=function(e) e)
                     if (!inherits(try_error, "error"))
                       y = tolower(x)
                     return(y)
                   }

                   sentence = sapply(sentence, tryTolower)
                   word.list = str_split(sentence, "\\s+")
                   words = unlist(word.list)
                   pos.matches = match(words, pos.words)
                   neg.matches = match(words, neg.words)
                   pos.matches = !is.na(pos.matches)
                   neg.matches = !is.na(neg.matches)
                   score = sum(pos.matches) - sum(neg.matches)
                   return(score)
                 }, pos.words, neg.words, .progress = .progress )

  scores.df = data.frame(text = sentences, score = scores)
  return(scores.df)
}
### Fim sentimento.score

# Mapeando as palavras positivas e negativas
pos = readLines("palavras_positivas.txt")
neg = readLines("palavras_negativas.txt")

# Criando massa de dados para teste
teste = c("Big Data is the future", "awesome experience",
          "analytics could not be bad", "learn to use big data")

# Testando a função em nossa massa de dados dummy
testesentimento = sentimento.score(teste, pos, neg)
class(testesentimento)

# Verificando o score
# 0 - expressão não possui palavra em nossas listas de palavras positivas e negativas ou
# encontrou uma palavra negativa e uma positiva na mesma sentença
# 1 - expressão possui palavra com conotação positiva
# -1 - expressão possui palavra com conotação negativa
```

```
testesentimento$score


## Etapa 6 - Gerando Score da Análise de Sentimento

# Tweets por país
catweets = searchTwitter("ca", n = 500, lang = "en") # Canada
usatweets = searchTwitter("usa", n = 500, lang = "en") # EUA

# Obtendo texto
catxt = sapply(catweets, function(x) x$getText())
usatxt = sapply(usatweets, function(x) x$getText())

# Vetor de tweets dos países
paisTweet = c(length(catxt), length(usatxt))

# Juntando os textos
paises = c(catxt, usatxt)

# Aplicando função para calcular o score de sentimento
scores = sentimento.score(paises, pos, neg, .progress = 'text')

# Calculando o score por país
scores$paises = factor(rep(c("ca", "usa"), paisTweet))
scores$muito.pos = as.numeric(scores$score >= 1)
scores$muito.neg = as.numeric(scores$score <= -1)

# Calculando o total
numpos = sum(scores$muito.pos)
numneg = sum(scores$muito.neg)

# Score global
global_score = round( 100 * numpos / (numpos + numneg) )
head(scores)
boxplot(score ~ paises, data = scores)

# Gerando um histograma com o lattice
install.packages("lattice")
library("lattice")
histogram(data = scores, ~score|paises, main = "Análise de Sentimentos", xlab = "", sub = "Score")



## Usando Classificador Naive Bayes para analise de sentimento
# https://cran.r-project.org/src/contrib/Archive/Rstem/
# https://cran.r-project.org/src/contrib/Archive/sentiment/

# Pacotes que não estão no repositório ativo do CRAN (estão no archive)
install.packages("Rstem_0.4-1.tar.gz", sep = "", repos = NULL, type = "source")
install.packages("sentiment_0.2.tar.gz",sep = "", repos = NULL, type = "source")
#
install.packages("ggplot2")
library(Rstem)
library(sentiment)
library(ggplot2)

# Coletando os tweets
tweetpt = searchTwitter("bigdata", n = 1500, lang = "pt")

# Obtendo o texto
tweetpt = sapply(tweetpt, function(x) x$getText())

# Removendo caracteres especiais
```

```
tweetpt = gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", tweetpt)
# Removendo @
tweetpt = gsub("@\\w+", "", tweetpt)
# Removendo pontuação
tweetpt = gsub("[[:punct:]]", "", tweetpt)
# Removendo digitos
tweetpt = gsub("[[:digit:]]", "", tweetpt)
# Removendo links html
tweetpt = gsub("http\\w+", "", tweetpt)
# Removendo espacos desnecessários
tweetpt = gsub("[ \t]{2,}", "", tweetpt)
tweetpt = gsub("^\\s+|\\s+$", "", tweetpt)

# Criando função para tolower
try.error = function(x)
{
  # Criando missing value
  y = NA
  try_error = tryCatch(tolower(x), error=function(e) e)
  if (!inherits(try_error, "error"))
    y = tolower(x)
  return(y)
}

