Coleta e Preparação de Dados

Bootcamp Engenheiro(a) de Dados

Walisson Ferreira de Carvalho

2021

Coleta e Preparação de Dados

Bootcamp Engenheiro(a) de Dados

Walisson Ferreira de Carvalho

Sumário

Capítulo 1. Introdução

Nos últimos anos, o volume de dados disponível para análise cresceu exponencialmente. É possível encontrar bases de dados com mais de 29 milhões de atributos. Por exemplo, a base KDD2010 do repositório LIBSVM possui 29.980.095 atributos. Essa explosão do volume de dados acabou por originar um novo conceito, o *Big Data*. O termo *Big Data* foi utilizado pela primeira vez no artigo *"Visually exploring gigabyte data sets in real time"*, publicado em 1999 (BRYSON *et al.*, 1999). Em seu glossário de Tecnologia da Informação, o *Gartner Group* define *Big Data* como um grande **V**olume, grande **V**ariedade e grande **V**elocidade de dados que precisa de formas inovadoras para análise. Os 3Vs destacados pelo *Gartner* representam as características principais do *Big Data*.

Mais recentemente, dois novos Vs foram inseridos entre as características do *Big Data*, a **V**eracidade e o **V**alor dos dados. Para agregar valor aos dados, é necessário transformar o *Big Data* em *Smart Data*. O *Gartner*, em 2015, definiu o *Smart Data* como *"Smart data discovery is a next-generation data discovery capability that provides business users or citizen data scientists with insights from advanced analytics."* Portanto, a descoberta do *Smart Data* é a extração de informação útil a partir o *Big Data.*

Triguero et al. (2019) destacam que o dado somente terá valor se for extraído conhecimento a partir dele. No processo de transformação de dado em informação (Figura 1), é necessário garantir a qualidade dos dados, entrada e processo, para ter qualidade no resultado obtido. O princípio GIGO (*Garbage in Garbage Out*) já mostra que o grande volume de dados não é garantia de qualidade de dados e, consequentemente, não necessariamente produz informação útil.

Além do grande volume, a característica variedade também merece destaque nesse material, pois está relacionada com a heterogeneidade dos dados, ou seja, os dados estão em diversos formatos estruturados e não estruturados. Dados não estruturados são dados produzidos nos formatos de vídeos, imagens, áudios, textos, entre outros formatos. A Figura 2, extraída do IDC, exibe o crescimento do volume de dados não estruturados.

Figura 1 - Transformação de dados em informação.

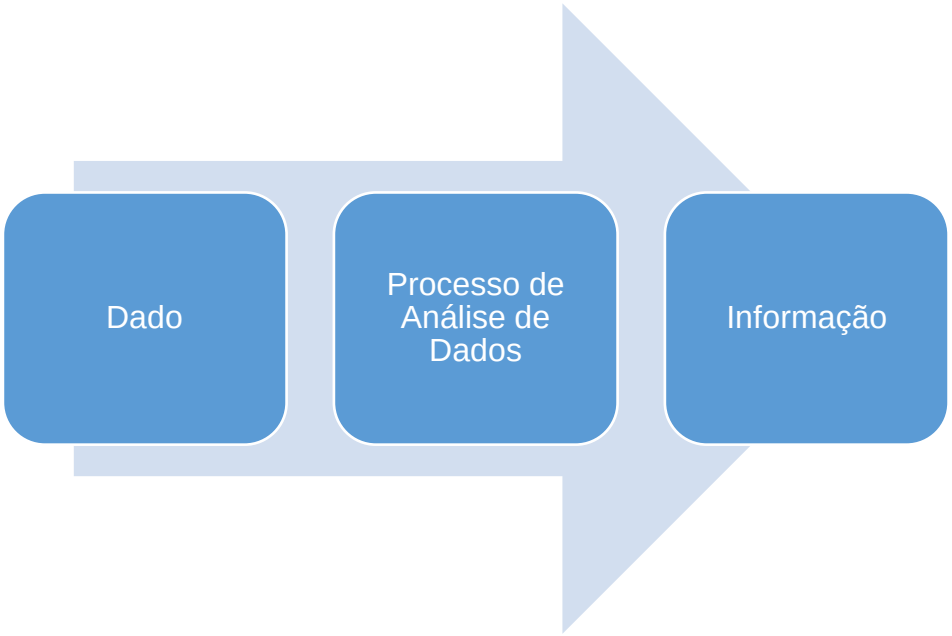
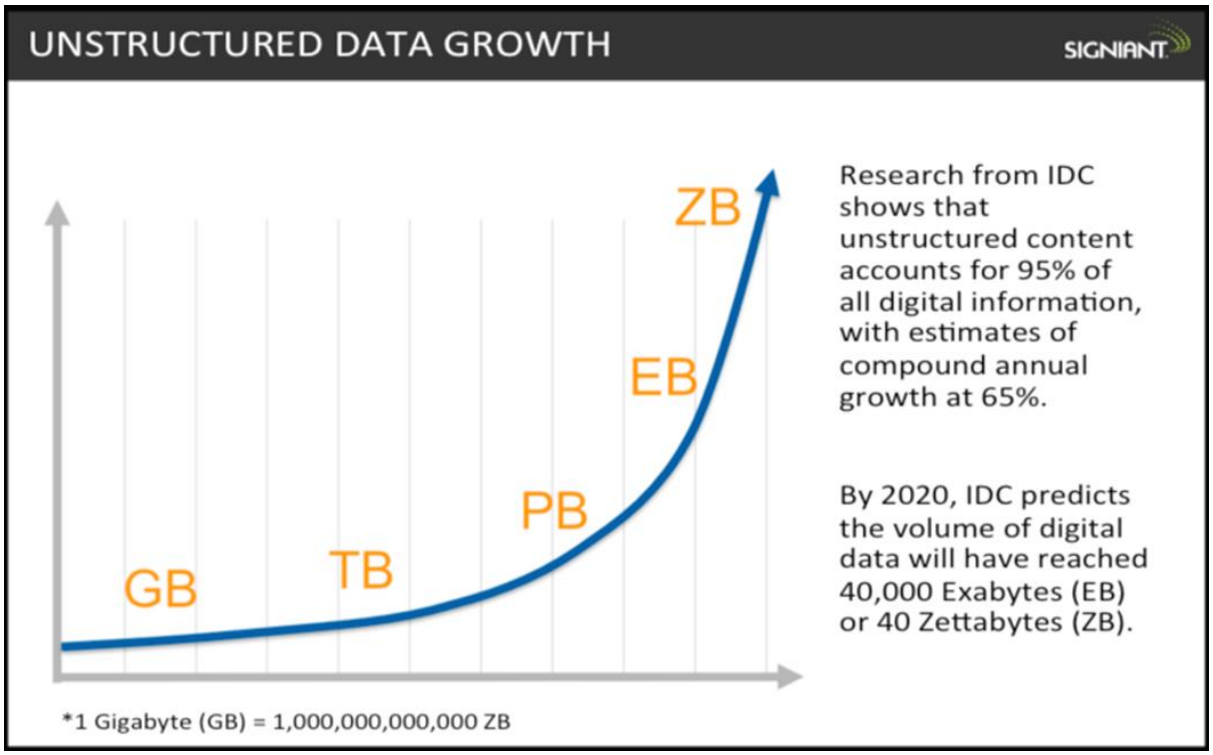


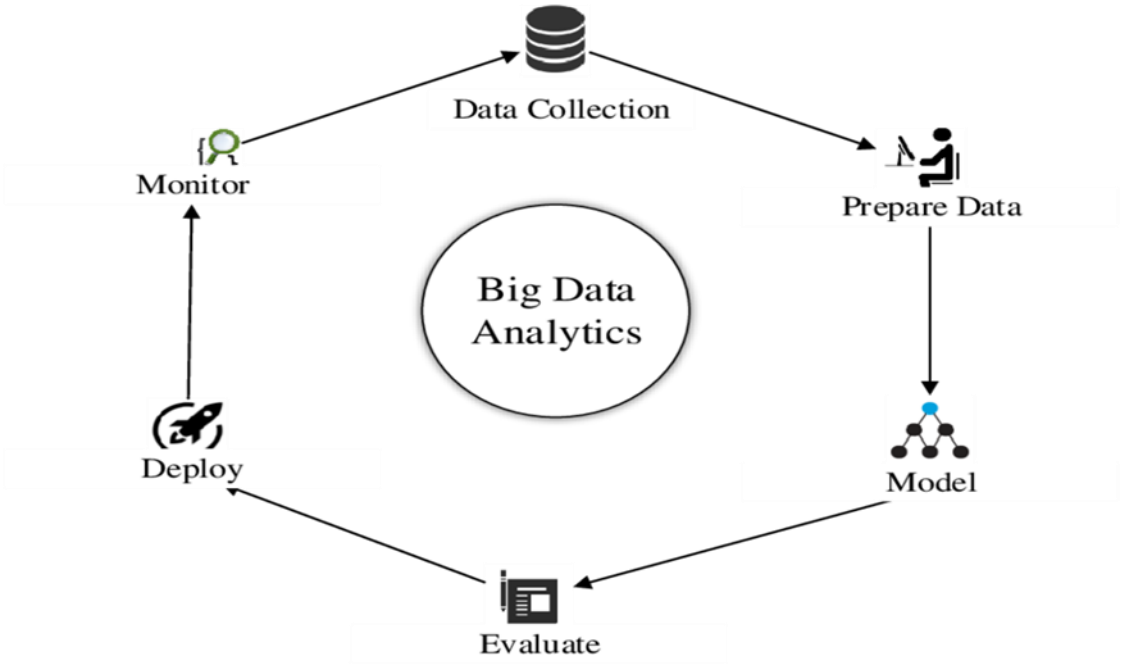
Figura 2 - Crescimento de dados não estruturados.



Fonte: IDC (Instituto Contemporâneo de Direito).

Conforme exibido nas Figuras 3 e 4, a tarefa de preparação de dados está inserida no processo de Análise de Dados, assim como no processo de Mineração de Dados. É importante destacar que essa fase de preparação de dados muitas vezes é chamada de pré-processamento.

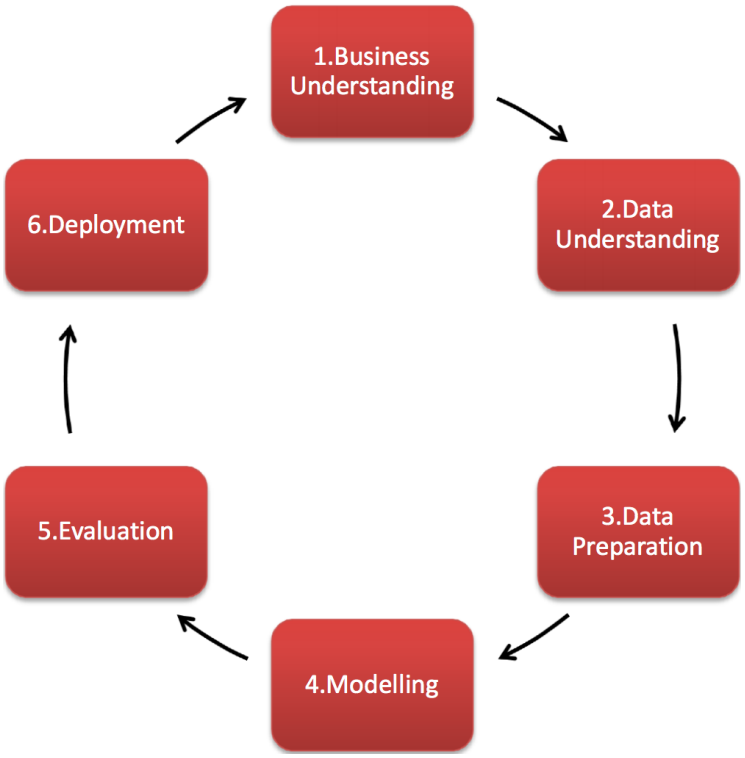
Figura 3 - Etapas do processo de Big Data Analytics.



Fonte: Rehman et al. (2016).

Pode-se definir a preparação de dados como um passo obrigatório dentro do processo de análise de dados que visa converter dados brutos em dados que se adequam aos algoritmos de análise de dados.

Figura 4 - Metodologia CRISP de Data Mining.



Fonte: https://www.sv-europe.com/crisp-dm-methodology/.

Trigueiro *et al.* (2019) destacam que a fase de preparação de dados é claramente a fase mais importante no processo de análise de dados, pois os dados brutos, por natureza, não estão prontos para serem analisados. Assim, somente com um bom processo de preparação dos dados é possível a extração informação útil a partir dos dados.

Além da importância na transformação dos dados brutos em *Smart Data,* a fase de preparação de dados consome o maior esforço de todo processo de análise de dados. Segundo estudos divulgados pela Forbes, 80% do esforço do analista de dados é dedicado a preparação dos dados para análise.

1.1. A linguagem R

Essa seção do material não pretende ser um material de referência da linguagem R, mas apresentar um pequeno panorama da linguagem e seus principais conceitos. No site do *The Comprehensive R Archive Network*, estão disponíveis, além do próprio R, materiais de apoio, inclusive em português, manuais, códigos, FAQs, entre outros recursos.

A linguagem R foi desenvolvida, a partir da linguagem S, no departamento de estatística da Universidade Auckland na Austrália. O R é uma linguagem de programação de código aberto, sob os termos da *Free Software Foundation's GNU General Public*. O ambiente R está disponível para diversas plataformas do Linux, MAC e Windows.

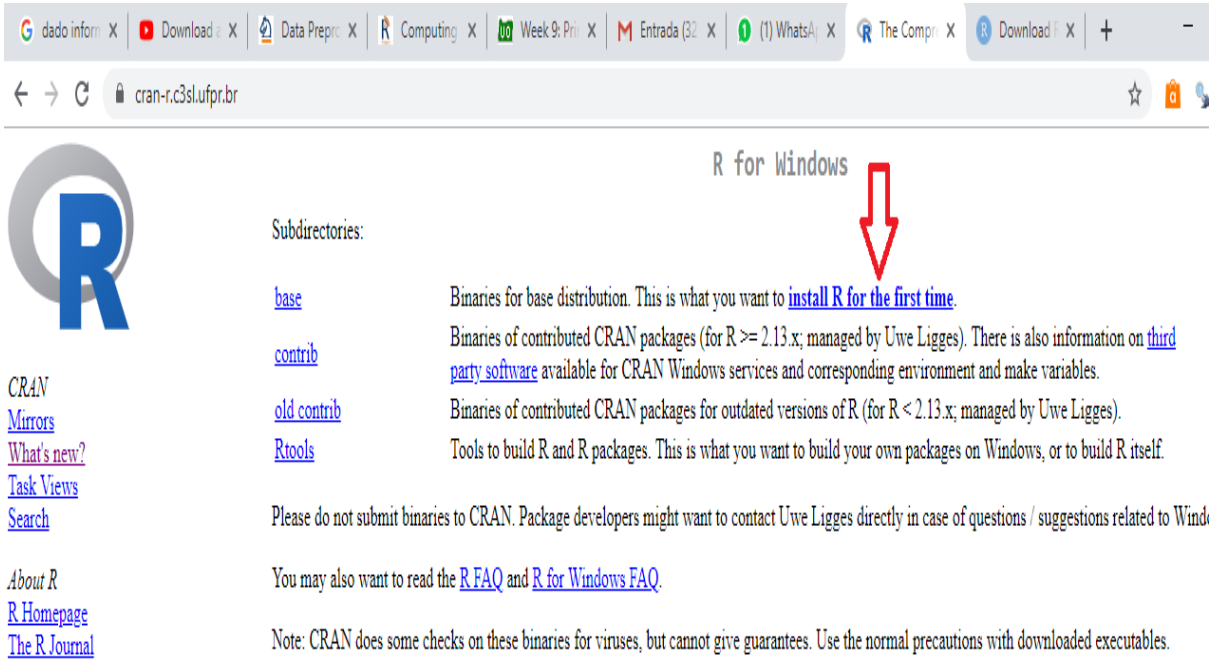
De acordo com o site do R, a linguagem R é um ambiente integrado para manipulação de dados, cálculos e visualização gráfica. Para uso mais eficiente do R é recomendado, embora não necessário, que se instale uma IDE (***I****ntegrated* ***D****evelopment* ***E****nvironment*). Para esse curso, recomenda-se a instalação do RStudio.

A instalação do R é bastante simples. Incialmente deve-se fazer o download do ambiente R, seguindo os passos abaixo:

1. Acessar o site https://cran.r-project.org/index.html;
2. Selecionar a opção Mirror;
3. Selecionar o servidor mais próximo da sua localização;
4. Selecionar o ambiente operacional onde o R será instalado;
5. Clicar no link Install R for the first time, conforme Figura 5;
6. Efetuar o download do aplicativo.

Após efetuar o download, será iniciado o processo de instalação e é recomendável que seja instalada a versão padrão, pois não é necessária nenhuma configuração especial.

Figura 5 - Instalação do R.



Fonte: cran-r.project.org.

Uma vez instalado o R, pode-se instalar o RStudio que está disponível em: https://www.rstudio.com/products/rstudio/download/. Para o desenvolvimento desse curso, a versão *RStudio Desktop Open Source License* (FREE) atende perfeitamente. Após o download do RStudio, basta seguir o caminho padrão de instalação, pois assim como no R não é necessária nenhuma configuração especial.

A linguagem R é estruturada por pacotes e atualmente possui mais de 14 mil pacotes abertos e acessíveis para toda comunidade. A listagem dos pacotes disponíveis está disponibilizada no site do https://cran.r-project.org/, na opção *packages*. Para instalar um pacote, usando o RStudio, basta acessar a opção *tools→Install Packages*, conforme exibido na Figura 6.

Ao selecionar a opção *tools→Install Packages*, será exibida a tela conforme exibido na Figura 7, nessa tela basta informar o nome do pacote desejado e marcar opção *Install dependencies*. Em relação ao local de instalação, *Install to Library*, basta deixa o padrão de instalação.

Figura 6 - Instalação de pacotes no R.

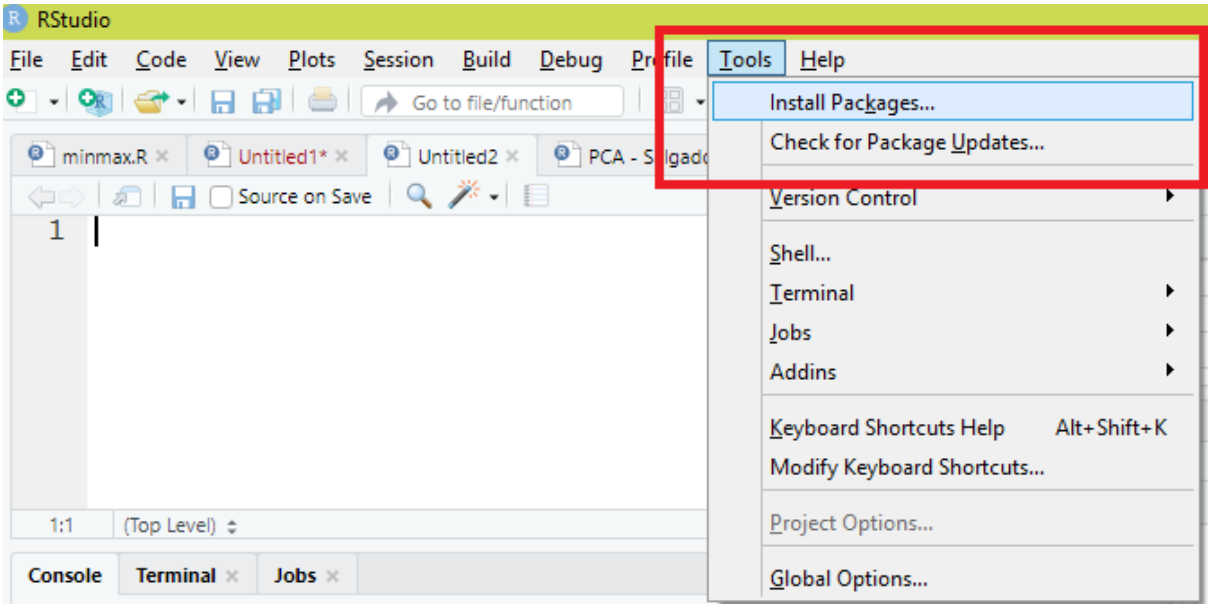
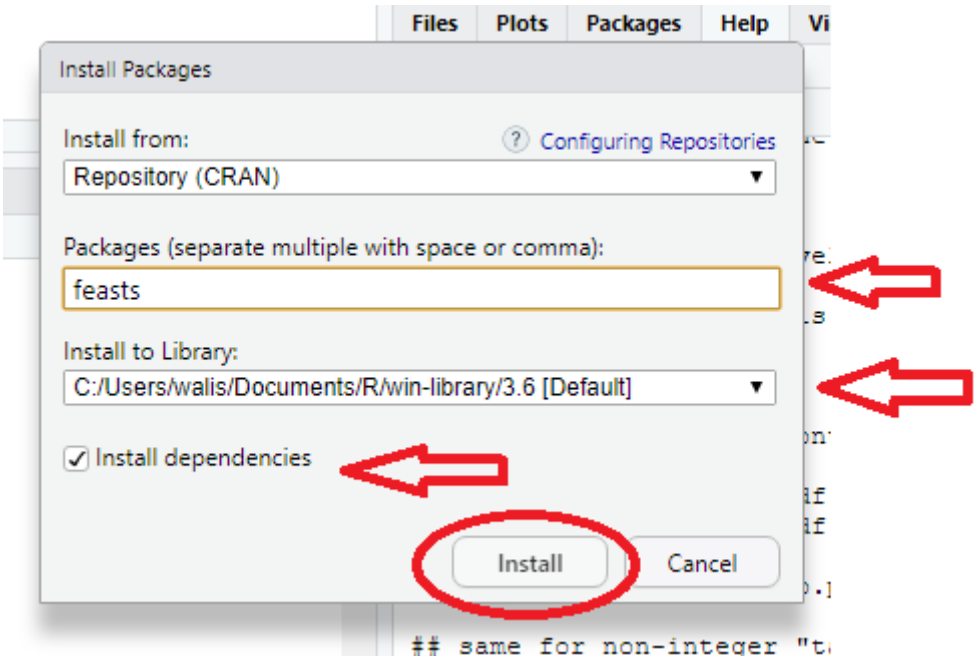


Figura 7 - Instalação de pacotes no R.



Após instalar o pacote, para utilizar o pacote, é necessário invocar o pacote através do comando *library*, *library(nome do pacote).*

O R é uma linguagem “não tipada”, ou seja, não é necessário declarar as variáveis antes de usá-las e nem definir o tipo da variável. A linguagem R é *case*

sensitive, ou seja, letras maiúsculas e minúsculas são diferentes para o R, assim a variável x é diferente da variável X. A linguagem R possui os operadores relacionais, lógicos e aritméticos, que permitem as operações lógicas e aritméticas como qualquer outra linguagem.

A linguagem R possui um grande número de funções que são organizadas dentro dos pacotes. O uso básico da função é seu *some* e sua lista de parâmetros. Por exemplo, a função *sqrt* extrai a raiz quadrada de um número. Assim, *sqrt(10)* extrai a raiz quadrada do número 10.

O R possui uma boa documentação sobre suas funções, sendo que para consultar o *help* de uma função basta digitar "*?nome da função*", no console do R ou do RStudio, por exemplo "*?sqrt*".

Importante destacar que a maioria das funções da linguagem R retorna um objeto do tipo lista. Além das funções que são tratadas como objetos no R, a linguagem possui mais quatro tipos de objetos, sendo eles:

1. **Vetores**: sequência de valores que podem ser numéricos ou caracteres;
2. **Matrizes**: conjunto de vetores em linhas e colunas, sendo que todos os vetores devem ser do mesmo tipo;
3. **Dataframes**: são como matrizes, mas os vetores que os compõem podem possuir tipos diferentes. Como as bases de dados possuem tipos diferentes de atributos, os dataframes são usados para armazenar as bases de dados.
4. **Listas**: conjunto de vetores, matrizes ou dataframes, que não precisam ter o mesmo tamanho.

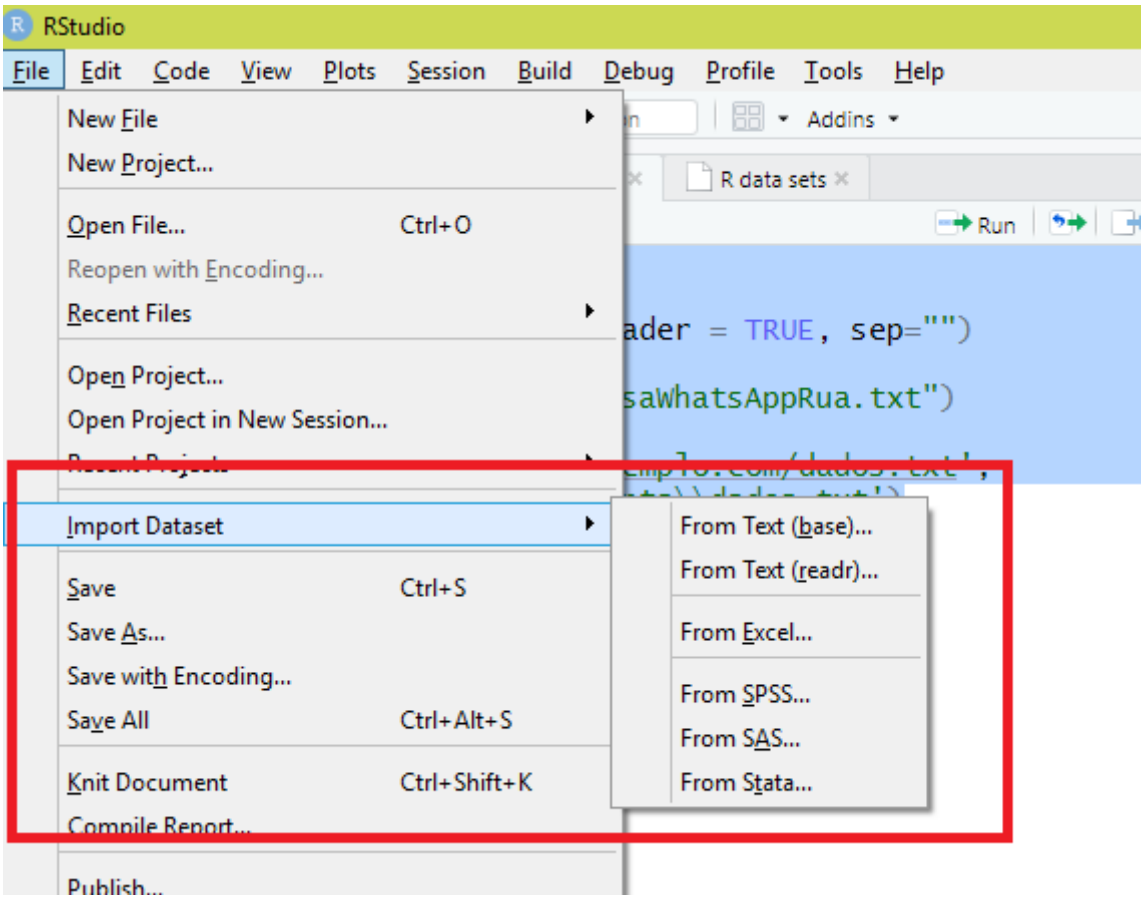
Outro ponto que merece destaque nesse material é o uso das bases de dados do R. O R oferece em seus pacotes, além de uma gama de funções, bases de dados para serem utilizadas. Para visualizar as bases instaladas em seu ambiente, basta aplicar o comando data() que serão exibidas todas as bases de dados agrupadas por pacotes, instalados no ambiente R. A cada novo pacote instalado, novas bases de

dados são instaladas. Para colocar uma base de dados em uso, deve-se digitar o seguinte comando: data(nome da base de dados desejada).

Além das próprias bases do R, é possível importar bases de dados em diversos formatos, tais como xls, txt, SAS, entre outros. A forma mais fácil de fazer essa importação é a partir da IDE RStudio.