# Lower case
tweetpt = sapply(tweetpt, try.error)

# Removendo os NAs
tweetpt = tweetpt[!is.na(tweetpt)]
names(tweetpt) = NULL

# Classificando emocao
class_emo = classify_emotion(tweetpt, algorithm = "bayes", prior = 1.0)
emotion = class_emo[,7]

# Substituindo NA's por "Neutro"
emotion[is.na(emotion)] = "Neutro"

# Classificando polaridade
class_pol = classify_polarity(tweetpt, algorithm = "bayes")
polarity = class_pol[,4]

# Gerando um dataframe com o resultado
sent_df = data.frame(text = tweetpt, emotion = emotion,
                     polarity = polarity, stringsAsFactors = FALSE)

# Ordenando o dataframe
sent_df = within(sent_df,
                 emotion <- factor(emotion, levels = names(sort(table(emotion),
                                                                decreasing=TRUE))))


# Emoções encontradas
ggplot(sent_df, aes(x = emotion)) +
  geom_bar(aes(y = ..count.., fill = emotion)) +
  scale_fill_brewer(palette = "Dark2") +
  labs(x = "Categorias", y = "Numero de Tweets")

# Polaridade
ggplot(sent_df, aes(x=polarity)) +
  geom_bar(aes(y=..count.., fill=polarity)) +
  scale_fill_brewer(palette="RdGy") +
  labs(x = "Categorias de Sentimento", y = "Numero de Tweets")
```

```
# Funções auxiliares

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
# setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Projetos/Mini-Projeto01")
# getwd()

# Function para limpeza dos tweets
limpaTweets <- function(tweet){
  # Remove http links
  tweet = gsub("(f|ht)(tp)(s?)(://)(.*)[.|/](.*)", " ", tweet)
  tweet = gsub("http\\w+", "", tweet)
  # Remove retweets
  tweet = gsub("(RT|via)((?:\\b\\W*@\\w+)+)", " ", tweet)
  # Remove “#Hashtag”
  tweet = gsub("#\\w+", " ", tweet)
  # Remove nomes de usuarios “@people”
  tweet = gsub("@\\w+", " ", tweet)
  # Remove pontuacão
  tweet = gsub("[[:punct:]]", " ", tweet)
  # Remove os números
  tweet = gsub("[[:digit:]]", " ", tweet)
  # Remove espacos desnecessários
  tweet = gsub("[ \t]{2,}", " ", tweet)
  tweet = gsub("^\\s+|\\s+$", "", tweet)
  # Convertendo encoding de caracteres e convertendo para letra minúscula
  tweet <- stringi::stri_trans_general(tweet, "latin-ascii")
  tweet <- tryTolower(tweet)
  tweet <- iconv(tweet, from = "UTF-8", to = "ASCII")
}

# Function para limpeza de Corpus
limpaCorpus <- function(myCorpus){
  library(tm)
  myCorpus <- tm_map(myCorpus, tolower)
  # Remove pontuação
  myCorpus <- tm_map(myCorpus, removePunctuation)
  # Remove números
  myCorpus <- tm_map(myCorpus, removeNumbers)
}

# Converte para minúsculo
tryTolower = function(x)
{
  # Cria um dado missing (NA)
  y = NA
  # faz o tratamento do erro
  try_error = tryCatch(tolower(x), error=function(e) e)
  # se não der erro, transforma em minúsculo
  if (!inherits(try_error, "error"))
    y = tolower(x)
  # Retorna o resultado
  return(y)
}
```

Capítulo 18:

```
# Construindo um Modelo Preditivo para Análise de Risco

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Projetos/Mini-Projeto04")
getwd()

# Carregando o dataset em um dataframe
credit.df <- read.csv("credit_dataset.csv", header = TRUE, sep = ",")
head(credit.df)

## Convertendo as variáveis para o tipo fator (categórica)
to.factors <- function(df, variables){
  for (variable in variables){
    df[[variable]] <- as.factor(df[[variable]])
  }
  return(df)
}

## Normalização (criando função para colocar os dados na mesma escala)
scale.features <- function(df, variables){
  for (variable in variables){
    df[[variable]] <- scale(df[[variable]], center=T, scale=T)
  }
  return(df)
}

# Normalizando as variáveis
numeric.vars <- c("credit.duration.months", "age", "credit.amount")
credit.df <- scale.features(credit.df, numeric.vars)

# convertendo variáveis numéricas categóricas para tipo fator
categorical.vars <- c('credit.rating', 'account.balance', 'previous.credit.payment.status',
                      'credit.purpose', 'savings', 'employment.duration', 'installment.rate',
                      'marital.status', 'guarantor', 'residence.duration', 'current.assets',
                      'other.credits', 'apartment.type', 'bank.credits', 'occupation',
                      'dependents', 'telephone', 'foreign.worker')

credit.df <- to.factors(df = credit.df, variables = categorical.vars)