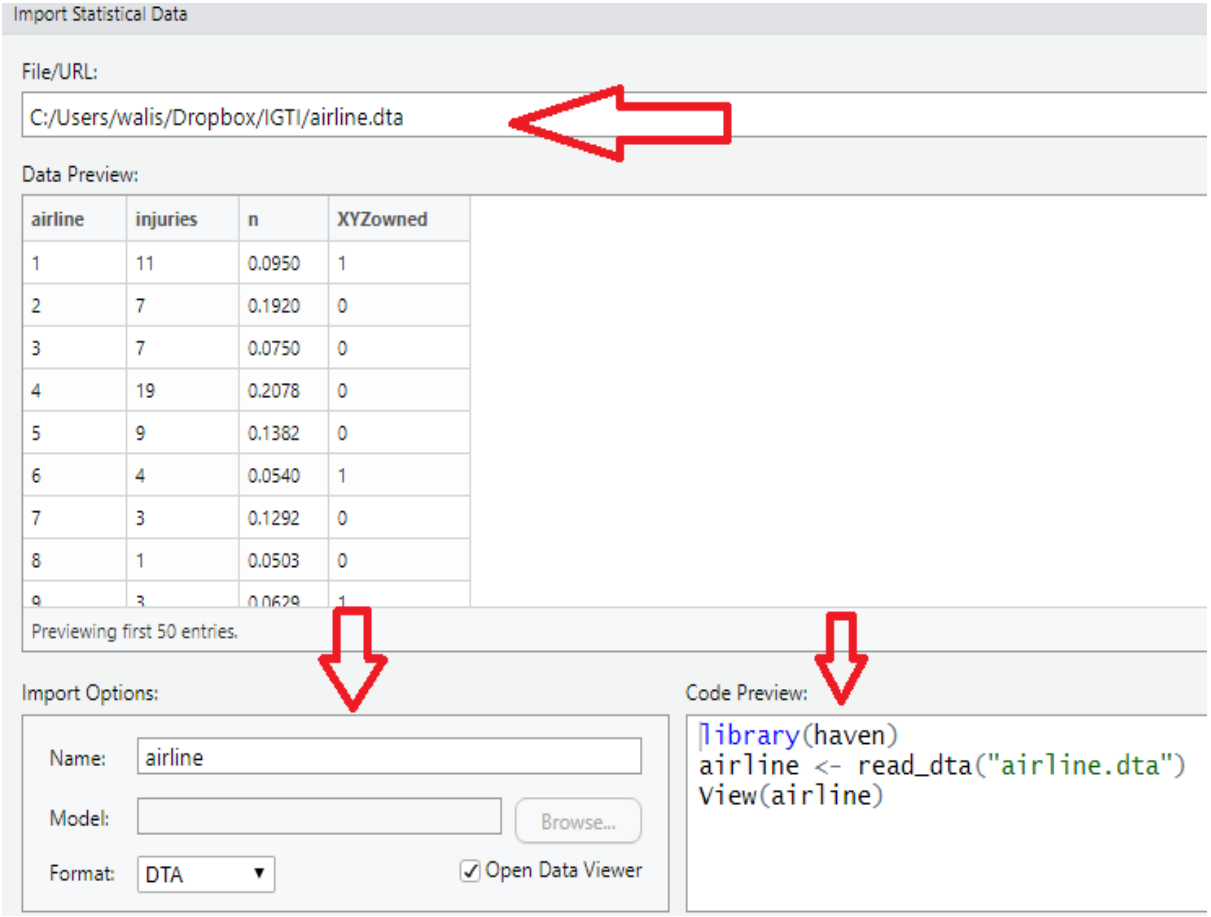
Para importar uma base no RStudio, deve-se selecionar *file→Import Dataset* e selecionar o tipo de base a ser importada, conforme mostrado na Figura 8.

Figura 8 - Importação de base de dados.



Após selecionar a opção e o tipo de importação a ser realizada (Figura 8), será exibida uma janela (Figura 9), para que seja informado o local onde a base de dados se encontra e o nome do *dataframe,* que armazenará os dados. Além disso, essa janela exibe uma pré-visualização dos dados que serão carregados, assim com o trecho de código para importação.

Figura 9 - Importação de base de dados.



Uma vez carregada a base de dados, ela estará disponível para entendimento, preparação e análise dos dados. A Figura 10 exibe alguns comandos simples para compreensão da base de dados. O comando *dim* exibe as dimensões, linhas e colunas, da base. Já o comando *head* exibe os primeiros registros da base em questão. Por sua vez, o comando *summary* resume os dados de cada atributo da base de dados alvo de análise.

Figura 10 - Entendendo os dados.

```
> data("airquality")
> dim(airquality)
[1] 153   6
> head(airquality)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
> summary(airquality)
     Ozone           Solar.R           Wind             Temp      
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00  
 1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00  
 Median : 31.50   Median :205.0   Median : 9.700   Median :79.00  
 Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88  
 3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00  
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00  
 NA's   :37       NA's   :7                                       
     Month            Day      
 Min.   :5.000   Min.   : 1.0  
 1st Qu.:6.000   1st Qu.: 8.0  
 Median :7.000   Median :16.0  
 Mean   :6.993   Mean   :15.8  
 3rd Qu.:8.000   3rd Qu.:23.0  
 Max.   :9.000   Max.   :31.0  
```

Capítulo 2. Coleta de Dados

Conforme apresentado na unidade 1 desse material, antes de iniciar o processo de preparação de dados é necessário identificar as fontes e realizar a coleta dos dados. Essa coleta deve estar sempre alinhada com os objetivos da análise de dados como um todo.

Para realizar a coleta de dados é necessário identificar as fontes, que podem ser internas ou externas. As fontes internas referem-se aos dados disponíveis na organização, tais como dados oriundos dos CRMs, ERPs, SCMs entre outros sistemas de processamento de transação usados pela empresa; os dados externos são os dados disponíveis na web, dados adquiridos junto às empresas especializadas e dados abertos disponibilizados, por exemplo, por órgãos governamentais.

Conforme definição do *Open Data Handbook*[1] "dados abertos são dados que podem ser livremente usados, reutilizados e redistribuídos por qualquer pessoa - sujeitos, no máximo, à exigência de atribuição da fonte e compartilhamento pelas mesmas regras."

Existem diversas fontes de dados abertos no Brasil e no mundo, entre elas pode-se destacar:

1. *http://dados.gov.br/*.
2. *http://www.ipeadata.gov.br*.
3. *https://data.europa.eu/euodp/en/home*.
4. *https://data.gov.uk/*.
5. *https://registry.opendata.aws/*.

[1] http://opendatahandbook.org

Outra fonte que merece destaque é a web. Segundo o dicionário de Tecnologia da Informação do Gartner Group (2015), "*your company's biggest database isn't your transaction, CRM, ERP or other internal database. Rather it's the Web itself…Treat the Internet itself as your organization's largest data source*".[2] Ou seja, o Gartner enfatiza que a principal fonte de dados é externa, mais especificamente a principal fonte é a web.

Para recuperação dos dados na web um procedimento comumente utilizado é o w*eb scraping* que, em poucas palavras, significa vários métodos para recuperar informações na web. Para a realização de *scraping,* é possível contratar soluções prontas no mercado ou desenvolver soluções, usando a linguagem R ou Phyton por exemplo.

Para realizar scraping numa base de dados é necessário conhecer um pouco sobre HyperText Markup Language (HTML), para identificar os campos (TAGs) que contêm as informações desejadas. Para identificação das TAGs no site é necessário inspecionar o site, a Figura 11 exibe um exemplo de inspeção de site. Observe através da Figura 11 que o nome do filme está armazenado no campo *"lister-item-content h3 a*" e o ano do filme no campo "*lister-item-content h3 .lister-item-year*".

Outro fator importante no *scraping* é o arquivo *robots.txt*, que exibe as pastas de um site que estão disponíveis, ou não, para realização da recuperação de informação. A Figura 12 exibe o arquivo robots.txt do LinkedIn. A análise desse arquivo mostra, por exemplo, que a pasta *analytics* está desabilitada para *scraping,* o *Googlebot.*

A linguagem R fornece o pacote *rvest* para fazer *scraping* na WEB, entre as funções desse pacote pode-se destacar:

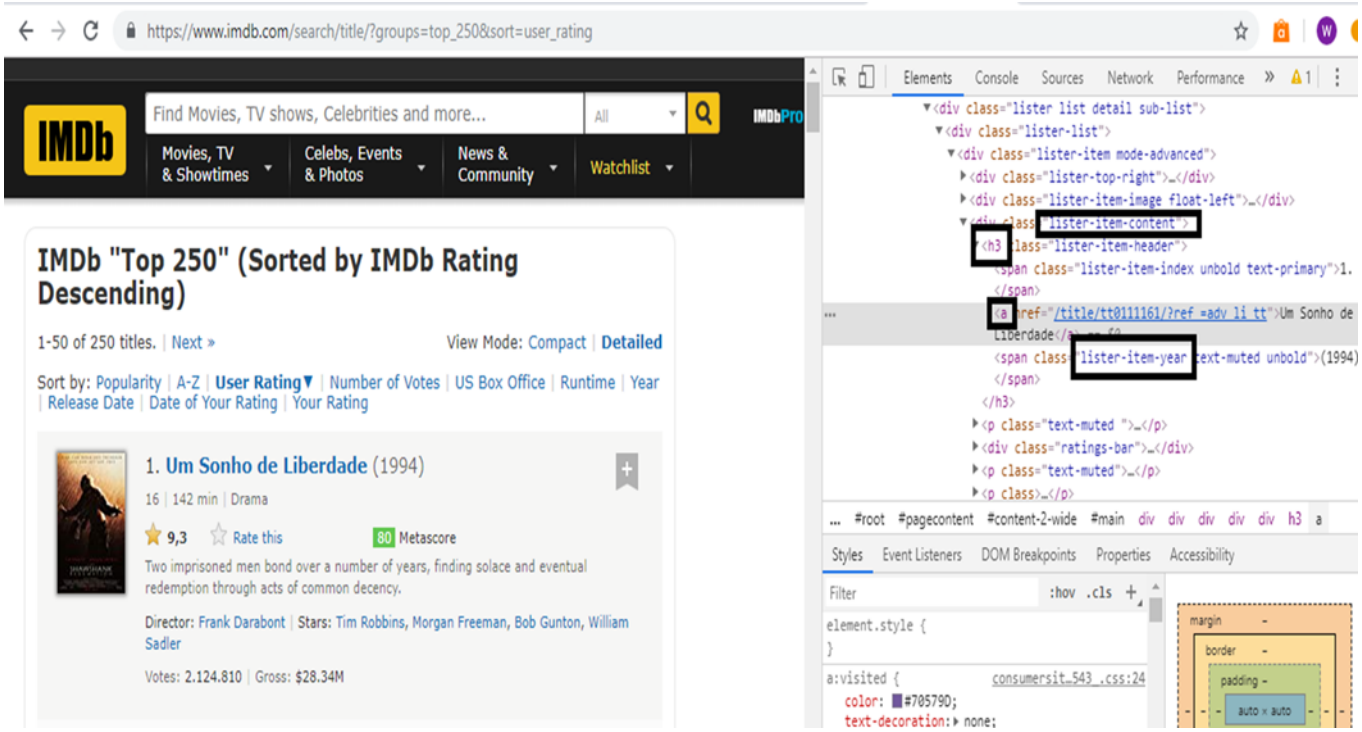
1. read_html(): faz a leitura da webpage para o R;

[2] www.gartner.com

2. html_nodes(): extrai partes do HTML;

3. html_text(): extrai textos do HTML.

Figura 11 - Inspeção de site.



Para verificar se um diretório pode ou não ser vasculhado por uma um script em R, deve-se utilizar a função paths_allowed. Por exemplo, o comando *paths_allowed(paths = c("https://www.imdb.com/ads/"))*, que retorna verdadeiro ou falso caso a pasta ads do site https://www.imdb.com esteja liberada ou não para raspagem.

O código implementado em R, exibido na Figura 12, realiza a recuperação de dados do site https://www.imdb.com. Inicialmente foi verificada a permissão, através do acesso ao arquivo robots.txt, para a realização ou não do *scraping*. Após validar a permissão, o site é carregado para uma variável imdb e o conteúdo das variáveis nome e ano do filme são recuperadas.

Figura 12 - Arquivo robots.txt do LinkedIn.

```
# Notice: The use of robots or other automated means to access LinkedIn without
# the express permission of LinkedIn is strictly prohibited.
# See https://www.linkedin.com/legal/user-agreement.
# LinkedIn may, in its discretion, permit certain automated access to certain LinkedIn pages,
# for the limited purpose of including content in approved publicly available search engines.
# If you would like to apply for permission to crawl LinkedIn, please email whitelist-crawl@linkedin.com.
# Any and all permitted crawling of LinkedIn is subject to LinkedIn's Crawling Terms and Conditions.
# See http://www.linkedin.com/legal/crawling-terms.

User-agent: Googlebot
Disallow: /addContacts*
Disallow: /addressBookExport*
Disallow: /analytics/
Disallow: /answers*
Disallow: /authwall
Disallow: /badges/profile/create
Disallow: /cap/
Disallow: /companyDir*
Disallow: /connections*
Disallow: /csp/
Disallow: /edurec*
Disallow: /embed/feed/update/
Disallow: /endorsements
Disallow: /feed/update/
Disallow: /find/
Disallow: /groupAnswers*
Disallow: /groupSharingMsg*
Disallow: /inviteFromProfile*
Disallow: /inviteMany*
Disallow: /jobs?runSearch*
Disallow: /jsearch*
```

Fonte: https://www.linkedin.com/robots.txt.

O resultado do *scraping* é exibido na Figura 13. Uma vez coletado, é possível armazenar esses dados em um banco de dados relacional ou mesmo um NoSQL, como por exemplo o MongoDB.

Figura 13 - Implementação R para scraping.

```
1  library(rvest)
2  paths_allowed(
3    paths = c("https://www.imdb.com/search/title?groups=top_250&sort=user_rating")
4  )
5  imdb <- read_html("https://www.imdb.com/search/title?groups=top_250&sort=user_rating")
6  titulo = imdb %>%
7  html_nodes(".lister-item-content h3 a") %>%
8  html_text()
9  ano = imdb %>%
10 html_nodes(".lister-item-content h3 .lister-item-year") %>%
11 html_text() %>%
12 str_sub(start = 2, end = 5) %>%
13 as.Date(format = "%Y") %>%
14 year()
15 top_50 <- tibble(Titulo = titulo, Ano = ano)
```

Figura 14 - Resultado do scraping.

```
# A tibble: 50 x 2
   Titulo                                  Ano
   <chr>                                 <dbl>
 1 Um Sonho de Liberdade                  1994
 2 O Poderoso Chefão                      1972
 3 Batman: O Cavaleiro das Trevas         2008
 4 O Poderoso Chefão II                   1974
 5 O Senhor dos Anéis: O Retorno do Rei   2003
 6 Pulp Fiction: Tempo de Violência       1994
 7 A Lista de Schindler                   1993
 8 12 Homens e uma Sentença               1957
 9 A Origem                               2010
10 Clube da Luta                          1999
# ... with 40 more rows
>
```

Capítulo 3. Limpeza de Dados

Durante a fase de preparação de dados, um dos procedimentos a ser realizado na base de dados é a limpeza dos dados, uma vez que os dados brutos não estão prontos para análise. Portanto, se a sujeira dos dados é um problema inevitável, a limpeza dos dados precisa ser realizada pelo analista de dados.

Existem diversas fontes de dados sujos, entre elas pode-se destacar: dados ausentes, erros de entrada de dados, erros de transmissão, *bugs* nos sistemas transacionais, entre outros.

Os dados ruidosos podem ser classificados em dois grandes grupos: ruído de classe (*Class Noise)* e ruído de atributo (*Attribute Noise*).

Ruído de classe, também conhecido como ruído de rótulo (*label noise*), ocorre quando uma instância é rotulada incorretamente. Existem dois tipos de *class noise,* exemplos contraditórios e erro de classificação. A Figura 15 exibe um exemplo contraditório, destacado em vermelho. Nesse exemplo as primeiras instâncias possuem os mesmos valores para os atributos explicativos (atributo 1 e atributo 2), mas o atributo classe possui valor contraditório, pois para a primeira o valor é positivo e, para a segunda, negativo.

O ruído de atributo refere-se à corrupção de dados em atributos, corrupção essa que pode ser um dado ausente, valor digitado errado na entrada ou qualquer outra impureza que está no dado. Importante também inserir a análise do *outlier* para limpeza de dados, pois o valor errado pode ser um *outlier* que pode ser o alvo da análise.