# Dividindo os dados em treino e teste - 60:40 ratio
indexes <- sample(1:nrow(credit.df), size = 0.6 * nrow(credit.df))
train.data <- credit.df[indexes,]
test.data <- credit.df[-indexes,]

# Feature Selection
library(caret)
library(randomForest)

# Função para seleção de variáveis
run.feature.selection <- function(num.iters=20, feature.vars, class.var){
  set.seed(10)
  variable.sizes <- 1:10
  control <- rfeControl(functions = rfFuncs, method = "cv",
                        verbose = FALSE, returnResamp = "all",
                        number = num.iters)
  results.rfe <- rfe(x = feature.vars, y = class.var,
```

```
                sizes = variable.sizes,
                rfeControl = control)
  return(results.rfe)
}

# Executando a função
rfe.results <- run.feature.selection(feature.vars = train.data[,-1],
                                     class.var = train.data[,1])


# Visualizando os resultados
rfe.results
varImp((rfe.results))


# Criando e Avaliando o Modelo
library(caret)
library(ROCR)

# Biblioteca de utilitários para construção de gráficos
source("plot_utils.R")

## separate feature and class variables
test.feature.vars <- test.data[,-1]
test.class.var <- test.data[,1]

# Construindo um modelo de regressão logística
formula.init <- "credit.rating ~ ."
formula.init <- as.formula(formula.init)
lr.model <- glm(formula = formula.init, data = train.data, family = "binomial")

# Visualizando o modelo
summary(lr.model)

# Testando o modelo nos dados de teste
lr.predictions <- predict(lr.model, test.data, type="response")
lr.predictions <- round(lr.predictions)

# Avaliando o modelo (com uma Matriz de Confusão)
confusionMatrix(table(data = lr.predictions, reference = test.class.var), positive = '1')

## Feature selection (de maneira visual)
formula <- "credit.rating ~ ."
formula <- as.formula(formula)
control <- trainControl(method = "repeatedcv", number = 10, repeats = 2)
model <- train(formula, data = train.data, method = "glm", trControl = control)
importance <- varImp(model, scale = FALSE)
plot(importance)


### 2a Versão do Modelo (tentativa de aumento de acurácia)

# Construindo o modelo com as variáveis selecionadas
formula.new <- "credit.rating ~ account.balance + credit.purpose + previous.credit.payment.status + savings +
credit.duration.months"
formula.new <- as.formula(formula.new)
lr.model.new <- glm(formula = formula.new, data = train.data, family = "binomial")

# Visualizando o modelo
summary(lr.model.new)

# Testando o modelo nos dados de teste
lr.predictions.new <- predict(lr.model.new, test.data, type = "response")
```

```
lr.predictions.new <- round(lr.predictions.new)

# Avaliando o modelo
confusionMatrix(table(data = lr.predictions.new, reference = test.class.var), positive = '1')

# Avaliando a performance do modelo

# Criando curvas ROC
lr.model.best <- lr.model
lr.prediction.values <- predict(lr.model.best, test.feature.vars, type = "response")
predictions <- prediction(lr.prediction.values, test.class.var)
par(mfrow = c(1,2))
plot.roc.curve(predictions, title.text = "Curva ROC")
plot.pr.curve(predictions, title.text = "Curva Precision/Recall")

# Biblioteca para geração de curvas ROC

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

library(ROCR)

plot.roc.curve <- function(predictions, title.text){
  perf <- performance(predictions, "tpr", "fpr")
  plot(perf, col = "black", lty = 1, lwd = 2,
       main = title.text, cex.main = 0.6,
       cex.lab = 0.8, xaxs="i", yaxs="i")
  abline(0,1, col = "red")
  auc <- performance(predictions, "auc")
  auc <- unlist(slot(auc, "y.values"))
  auc <- round(auc,2)
  legend(0.4,0.4, legend = c(paste0("AUC: ",auc)), cex = 0.6, bty = "n", box.col = "white")

}

plot.pr.curve <- function(predictions, title.text){
  perf <- performance(predictions, "prec", "rec")
  plot(perf, col = "black", lty = 1, lwd = 2,
       main = title.text, cex.main = 0.6, cex.lab = 0.8, xaxs = "i", yaxs = "i")
}
```

Capítulo 19:

```
# Mapeando a ocorrência do vírus Zika

# Obs: Caso tenha problemas com a acentuação, consulte este link:
# https://support.rstudio.com/hc/en-us/articles/200532197-Character-Encoding

# Configurando o diretório de trabalho
# Coloque entre aspas o diretório de trabalho que você está usando no seu computador
setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Projetos/Mini-Projeto05")
getwd()