As seções a seguir desse capítulo tratam dos dados ausentes e dos outliers.

Figura 15 - Dados sujos.

Atributo 1	Atributo 2	Classe
0.25	Vermelho	Positivo
0.25	Vermelho	Negativo
0.99	Verde	Negativo
1.02	Verde	Negativo
2.05	?	Negativo
?	Verde	Positivo
200.05	Verde	Positivo

3.1. Dados ausentes

Dados ausentes é um problema investigado há muitos anos pela comunidade científica e é um dos processos mais comuns durante a etapa de preparação de dados. A ausência de dados pode ocorrer por uma série de fatores, tais como a recusa de informar um determinado dado, o desconhecimento de um dado, erro na entrada de dados, entre outros casos.

A ocorrência de dados ausentes pode incidir em um atributo, em vários atributos, em atributos alvos ou não. A existência de dados ausentes pode causar diversas dificuldades para o analista de dados, tais como:

1. Perda de eficiência;
2. Dificuldades na análise e preparação dos dados;
3. Viés resultante da diferença da análise, com dados ausentes e dados completos.

Um marco no tratamento de dados ausentes é o trabalho de Donald Rubin (1976), que definiu uma taxonomia para a produção dos dados ausentes. O autor classificou a produção dos dados ausentes em MCAR (Missing Completely at Random), MAR (Missing at Random) e NMAR (not missing at random).

Considerando que X_{obs} represente os atributos com valores observados (não ausentes) e X_{mis} os dados ausentes. Se os dados ausentes dependem de X_{obs}, mas não dependem de X_{mis}, ocorre o MAR. Em outras palavras, a ausência de dados no atributo Y (X_{mis}) depende de outras variáveis (X_{obs}), mas não depende do próprio Y(X_{mis}). Por exemplo, uma pessoa do sexo feminino que não deseja informar a idade, ou seja, o dado ausente idade, depende do sexo, mas não depende da própria idade.

O MCAR ocorre quando a produção de dados ausentes em X_{mis} não depende de X_{obs} nem do próprio dado ausente. Assim, no MCAR o dado ausente não está relacionado com nenhum outro atributo da base e nem com o próprio atributo.

No terceiro e último método é NMAR, existe uma razão para o dado ausente, nesse caso a ausência está relacionada ao próprio atributo. Por exemplo, alguém pode preferir não informar o próprio salário quando o valor é muito alto ou muito baixo.

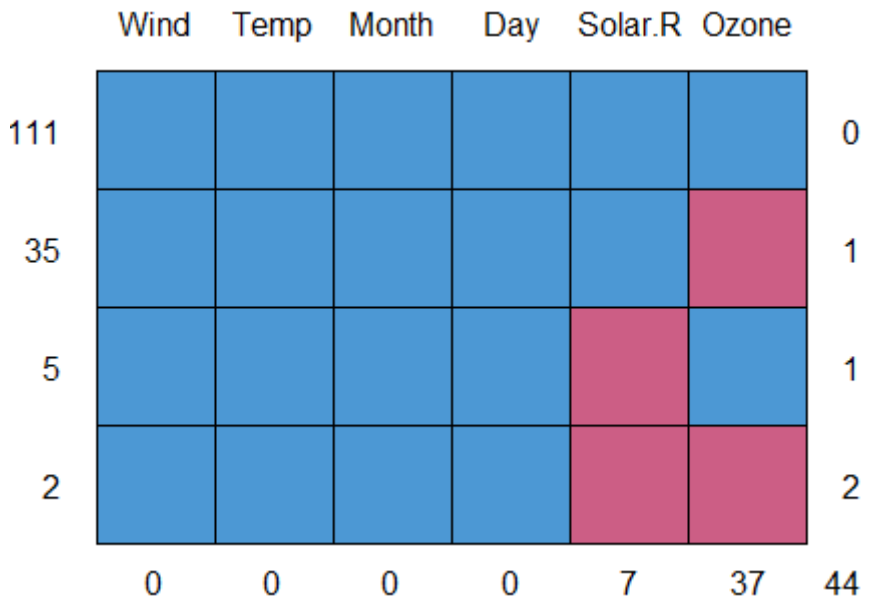
O R oferece vários pacotes para identificação e tratamento de dados ausentes. No exemplo apresentado na imagem a seguir é utilizado o pacote *mice* e a função *md.pattern*. Nesse exemplo a função foi aplicada na base *airquality*, que possui 153 instâncias e 6 atributos. Os atributos Solar R e Ozone possuem dados ausentes, sendo que 37 instâncias não possuem dados sobre ozônio e 7 não possuem sobre Solar R.

Figura 16 - Identificação de dados ausentes.

```
> library(mice)
> md.pattern(airquality)
    Wind Temp Month Day Solar.R Ozone
111    1    1     1   1       1     1  0
35     1    1     1   1       1     0  1
5      1    1     1   1       0     1  1
2      1    1     1   1       0     0  2
       0    0     0   0       7    37 44
```

A função *md.pattern* produz, como resultado, o gráfico apresentado na Figura 16. Analisando essa imagem é possível observar que 111 registros estão completos, 35 possuem dados ausentes apenas no atributo Ozone, 5 registros não possuem dados sobre Solar R e 2 não possuem dados sobre Solar R e nem sobre Ozone.

Figura 17 - Análise de dados ausentes.

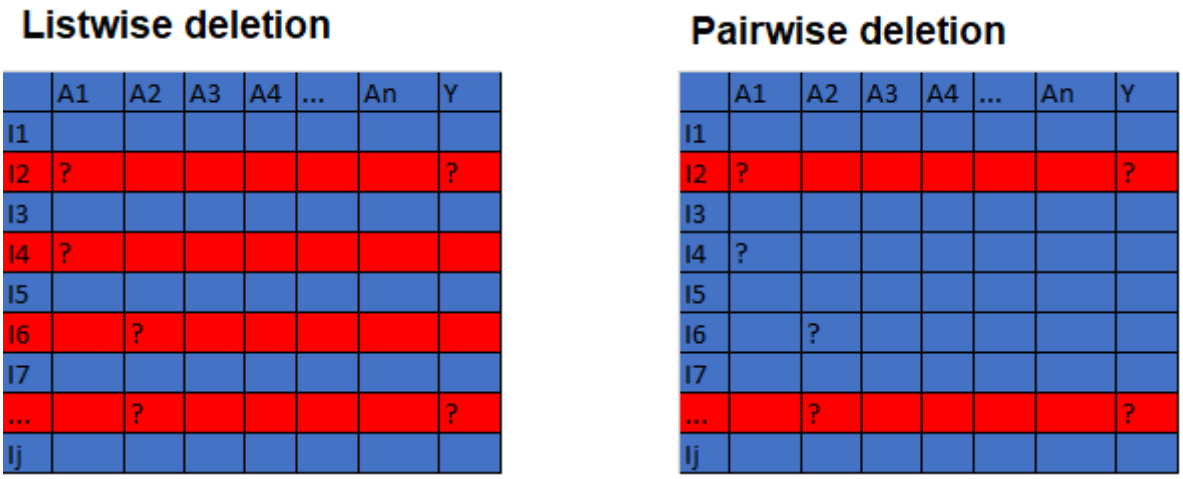


Existem várias maneiras de lidar com os dados ausentes, entre elas pode-se destacar o descarte do dado, a imputação de dados baseado em métodos simples, imputação múltipla, modelos preditivos, entre outros.

A remoção de exemplos com dados ausentes é um dos métodos mais simples e ainda hoje utilizado para lidar com dados ausentes. Duas técnicas de remoção de

dados mais conhecidas são *Listwise deletion* e *Pairwise deletion*. No *Listwise deletion* qualquer tupla que contenha dado ausente é removida, no *Pairwise deletion* apenas os exemplos que contenham dados ausentes no atributo alvo são deletados.

Figura 18 - Remoção de exemplos com dados ausentes.

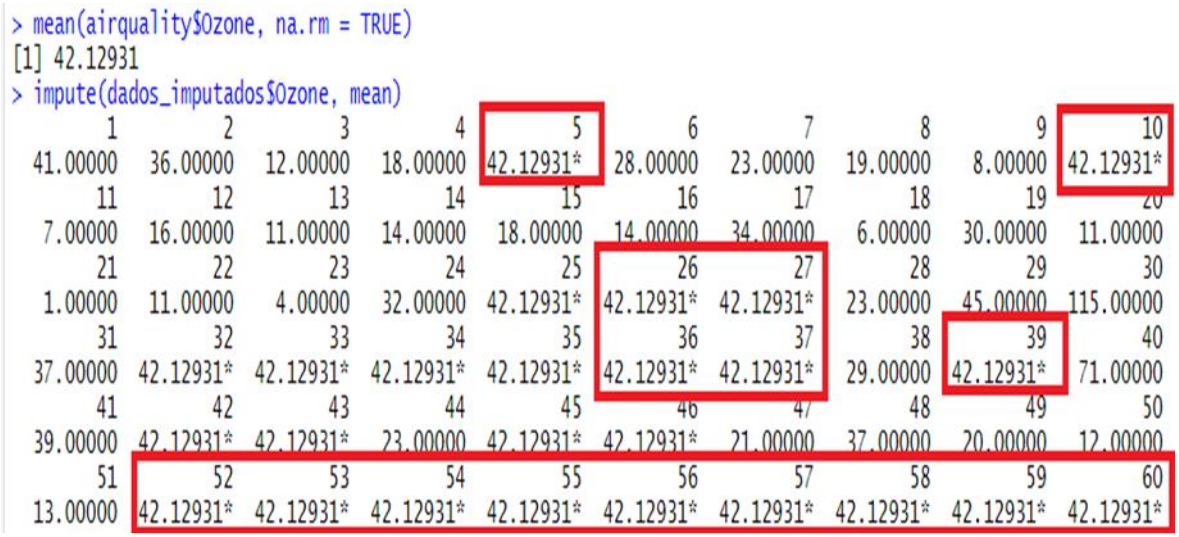


A Figura 18 exibe a aplicação dos métodos de remoção de dados em uma base de dados. Na imagem a esquerda foi aplicado o *Listwise deletion* e todos os exemplos que continham dados ausentes foram removidos (destaque em vermelho). Por outro lado, no *Pairwise deletion*, imagem à direita, apenas foram removidos exemplos quando o dado ausente estava no atributo alvo (y).

Naturalmente, a aplicação de remoção de exemplos com dados ausentes reduz a amostra de dados em análise, sendo que a redução ao adotar o *Pairwise deletion* é menor do que na *Listwise deletion.*

Outra técnica simples de lidar com dados ausentes são as imputações de dados usando métodos simples de imputação. Nesses métodos simples a imputação é feita aplicando média, moda e/ou mediana nos dados ausentes. A Figura 19 exibe a aplicação da função de imputação de dados do R, *impute*, usando a média como técnica de imputação.

Figura 19 - Imputação simples de dados.



Outro método de imputação é a imputação pela estimativa da máxima verossimilhança. Esse método é bastante comum e usa o algoritmo *Expectation-Maximization* (EM) que, em poucas palavras, estima a distribuição de probabilidades para imputação de dados.

No R a função *imputeEM* do pacote *mvdalab*, realiza a imputação de dados de acordo com o algoritmo *Expectation-Maximization.*

Outra técnica de imputação de dados ausentes é a regressão. Nessa técnica o modelo de regressão é gerado e os dados ausentes são imputados com base nesse modelo.

A Figura 20 exibe um exemplo de imputação via regressão. Inicialmente foi gerado um modelo entre y, variável com dados ausentes, e x, variável com maior correlação com y. Uma vez identificado o modelo, foram imputados valores conforme o modelo que no exemplo da Figura 20, y = 1.509*x + 9.743.

Figura 20 - Imputação via regressão.

```
> #modelo de regressão
> lm(y~x,data=dados)

Call:
lm(formula = y ~ x, data = dados)

Coefficients:
(Intercept)            x
      9.743        1.509

> for(i in 1:nrow(dados)){
+   if (dados$I[i]==0){
+     dados$y[i]=dados$x[i]*1.509+9.743
+   }
+ }
> dados
    x      y  z  w I
1   1 11.000 19  1 1
2   2 12.000 11  4 1
3   3 18.000  2  7 1
4   4 14.000 14 10 1
5   5 17.000 20  3 1
6   6 18.797  4  5 0
7   7 20.306  9  7 0
8   8 19.000 10  6 1
9   9 23.324 18  6 0
10 10 27.000  1  9 1
```

A imputação de dados também pode ser realizada através de algoritmos de aprendizado de máquina, tais como SVM, SVD e KNN. No KNN, por exemplo, o vizinho mais próximo fornece os dados a serem imputados. A definição do vizinho mais próximo é baseada em critérios, tais como distância de Gower. No R, a imputação usando KNN é realizada através da função *knn.impute* do pacote *bnstruct.*

Por fim, também é possível fazer a imputação múltipla. A imputação múltipla foi proposta por Rubin, e a sua proposta básica é para cada dado faltante fazer diversas (múltiplas) imputações. Como resultado dessa múltipla imputação, são obtidas *n (*número de imputações) e bases de dados completas. Cada base completa é analisada separadamente por um método estatístico. Após essa análise, são obtidas as estimativas de imputação com base na média das múltiplas imputações e do erro padrão.

Na linguagem R, a função *mice* do pacote *mice* realiza a imputação múltipla em dados ausentes.

3.2. Outliers

O conceito de *outlier* está relacionado a um comportamento anômalo de um determinado objeto, cabe destacar que o outlier é, em princípio, um dado verídico que merece atenção no processo de preparação de dados.

Uma definição clássica de *outlier* é apresentada por Hawkins (1980), "*an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism*". Nessa definição, é importante destacar que o *outlier* é uma observação que desvia significativamente das outras observações da base de dados.

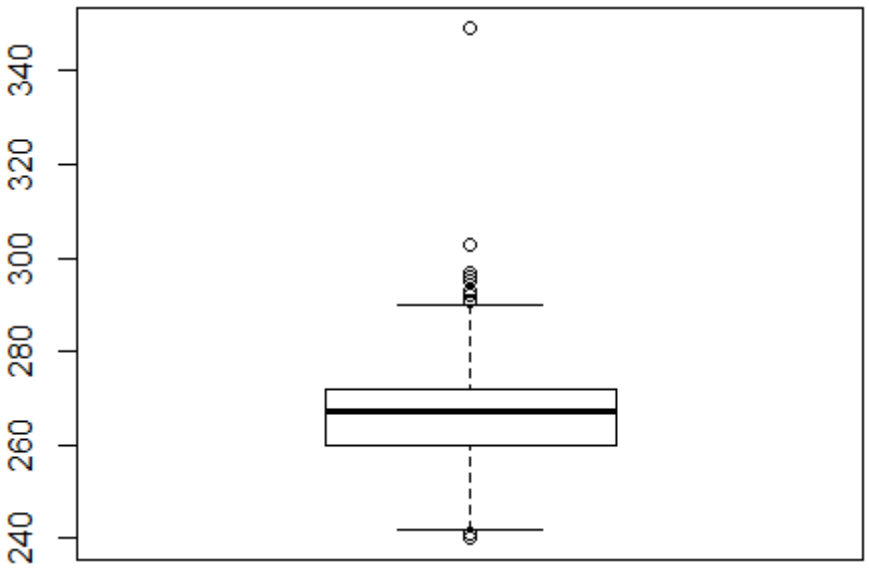
Uma técnica simples de identificação de *outlier* é geração do *box-plot,* que é uma ferramenta gráfica que permite visualizar a distribuição dos dados. Medidas como mínimo, máximo, primeiro, segundo e terceiro quartis, formam o gráfico *box-plot.* A Figura 21 exibe um exemplo de *box-plot*.

Ao analisar o gráfico da Figura 21 é possível notar que algumas observações estão acima da linha máxima do *box-plot*. Esses "pontos fora da curva" são *outliers.* Uma técnica muito utilizada para identificação dos *outliers* é a identificação dos limites de Tukey.

Os limites de Tukey são identificados a partir dos seguintes passos:

1. Identificar Q1 e Q3;
2. Calcular o IQR (InterQuartile Range);
3. Cálculo do limite inferior: Q1 – 1.*5 * IQR;
4. Cálculo do limite superior: Q3 + 1.5*IQR.

Figura 21 - Exemplo Box-plot.

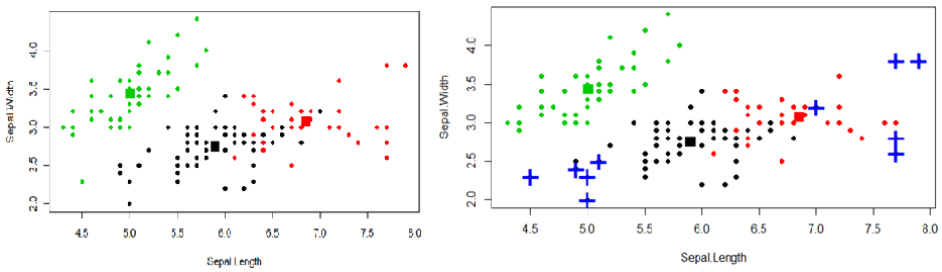


Na Figura 21 os limites inferior e superior de Tukey estão representados pelas linhas inferior e superior, respectivamente. Assim, as observações foram desse intervalo, representadas por círculos, são *outliers* da base de dados.

Outra técnica de identificação de *outlier* é a utilização de algoritmos de aprendizado máquina como o *k-means*. Nessa técnica, inicialmente a base de dados é agrupada e depois a distância entre as observações e o centro dos grupos analisadas para identificação dos *outliers*.

A Figura 22 exibe duas imagens, a imagem da esquerda mostra o agrupamento de uma base de dados em três grupos: verde, vermelho e verde. Os pontos em destaque em cada grupo representam o centro de cada grupo, sendo possível observar que alguns pontos estão distantes do centro do seu respectivo grupo.

Figura 22 - Identificação de outliers através de k-means.

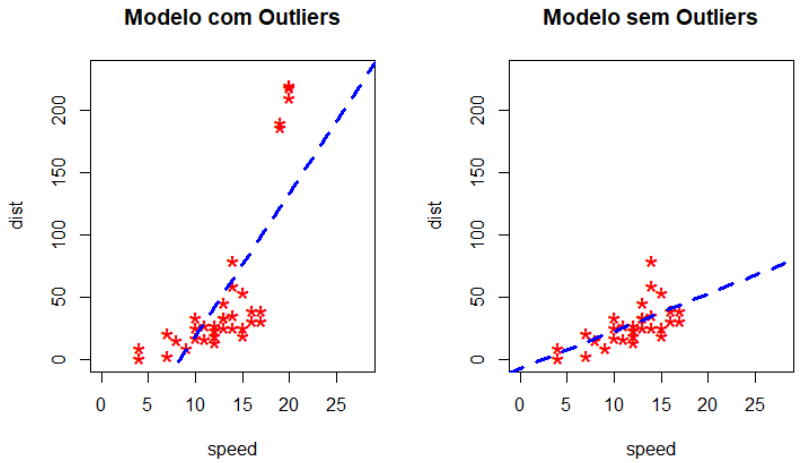


A imagem à direita da Figura 22 exibe os pontos representado pelo sinal '+' em azul, distantes do centro de seu respectivo. Esses pontos são potenciais *outliers* da base de dados.

Uma vez identificado o *outlier* é necessário mecanismos para lidar com esses dados, pois conforme representado na Figura 23, o *outlier* afeta significativamente o modelo de dados.

Para tratamento dos *outliers* uma ação simples é a remoção dos exemplos contendo *outlier*. Ou seja, nessa técnica deve-se simplesmente eliminar as instâncias. Outra técnica para o tratamento de *outlier* é a Winsorization. Essa técnica, proposta por Charles Winson, por isso o nome, consiste na substituição dos valores do *outlier* por outros valores da base de dados. Por exemplo, os *outliers* podem ser substituídos pelos limites superior ou inferior de Tukey.

Figura 23 - Influência do outlier no modelo.



Durante a análise dos *outliers* é necessário refletir se o *outlier* representa um ruído ou um sinal, pois *outliers* podem representar fraudes, comportamentos fora de padrão, eventos raros, entre outros. Ou seja, o *outlier* pode ser o objeto de estudo e, nesse caso, a remoção do *outlier* e/ou a substituição do seu valor não procede.

Capítulo 4. Transformação de Dados

A transformação de dados é mais uma etapa da preparação de dados. Embora o processo de imputar dados nos dados ausentes, tratar outliers, entre outros processos já apresentados anteriormente, estejam inseridos na fase de transformação do ETL, é importante tratar a transformação de dados sobre outro aspecto.

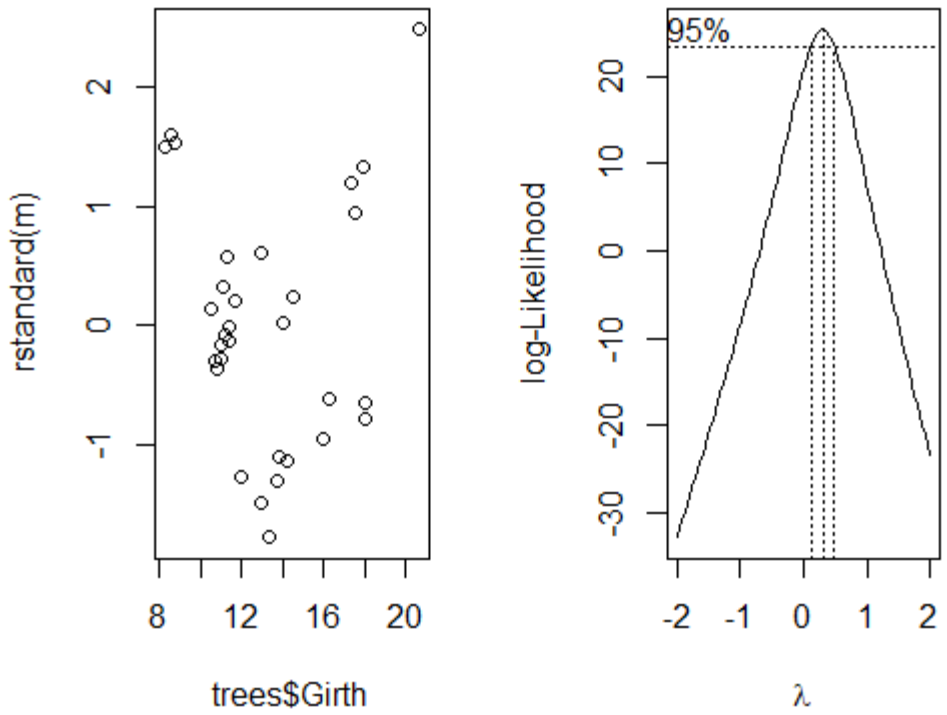
A transformação de dados tipicamente transforma valores de atributos originais em novos valores a partir de funções matemáticas, tais como transformações lineares, quadráticas, Box-Cox, YeoJhonson, entre outras técnicas de transformação.

As motivações para transformação podem ser de diversas naturezas, desde por conveniência, como por exemplo transformar um valor bruto em porcentagem, até por uma questão de assimetria dos dados. Pode-se também realizar transformações para a fase de integração de dados, enfim, existem diversas motivações para a transformação dos dados, mas o objetivo sempre é qualificar os dados para que esses possibilitem a descoberta de melhores modelos.

Como existem diversas técnicas de transformação, qual técnica deve ser usada? Sempre depende do dado que está sendo transformado, por exemplo, se o dado em questão possui uma assimetria para esquerda, pode-se aplicar transformações quadráticas ou cúbicas para ajustar a curva. Ou seja, a técnica de transformação sempre depende da natureza do dado a ser transformado. Além disso, cabe destacar que um grande desafio da transformação é que nunca é possível saber, antecipadamente, qual transformação produzirá o melhor modelo.

A linguagem R possui a função *preProcess*, que possibilita uma série de transformações nos dados. Observe a Figura 24 que mostra dois gráficos, um antes e outro depois da transformação. Nesse caso foi usada a transformação Box-Cox, observe a distribuição do gráfico após a transformação, segundo gráfico da imagem.

Figura 24 - Transformação BoxCox.



Nesse capítulo abordaremos a transformação via normalização de dados e a discretização de atributos contínuos.

4.1. Normalização

Conforme apresentado na introdução desse capítulo, o dado em estado bruto pode não ser útil para análise. Assim, faz-se necessário a transformação dos dados, sendo que uma das formas de transformar esses dados é através a normalização. Os novos atributos, após a transformação, são normalmente conhecidos como atributos de modelagem.

A normalização é um procedimento que visa transformar a escala dos dados, de tal forma que os dados fiquem dentro de um intervalo específico. Esse procedimento é muito útil quando os algoritmos de análise de dados são baseados em distâncias, tais como a distância euclidiana.

Duas técnicas de normalização frequentemente utilizadas são: normalização max-min e normalização z-score.

A normalização max-min é uma transformação linear, computada através da seguinte fórmula:

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Na fórmula assim, min(x) e max(x) são os valores mínimos e máximos de um atributo respectivamente. Assim, o intervalo de valor do novo atributo será entre 0 e 1, sendo que 0 ocorrerá para o menor valor original e 1 para o maior valor original.

Considere o seguinte exemplo: em uma empresa, o maior salário de uma base é R$50.000,00 e o R$1.000,00. Para normalizar, o salário de R$10.000,00, deve aplicar na equação acima, assim:

$$z = \frac{10000-1000}{50000-1000} \rightarrow \frac{9000}{49000} \rightarrow 0{,}18$$

Refaça o exemplo acima normalizando os valores no valor de R$50.000,00 e R$1.000,00. Quais foram os resultados?

A Figura 25 mostra a normalização do atributo *speed* de uma base de dados. Conforme exibido na figura abaixo, os dados originais variam de 4, valor mínimo, até 25, valor máximo. Os dados originais foram submetidos a função de transformação e, após a normalização, passaram a variar de 0 a 1.

Figura 25 - Normalização Min-Max.

```
> dados$speed
 [1]  4  4  7  7  8  9 10 10 10 11 11 12 12 12 12 13 13 13 13
[20] 14 14 14 14 15 15 15 16 16 17 17 17 18 18 18 18 19 19 19
[39] 20 20 20 20 20 22 23 24 24 24 24 25
> normalize <- function(x) {
+   return ((x - min(x)) / (max(x) - min(x)))
+ }
> dados$speed_norm<-normalize(dados$speed)
> dados$speed_norm
 [1] 0.0000000 0.0000000 0.1428571 0.1428571 0.1904762
 [6] 0.2380952 0.2857143 0.2857143 0.2857143 0.3333333
[11] 0.3333333 0.3809524 0.3809524 0.3809524 0.3809524
[16] 0.4285714 0.4285714 0.4285714 0.4285714 0.4761905
[21] 0.4761905 0.4761905 0.4761905 0.5238095 0.5238095
[26] 0.5238095 0.5714286 0.5714286 0.6190476 0.6190476
[31] 0.6190476 0.6666667 0.6666667 0.6666667 0.6666667
[36] 0.7142857 0.7142857 0.7142857 0.7619048 0.7619048
[41] 0.7619048 0.7619048 0.7619048 0.8571429 0.9047619
[46] 0.9523810 0.9523810 0.9523810 0.9523810 1.0000000
```

Outra técnica de normalização de dados é o *z-score*. Essa técnica tem como referência a média e o desvio padrão de um atributo. Essa transformação é útil quando existem ocorrências de *outliers* no atributo a ser normalizado. Essa normalização segue a seguinte fórmula:

$$z = \frac{x - \mu}{\sigma}$$

Onde μ é a média e σ o desvio padrão do atributo a ser normalizado.

Na linguagem R, a função *scale* faz a normalização da base de dados conforme a técnica *z-score*. A Figura 26 exibe a transformação dos dados do atributo *speed* de acordo com a técnica *z-score,* e o conteúdo do atributo após a transformação.

Figura 26 - Transformação *z*-score.

```
> dados$norm2 <- scale(dados$speed)
> dados$norm2
             [,1]
 [1,] -2.15596948
 [2,] -2.15596948
 [3,] -1.58860909
 [4,] -1.58860909
 [5,] -1.39948896
 [6,] -1.21036883
 [7,] -1.02124870
 [8,] -1.02124870
 [9,] -1.02124870
```

4.2. Discretização

Conforme citado anteriormente, os dados brutos vêm em diferentes formatos, ou seja, os dados são heterogêneos. Os dados podem ser discretos, numéricos, contínuos, categóricos etc. Dados numéricos são naturalmente ordenados, por outro lado, dados categóricos não necessariamente são ordenados.

Diversas técnicas de análise de dados dependem da natureza dos dados para performar corretamente. Assim, durante o pré-processamento dos dados, muitas

vezes é necessária transformar os dados de quantitativo para qualitativo, ou seja, transformar dados numéricos em dados discretos ou nominais. Esse processo de transformação é conhecido como: **discretização.**

A discretização é definida como o processo de transformar atributos contínuos em discretos (ou nominais), com intervalos de valores finitos, sem sobreposição das partições do domínio do atributo contínuo. Em outras palavras, a discretização faz o mapeamento de um grande conjunto de valores de atributo contínuo em um conjunto reduzido de valores discretos.

Considere, por exemplo, o atributo contínuo *nota,* que pode variar de 0 a 100. Esse atributo pode ser discretizado em conceitos A, B, C, D e E. A Tabela 1, a seguir, exibe o mapa de conversão do atributo numérico para categórico.

Tabela 1 - Tabela de conversão nota em conceito.

Atributo Original	Atributo Discretizado
90 a 100	A
70 a 89,9	B
50 e 69,9	C
40 a 49,9	D
Abaixo de 50	E

Entre as vantagens do processo de discretização, pode-se destacar a redução e a simplificação dos dados que torna a análise de dados mais rápida e com maior acurácia.