# http://combateaedes.saude.gov.br/pt/situacao-epidemiologica

# Carregando os pacotes
# devtools::install_github("wch/webshot")
library(dplyr)
library(ggplot2)

# Listando os arquivos e gerando uma lista com os respctivos nomes
temp_files <- list.files(pattern = ".csv")
temp_files

# Carregando todos os arquivos em um único objeto
myfiles <- lapply(temp_files, read.csv, stringsAsFactors = FALSE)

# Resumo dos arquivos
str(myfiles, 1)
lapply(myfiles, names)[1]
lapply(myfiles, head,2)[1:2]

# Organizando o shape dos dados
brazil <- do.call(rbind, myfiles)
View(brazil)
brazil <- brazil %>%
  mutate(report_date = as.Date(report_date))

# Visualizando o dataset
glimpse(brazil)

# Transformando o dataframe um uma tabela dplyr e removendo as colunas 5 a 7
brazil <- brazil %>% select(-(6:7))

# Visualizando as primeiras 20 linhas
brazil %>% slice (1:20)

# Para cada reporting_date nós temos 5 regiões
brazil %>% filter(location_type == "region")
brazil %>% filter(location_type == "region") %>%
  ggplot(aes(x = report_date, y = value, group = location, color = location)) +
  geom_line() +
  geom_point() +
  ggtitle("Casos de Zika por Região do Brasil")


# Separando as regiões e Visualizando os Dados
region <- brazil %>%
  filter(location_type == "region")

region %>%
  ggplot(aes(x =location, y = value)) + geom_bar(stat = "identity") +
  ylab("Número de Casos Reportados") + xlab("Region") +
```

```
  ggtitle("Casos de Zika Reportados no Brasil")

region %>%
  slice(1:length(unique(region$location))) %>%
  arrange(desc(value)) %>%
  mutate(location = factor(location, levels = location,ordered = TRUE)) %>%
  ggplot(aes(x = location, y = value)) + geom_bar(stat = "identity") +
  ylab("Número de Casos Reportados") + xlab("Region") +
  ggtitle("Casos de Zika Reportados no Brasil")

# Obtendo localidades únicas
region %>%
  slice(1:length(unique(region$location)))

# Organziando as localidades únicas por número de casos reportados
region %>%
  slice(1:length(unique(region$location))) %>%
  arrange(desc(value))

# Criando variáveis do tipo fator
region %>%
  slice(1:length(unique(region$location))) %>%
  arrange(desc(value)) %>%
  mutate(location = factor(location,levels=location,ordered=TRUE)) %>%
  glimpse()

# Agrupando o Sumarizando
brazil_totals <- brazil %>% filter(location=="Brazil")
region_totals <- brazil %>% filter(location_type=="region") %>%
  group_by(report_date,location) %>%
  summarize(tot = sum(value))

# Padronizar os dados e remover as sumarizações
regvec <- vector()
length(regvec) <- nrow(brazil)
for (ii in 1:nrow(brazil)) {
  if (brazil[ii,]$location_type != "region")  {
    regvec[ii] <- newlab
  } else {
    newlab <- brazil[ii,]$location
    regvec[ii] <- newlab
  }
}

# Agregando o vetor de regiões ao dataframe brasil
statedf <- cbind(brazil,regvec)

# Eliminar o sumário de linhas por região e país
statedf <- statedf %>% filter(location != "Brazil")
statedf <- statedf %>% filter(location_type != "region")

# Gerar o total por regiões a partir dos dados transformados
statedf %>% group_by(report_date,regvec) %>%
  summarize(tot=sum(value)) -> totals

# Gerando os mapas de cada estado do Brasil
library(ggmap)
longlat <- geocode(unique(statedf$location)) %>%
  mutate(loc = unique(statedf$location))

# Salvando os geocodes do dataframe statedf e salvando em um novo dataframe chamado formapping
statedf %>% filter(as.character(report_date) == "2016-06-11") %>%
  group_by(location) %>% summarize(cases = sum(value)) %>%
```

```
  inner_join(longlat, by = c("location" = "loc")) %>%
  mutate(LatLon = paste(lat, lon, sep = ":")) -> formapping

# Visualizando os dados
head(formapping)

# Formatando a saída e gerando um movo dataframe chamado long_formapping
num_of_times_to_repeat <- formapping$cases
long_formapping <- formapping[rep(seq_len(nrow(formapping)),
                                  num_of_times_to_repeat),]

# Visualizando os dados
head(long_formapping)

# Instalando o pacote leaflet
install.packages("leaflet")
library(leaflet)

# Gerando o mapa com o dataframe
# Aplique o zoom
leaflet(long_formapping) %>%
  addTiles() %>%
  addMarkers(clusterOptions = markerClusterOptions())
```