Na literatura existem diversos métodos de discretização de dados. Esses métodos podem ser classificados em estático ou dinâmico; univariado ou multivariado; supervisionado ou não supervisionado; global ou local; entre outras categorias de classificação. Nesse material, o foco é dado nos métodos supervisionados e não supervisionados.

Nos métodos supervisionados, os dados a serem transformados possuem intervalos definidos previamente, assim com o rótulo de cada intervalo. Como no exemplo anterior das notas o mapeamento já está predefinido, notas de 90 a 100 serão categorizadas no conceito A e assim por diante.

A Figura 27 mostra um exemplo de discretização supervisionada implementada na linguagem R, que utiliza a função *cut* para discretização, sendo que nesse exemplo, inicialmente, foi gerado um vetor com idade de pessoas. Após a geração do vetor, foram definidos os rótulos (*labels)* e os intervalos para cada rótulo. O resultado da discretização foi armazenado no vetor AgeD.

Figura 27 - Discretização supervisionada.

```
> Age[1:10]
 [1]  6 18 16 14  5 11  8 11 15 11
> AgeD = cut(Age, breaks=c(0,12,18,60,100),
+            labels= c("Criança","Adolescente","Adulto","Idos
o"))
> AgeD[1:10]
 [1] Criança     Adolescente Adolescente Adolescente
 [5] Criança     Criança     Criança     Criança
 [9] Adolescente Criança
Levels: Criança Adolescente Adulto Idoso
```

Nos métodos não supervisionados os intervalos e os rótulos não são predefinidos. As técnicas mais tradicionais e discretização não supervisionada são *Equal Width* (intervalos iguais) e *Equal Frequency* (frequências iguais). Na técnica *Equal Width* os dados originais são divididos em *k* intervalos de tamanhos iguais. Enquanto na técnica *Equal Frequency*, os intervalos são definidos de acordo com o número de ocorrências de registros em cada intervalo, nessa técnica o objetivo é gerar intervalos com frequências, aproximadamente, iguais.

Na linguagem R, além da função *cut* usada anteriormente, existe a função *discretize* do pacote *arules*. A Figura 28 exibe a discretização não supervisionada, usando função *discretize* do R. No primeiro comando é aplicada a discretização *Equal Frequency,* em que os intervalos são definidos em função do número, aproximadamente, igual de ocorrências. No segundo comando, os intervalos são iguais e as ocorrências são classificadas no intervalo.

Figura 28 - Discretização não supervisionada.

```
> table(discretize(x, method = "frequency", breaks = 3))

 [0.8,5.43) [5.43,9.47) [9.47,17.4]
         17          16          17
>
>
> table(discretize(x, method = "interval", breaks = 3))

 [0.8,6.33) [6.33,11.9) [11.9,17.4]
         22          17          11
```

Além dos desses dois métodos apresentados, também é possível utilizar métodos de Aprendizado de Máquina para definição do intervalo. Um dos métodos de aprendizado de máquina mais utilizado é a *clusterização* usando *k-means.* O objetivo da discretização usando *k-means* é particionar os dados em *k* grupos (*clusters*). A Figura a seguir mostra o método *discretize*, aplicando o método de discretização baseado na clusterização.

Figura 29 - Discretização usando clusterização.

```
> table(discretize(USArrests$Murder, method = "cluster", breaks = 3))

 [0.8,5.44) [5.44,10.6) [10.6,17.4]
         17          19          14
>
```

Capítulo 5. Redução de dimensionalidade

Atualmente não é difícil de imaginar o grande volume de dados gerado diariamente. Conforme apresentado na introdução desse material, quintilhões de bytes são gerados diariamente e existem bases de dados com mais de 29 milhões de atributos.

Um dos grandes problemas da Análise de Dados e do Data Mining é a maldição da dimensionalidade, *Curse of Dimensionality. Curse of Dimensionality* é um termo proposto por Richard Bellmann em 1961 para representar algoritmos que se tornam intratáveis quando a dimensão da entrada de dados aumenta.

Com o objetivo de reduzir os problemas causados pela alta dimensionalidade dos dados, diversas técnicas de redução de dimensionalidade foram desenvolvidas ao longo dos anos. Essas técnicas de redução de dimensionalidade podem ser categorizadas em três grandes grupos: redução do número de atributos, redução do número de exemplos (tuplas) e redução da cardinalidade. Nesse material serão abordadas duas subcategorias para redução de dimensionalidade quanto ao número de atributos: extração de atributos (*Feature Extraction*) ou seleção de atributos (*Feature Selection*).

Abordagens baseadas em *Feature Extraction* visam criar um novo espaço de atributos com dimensionalidade reduzida. Entre as técnicas que adotam essa abordagem, pode-se destacar: *Principle Component Analysis* (PCA), Linear *Discriminant Analysis* (LDA), *Canonical Correlation Analysis* (CCA), entre outras.

Por outro lado, a abordagem *Feature Selection* (FS) visa selecionar um subconjunto de atributos que minimize a redundância dos dados e maximize a relevância. O principal objetivo do FS é encontrar o melhor subconjunto de atributos que possibilita a construção de modelos de um fenômeno em estudo.

A redução da dimensionalidade das bases de dados proporciona os seguintes benefícios: melhoria na performance dos algoritmos de aprendizado de máquina, redução da complexidade computacional, construção de melhores modelos e redução no espaço de armazenamento.

5.1. Extração de atributos

Na extração de atributos, *Feature Extraction,* novos atributos derivados da base original são identificados. Nesse sentido técnicas como PCA e Análise de Fatores têm sido bastante utilizadas.

A Análise de Componentes Principais (PCA) é um dos métodos multivariados mais antigos e mais utilizados para redução de dimensionalidade. O PCA foi proposto por Pearson em 1901 e revisada por Hotteling em 1933, por isso também é conhecida como Transformação de Hotteling. PCA é uma técnica da álgebra linear que objetiva, através de transformações ortogonais no espaço original de atributos, identificar um subespaço de dimensões reduzidas sem perda de informação.

O objetivo do PCA é encontrar um novo subconjunto de variáveis, menor que o original, que melhor represente a variabilidade dos dados. Em outras palavras, o PCA elimina atributos redundantes ao criar novos atributos, que são independentes uns dos outros.

Os passos do algoritmo do PCA são:

1) Calcular a matriz de covariância da base de dados;

2) Calcular os autovalores e autovetores da matriz de covariância;

3) Escolher as componentes principais;

4) Construir a nova base dados.

A seguir a implementação do PCA na linguagem R. Esse exemplo é baseado no livro “Análise de dados através de métodos de estatística multivariada: uma abordagem aplicada” de Sueli Mingoti. A base de dados, exibida na Figura 30, é o exemplo da Mingoti - Tabela 3.2, pag. 70.

Figura 30 - Base de dados: salgados.

```
> head(salgados)
  Sabor Aroma Massa Recheio
1  2.75  4.03  2.80    2.62
2  3.90  4.12  3.40    3.52
3  3.12  3.97  3.62    3.05
4  4.58  4.86  4.34    4.82
5  3.97  4.34  4.28    4.98
6  3.01  3.98  2.90    2.82
```

Fonte: Mingoti (2005).

Após carregar a base de dados, conforme descrito anteriormente, o primeiro passo é calcular a matriz de covariância. A linguagem R oferece a função “cov(base de dados)”, a Figura 31 exibe a matriz de covariância gerada pela função cov da linguagem R.

Figura 31 - Matriz de covariância.

```
> mc=cov(salgados)
> mc
            Sabor     Aroma     Massa   Recheio
Sabor   0.4069071 0.1822107 0.3573571 0.5504750
Aroma   0.1822107 0.1104411 0.1796357 0.2713804
Massa   0.3573571 0.1796357 0.4242000 0.5897357
Recheio 0.5504750 0.2713804 0.5897357 0.9105696
```

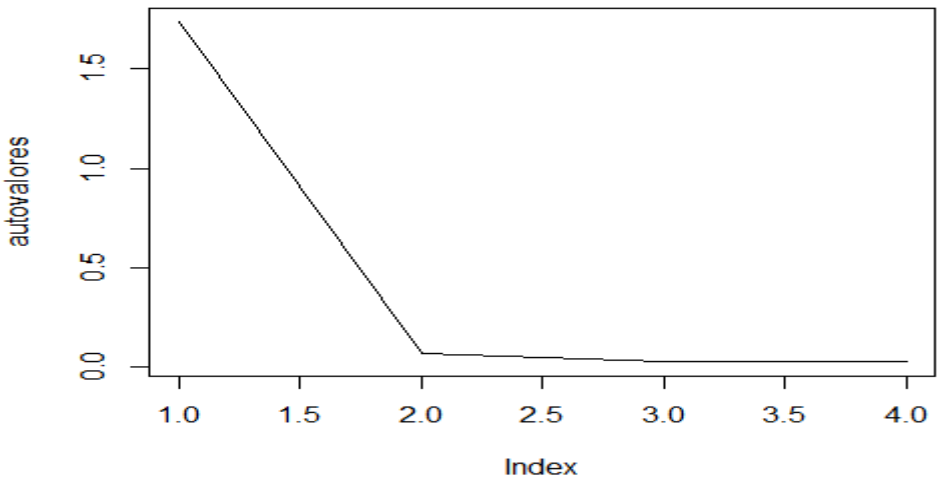
Uma vez gerada a matriz de covariância é necessário calcular seus autovalores (*eigenvalues) e* autovetores (*eigenvectors).* Para isso, a linguagem R oferece a função *eigen*. A Figura 32 exibe os autovetores e autovalores da matriz de covariância da Figura 31.

Figura 32 - Autovalores e autovetores da matriz de covariância.

```
> autovalores  = eigen(mc)$values
> autovetores  = eigen(mc)$vectors
> autovalores
[1] 1.73684287 0.06488506 0.02791357 0.02247636
> autovetores
           [,1]       [,2]       [,3]        [,4]
[1,] -0.4557692  0.8160940 -0.1121458 -0.33717690
[2,] -0.2234675  0.2149422 -0.2691445  0.91182420
[3,] -0.4770120 -0.4564207 -0.7177864 -0.22118397
[4,] -0.7174930 -0.2819051  0.6322715  0.07724004
```

A Figura 33 exibe a relação de cada componente. Observe que a partir do segundo componente não há ganho significativo. Assim, pode-se concluir que dois componentes representam a variância base de dados original.

Figura 33 - Gráfico de autovalores.



A linguagem R também fornece duas funções para identificação dos componentes principais: *prcomp* e *princomp*. A função prcomp() gera a PCA baseado no SVD, por outro lado princomp() gera a PCA baseado na matriz de covariância. A Figura 34 exibe o PCA calculado a partir da função princomp(), observe que o acumulado dos dois componentes principais representa 97.27% da base de dados.

Figura 34 - PCA identificado a partir da função princomp.

```
> pcacov = princomp(salgados)
> summary(pcacov)
Importance of components:
                          Comp.1    Comp.2     Comp.3     Comp.4
Standard deviation     1.2327763 0.2382739 0.15628300 0.14023842
Proportion of Variance 0.9377604 0.0350329 0.01507116 0.01213549
Cumulative Proportion  0.9377604 0.9727933 0.98786451 1.00000000
```

Assim como o PCA, a Análise Fatorial é uma técnica multivariada de análise de dados que pode ser utilizada para a redução de dimensionalidade de bases de dados. Análise Fatorial é uma técnica de estatística exploratória que, dada uma base de dados, busca identificar conjuntos de variáveis altamente correlacionadas, esses conjuntos são chamados de Fatores. Assim, pode-se dizer que os Fatores reduzem a base de dados sem perda informacional.

A linguagem R também fornece recursos para identificação dos Fatores da Análise Fatorial. Considere por exemplo a base de dados corrida[3], que contém os recordes femininos de corridas em 54 países. Assim, a base é composta por 54 instâncias e 8 atributos, sendo que o primeiro atributo identifica o país e os outros 7 atributos, os recordes nas distâncias de 100 m (s), 200 m (s), 400 m (s), 800 m (min), 1500 m (min), 3000 m (min) e maratona (min). Os dados são de 2005.

Nesse exemplo, a análise fatorial foi aplicada usando o método de componentes principais (PCA). Assim, a aplicação de PCA na base retorna os resultados exibidos na Figura 35.

[3] Base disponível em http://www.stat.wisc.edu/~rich/JWMULT06dat/T1-9.dat

Figura 35 - Resultados da PCA.

```
> pca = prcomp(dados, scale = TRUE)
> summary(pca)
Importance of components:
                          PC1     PC2    PC3     PC4    PC5     PC6     PC7
Standard deviation     2.4099 0.79290 0.5285 0.35292 0.3016 0.23349 0.11959
Proportion of Variance 0.8297 0.08981 0.0399 0.01779 0.0130 0.00779 0.00204
Cumulative Proportion  0.8297 0.91947 0.9594 0.97717 0.9902 0.99796 1.00000
>
```

A partir da análise dos resultados do PCA, é possível observar que os dois primeiros componentes (PC1 e PC2), a proporção é de praticamente 92% da variância total. Com base nessa análise, a Análise Fatorial será baseada em dois fatores.

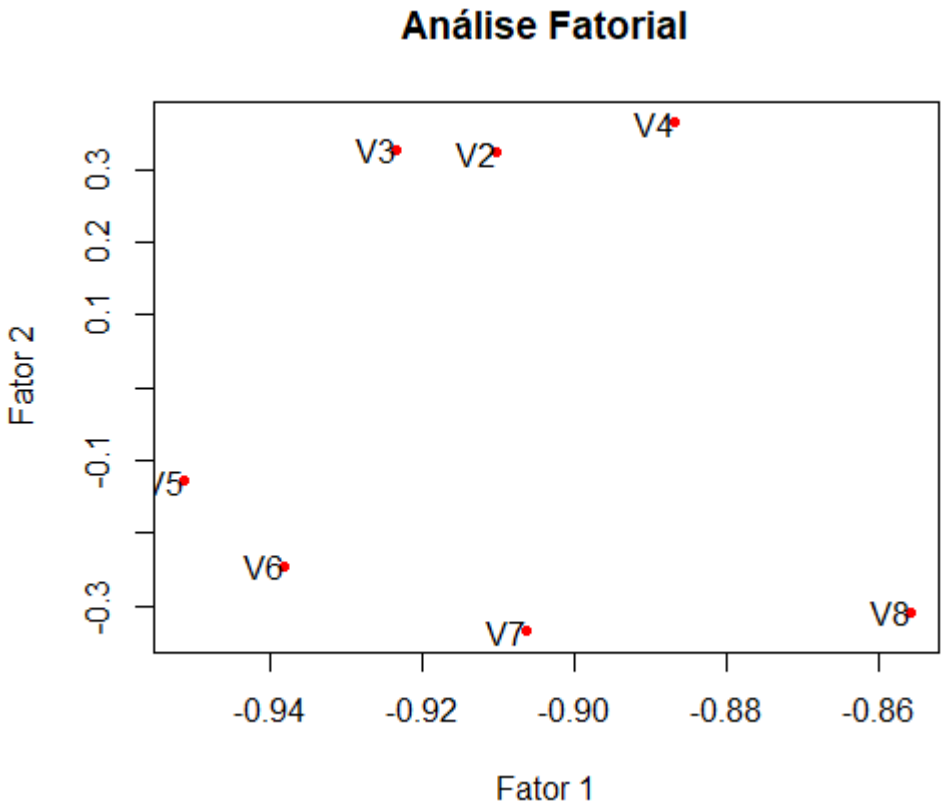
A carga fatorial é calculada a partir dos autovetores e do desvio padrão retornado na função *prcomp*, que faz a PCA. Assim, a carga fatorial dos dois primeiros componentes é calculada (MINGOTI, 2005), conforme exibido na Figura 36, sendo que o "$rotation" contém o os autovetores e o "$sdev" o desvio padrão.

Figura 36 - Carga dos fatores.

```
> carf = pca$rotation[,1:2] %*% diag(pca$sdev[1:2])
> carf
          [,1]       [,2]
V2 -0.9103780  0.3228503
V3 -0.9234990  0.3279673
V4 -0.8869307  0.3642220
V5 -0.9513832 -0.1278522
V6 -0.9380805 -0.2450762
V7 -0.9063506 -0.3355481
V8 -0.8560043 -0.3086096
```

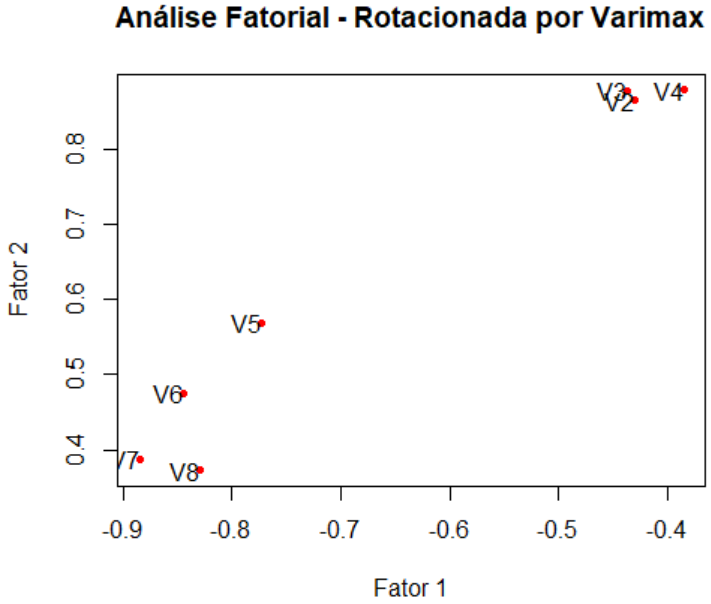
O gráfico da Figura 37 exibe a distribuição das variáveis nos dois fatores.

Figura 37 - Gráfico dos fatores.



Aplicando uma rotação “varimax” na carga dos fatores, através do comando “*carfrot <- varimax(carf)*”, obtém-se um novo gráfico conforme exibido na Figura 38.

Figura 38 - Gráfico rotacionado.



A rotação dos fatores facilita a visualização dos agrupamentos das variáveis em fatores. Observando o gráfico da Figura é possível visualizar claramente o agrupamento das variáveis V2, V3 e V4, com carga maior no Fator 2. Pode-se assim concluir que o Fator 2 está agrupando resultados obtidos em provas de distâncias curtas.

A linguagem R também oferece a função "*factanal*", que faz a identificação dos fatores baseada na matriz de covariância.

5.2. Seleção de atributos

Os algoritmos de seleção de atributos podem ser classificados em três categorias, dependendo do método de busca: filtro, *wrapper* e embutidos.

A proposta da abordagem filtro é fazer um ranqueamento dos atributos, a partir da base de dados, identificando os mais relevantes em relação a uma variável alvo de acordo com algum critério pré-estabelecido, como por exemplo, a correlação de Pearson.

Diferente da abordagem de filtro, a abordagem *wrapper* usa o algoritmo de aprendizado de máquina para avaliar o subconjunto de atributos selecionados. Em outras palavras, um subconjunto de atributos é gerado e submetido ao algoritmo e sua precisão é utilizada para avaliar a relevância do subconjunto de atributos. Esse processo é repetido para todos os subconjuntos de atributos, até que um critério de parada seja atingido. Essa abordagem é computacionalmente custosa.

Na abordagem embutida, as vantagens das abordagens anteriores são combinadas e são implementadas em algoritmos que possuem seus próprios critérios de seleção de atributos, tais como Lasso e Ridge.

A linguem R possui diversos pacotes de seleção de atributos, tais como Boruta, Caret, entre outros. O exemplo a seguir exibe as variáveis mais importantes

para explicação da diabetes. A base usada nesse exemplo é a PimaIndiansDiabetes que possui 9 atributos e 768 instâncias.

Figura 39 - Seleção de atributos.

```
> model <- train(diabetes~., data=PimaIndians[
Control=control)
> importance <- varImp(model, scale=FALSE)
> importance
ROC curve variable importance

         Importance
glucose      0.7881
mass         0.6876
age          0.6869
pregnant     0.6195
pedigree     0.6062
pressure     0.5865
triceps      0.5536
insulin      0.5379
```

A partir da Figura 39 é possível verificar que os três atributos mais importantes são “glucose”, “mass” e “age”, para a explicação da ocorrência, ou não, em diabetes.

A Figura 40 exibe um código de seleção de atributos e avaliação de desempenho do algoritmo de aprendizado *Random Forest,* com a base original e com a base reduzida.

Figura 40 - Algoritmo de seleção de atributos.

```
> library('Metrics')
> library('randomForest')
> data <- read.csv("C:/Users/walis/Dropbox/IGTI/stock_data.csv")
> data$Y<-as.factor(data$Y)
> train<-data[1:2000,]
> test<-data[2001:3000,]
> model_rf<-randomForest(Y ~ ., data = train)
> preds<-predict(model_rf,test[,-101])
> auc(preds,test$Y)
[1] 0.4941012
> #sort(importance(model_rf))
>
> model_rf<-randomForest(Y ~ X55+X11+X15+X64+X30
+                        +X37+X58+X2+X7+X89
+                        +X31+X66+X40+X12+X90
+                        +X29+X98+X24+X75+X56,
+                        data = train)
> preds<-predict(model_rf,test[,-101])
> table(preds)
preds
 -1   1
 58 942
> auc(preds,test$Y)
[1] 0.4721429
>
```

Fonte: https://www.analyticsvidhya.com.

No programa da Figura 40, a base utilizada possui uma variável alvo que informa se o estoque de um determinado produto vai subir ou abaixar. Além da variável alvo, existem 100 variáveis preditivas, ou seja, variáveis que explicam a variável alvo. Inicialmente o modelo, model_rf, foi gerado usando toda a base de dados. O resultado da predição desse modelo foi de 49,4%.

A partir desse modelo foram identificadas as 20 variáveis mais importantes, *importance(model_rf),* e um novo modelo foi gerado, considerando apenas essas 20 variáveis. Como resultado, a acurácia da predição foi 47,2%. Assim, ocorreu um decréscimo na acurácia no *Random Forest*, porém pequena se for considerada a redução de atributos. Importante destacar que em ambos os casos a acurácia é pequena.

Capítulo 6. Integração de Dados

A integração de dados visa criar uma visão única de dados provenientes de diversas fontes de dados. Em outras palavras, esse processo da Preparação de Dados tem como produto final uma base de dados contendo dados de diversas fontes, que são integradas em um único repositório. Importante destacar que caso essa etapa não seja realizada adequadamente, é possível que na base de dados final exista redundâncias e inconsistências que irão afetar a qualidade do processo de análise de dados como um todo.

Durante o processo de Integração de Dados vários desafios são encontrados pelos analistas de dados, entre eles pode-se destacar: i) integração do esquema; ii) redundância de atributos; e iii) redundância de tuplas.

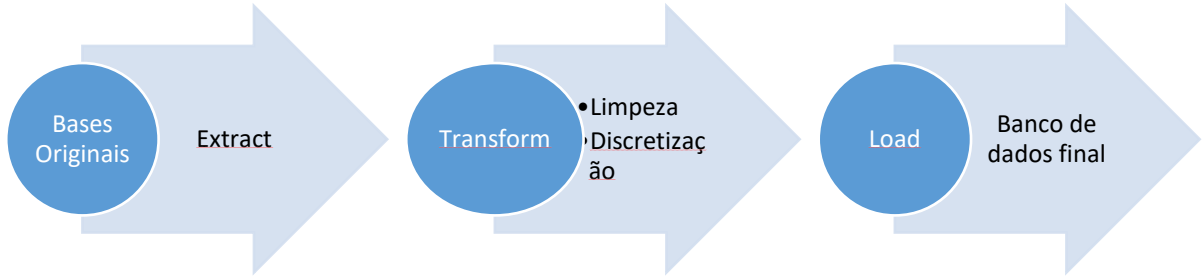
Na análise do esquema deve-se atentar para atributos que possuem nome diferentes em bases distintas, mas que assumem o mesmo valor. Por exemplo, em uma base de dados o atributo pode chamar “id_client” e em outra base “client_number”. Outro cenário possível é a existência de atributos com estruturas diferentes, por exemplo: composto e simples. Além de tipos diferentes, em uma base um atributo pode ser inteiro e esse mesmo atributo pode, em outra base, ser real. Esses, entre outros aspectos, devem ser considerados na integração de esquemas.

Além dos problemas relacionados à integração de esquema, devem ser analisadas as redundâncias inerentes ao processo de integração. Redundâncias essas que podem ser atributos e/ou de tuplas.

Algumas propostas de automatização da integração de esquemas e de redundância de tuplas podem ser encontradas em Doan et al. (2001) e em Elmagarmid; Ipeirotis e Verykios (2007).

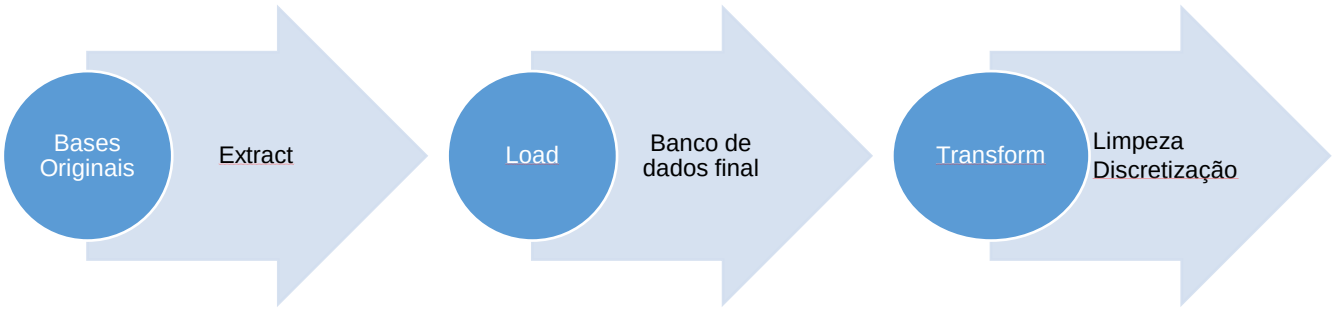
Para a realização da integração, dois processos podem seguidos: ETL ou ELT. No tradicional processo ETL (***E**xtract **T**ransform and **L**oad)*, representado pela Figura 41, os dados são extraídos das bases de dados e na sequência transformados, ou seja, preparados, para finalmente serem carregados (*Load)* em um repositório único.

Figura 41 - Processo ETL.



Por outro lado, no processo ELT (***E**xtract **L**oad and **T**ransform),* exibido na Figura 42, os dados são extraídos das fontes originais e carregados, ou seja, na etapa de ***L**oad* ocorre a carga dos dados brutos, sem nenhum tipo de preparação. Após a carga, já no repositório final, os dados são transformados de acordo com o problema a ser analisado.

Figura 42 - Processo ELT.



No primeiro processo o repositório final normalmente é o *Data Warehouse,* enquanto no ELT os dados brutos são carregados em um *Data Lake*. *Data Warehouse* e *Data Lake* são assuntos das próximas seções.

6.1. Data Warehouse

Data Warehouse (DW) é um termo cunhado por Bill Inmon no início da década de 1990. Bill Inmon, que é considerado o "pai do DW", define DW como "*A data*

warehouse is a ***subject-oriented****,* ***integrated****,* ***nonvolatile****, and* ***time-variant*** *collection of data in support of management's decisions*" (INMON, 1992 p. 31). (Grifo do autor)

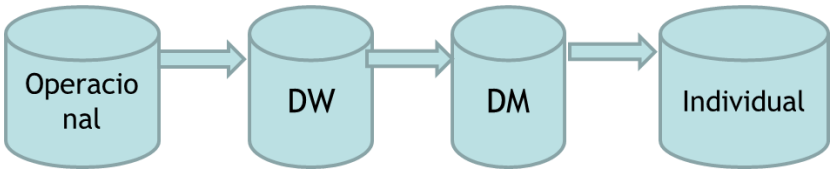
A definição de Inmon remete às quatro características do *Data Warehouse*: i) orientado a assunto; ii) integrado; iii) não volátil; e iv) variável com o tempo. Por ser orientado a assunto, entende-se que o DW é específico sobre um determinado assunto e não sobre todas as operações da empresa. Por exemplo, pode ser um DW referente às vendas de uma empresa.

A característica "integrado" está relacionada ao fato da coleta dos dados ser realizada em diversas fontes de dados. Não volátil significa que os dados, após a carga no processo ETL, estão disponíveis apenas para consulta, ou seja, não sofrem alterações. Por fim, variável no tempo significa que o dado se refere a um determinado período. Por exemplo, as vendas nos últimos 5 anos.

Outro autor especialista em DW é Ralph Kimball, que apresenta uma definição um pouco mais simplificada de DW. Para Kimball (1996), Data Warehouse pode ser definido como "*a copy of transaction data specifically structured for query and analysis*". Em sua definição, Ralph Kimball destaca que o DW é uma cópia das bases de dados transacionais, estruturada para consulta e análise.

Os especialistas em DW, Inmon e Kimball, além do conceito de DW, também apresentam as arquiteturas de DW, sendo que a arquitetura de Inmom é mais simples. A arquitetura de Inmon, apresentada na Figura 43, exibe as quatro camadas da produção da informação dentro da organização: operacional, Data Warehouse, *Data Mart* e individual.

Figura 43 - Arquitetura DW Inmon.



Fonte: Adaptada de Inmon (1992).

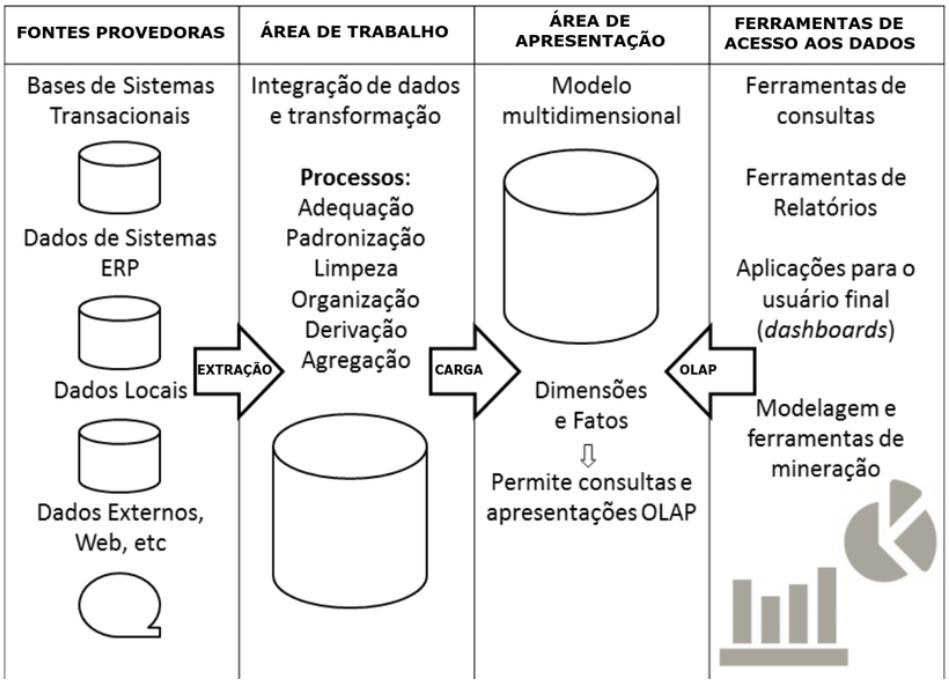
O nível operacional contém dados dos sistemas transacionais, que são os produtores de dados internos da organização. A segunda camada, DW, possui os dados transformados pelo processo ETL. Nessa camada, conforme o conceito de DW de Inmon, os dados são integrados e históricos. O terceiro nível da arquitetura de Inmon contém dados específicos modelados pelo usuário final, conforme necessidades do departamento. No nível individual, as análises heurísticas são realizadas.

A arquitetura de Kimball, Figura 44, destaca uma área de trabalho, *staging area*. Nessa área de trabalho, após a coleta dos dados de diversas fontes, é realizada a transformação dos dados para, na sequência, ocorrer a carga em DW que foi modelado usando os conceitos da modelagem dimensional. Uma vez que os dados estão preparados no DW, eles estão disponíveis para as consultas que são realizadas com apoio de diversas ferramentas.

Na modelagem dimensional proposta por Kimball, existe apenas dois tipos de tabelas: fato e dimensão. A tabela fato contém o fato principal a ser analisado e suas medidas. A granularidade do DW é definida a partir de cada linha, registro, da tabela fato. As dimensões da modelagem dimensional representam os descritores da tabela fato. Assim, cada dimensão representa uma característica do assunto, tabela fato, que está sendo modelado.

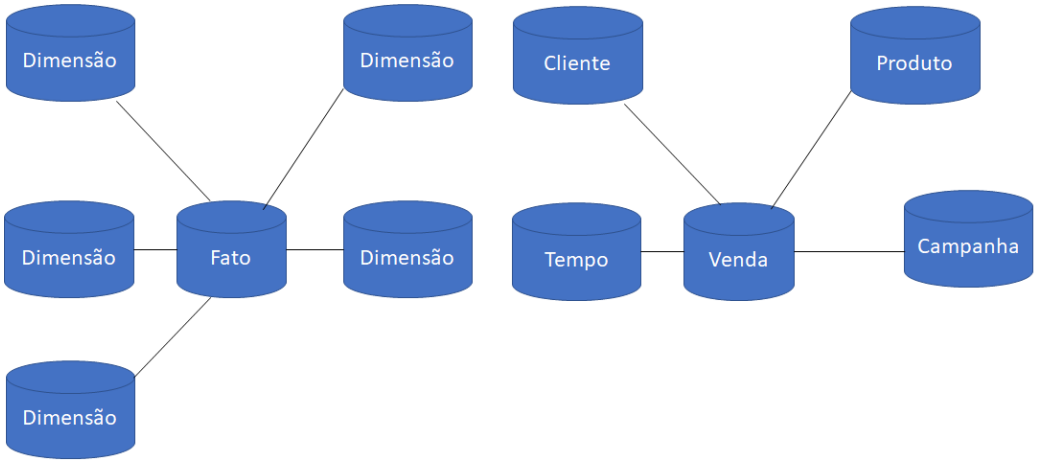
A técnica mais utilizada na modelagem dimensional é a Modelagem Estrela, *Star Schema*. Nessa modelagem a tabela fato é colocada no centro e as dimensões circundam a tabela fato conforme a Figura 44. A Figura 45 exibe um exemplo da modelagem dimensional que a tabela fato, vendas, possui: as dimensões cliente, produto, tempo e campanha.

Figura 44 - Arquitetura de Kimball.



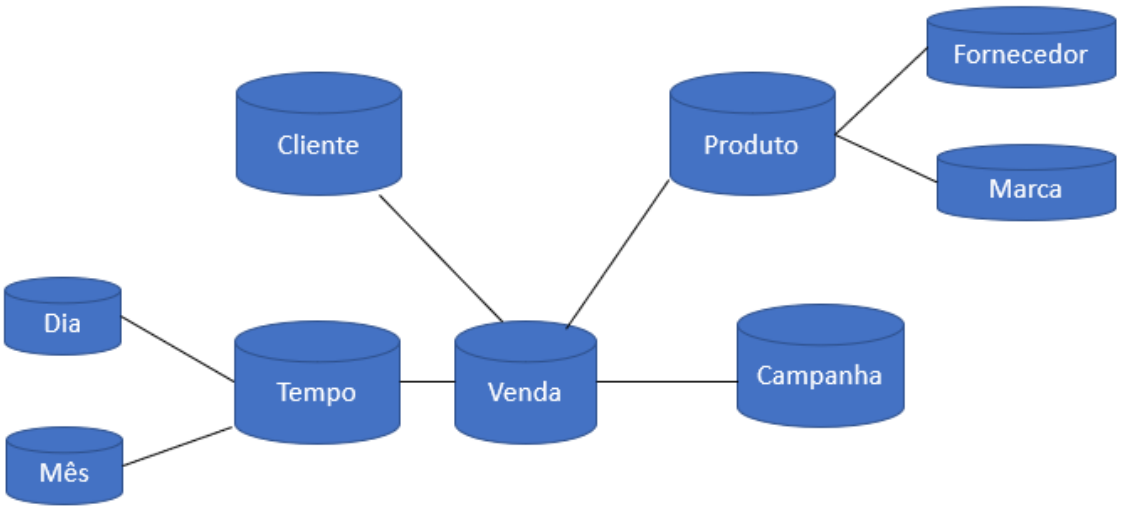
Fonte: Nogueira (2017).

Figura 45 - Modelagem Estrela.



Outra abordagem para modelagem do DW é a Floco de Neve, *Snow Flake*. Nesse paradigma as tabelas dimensões são expandidas conforme a análise a ser realizada. Por exemplo, na Figura 45 a dimensão produto poderia ser expandida para destacar a marca, o fornecedor do produto e assim por diante. A Figura 46 exibe um exemplo de uma modelagem *Snow Flake*.

Figura 46 - Modelagem Snow Flake.



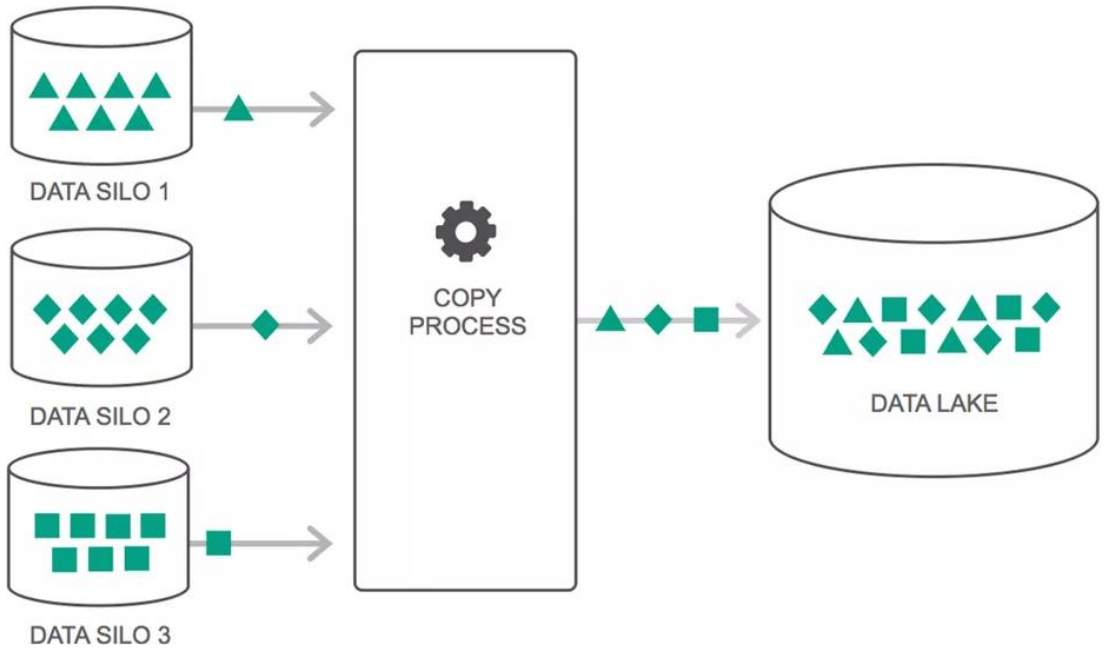
6.2. Data Lake

Data Lake é um termo recente, criado por James Dixon - CTO (*Chief Technical Officer*) do Pentaho. O glossário de TI do Gartner Group define Data Lake como "*a collection of storage instances of various data assets. These assets are stored in a near-exact, or even exact, copy of the source format and are in addition to the originating data stores"*. Cabe destacar no conceito do Gartner, o fato dos dados serem armazenados em estado bruto ou próximo disso (*near-exact, or even exact)*.

Outra definição de *Data Lake* pode ser encontrada em https://en.wiktionary.org/wiki/data_lake: "*A massive, easily accessible data repository built on (relatively) inexpensive computer hardware for storing 'big data'".* Nessa definição cabe destacar que *Data Lake* é enorme (*massive*) e de fácil acesso.

A Figura 47 exibe o fluxo de dados dentro do *Data Lake.* Nessa Figura os dados são copiados das bases locais(silos) e integrados em um único repositório, Data Lake, sem passarem pelo processo de transformação.

Figura 47 - Data Lake.

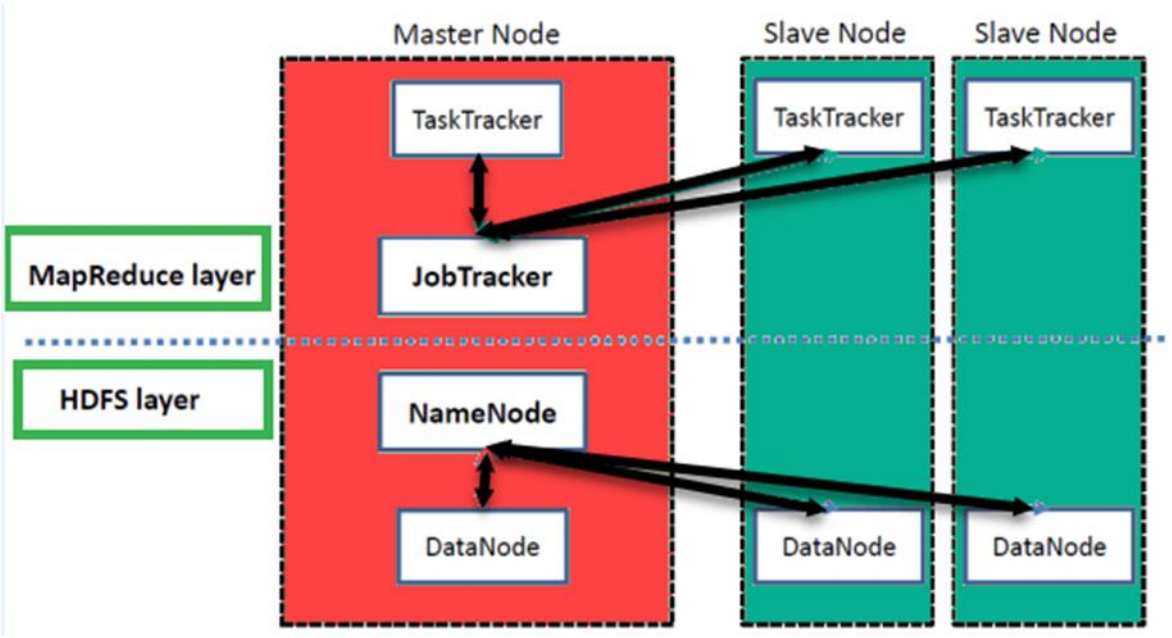


Fonte: http://datascienceacademy.com.br/blog/os-4-estagios-para-construir-um-data-lake-de-forma-eficiente/.

Pode-se dizer que a principal diferença do Data Lake em relação ao DW, é que no Data Lake os dados não são transformados para ir para o repositório, ou seja, seguem o ELT. A transformação dos dados já ocorre no próprio repositório final. A principal motivação é reduzir o tempo dedicado a Preparação dos Dados (80%) no processo de análise de dados.

Uma tecnologia adotada para implementação de Data Lake é o Hadoop, desenvolvido pela Apache. O Hadoop é um software open source que implementa o paradigma de programação paralela Map-Reduce, que possibilita o armazenamento e processamento de grandes volumes de dados. A arquitetura Hadoop mostrada na Figura 48 é estruturada na forma mestre-escravo.

Figura 48 - Arquitetura Hadoop.



Fonte: https://intellipaat.com/blog/tutorial/big-data-and-hadoop-tutorial/the-hadoop-module-high-level-architecture/.

O armazenamento dos dados é feito na camada *Hadoop Distributed File System (HDFS).* O HDFS normalmente divide o arquivo em partes menores, normalmente 64 MB. O Namenome da camada HDFS contém dados sobre os dados, metadados, para controlar a localização e as operações realizadas nesses dados. Os *Datanodes* são responsáveis pelo armazenamento dos dados e de reportar as operações para o Namenode, enquanto a camada MapReduce é responsável pelo processamento paralelo de dados.

O *MapReduce* é divido em duas etapas: *Map* e *Reduce.* Na fase de *Map* ocorre o recebimento dos dados a serem processados, nessa fase o desenvolvedor atribui os dados nós para processamento. Na fase *Reduce* o programador agrega esses dados conjuntos de dados menores.

Capítulo 7. Pipeline de Dados

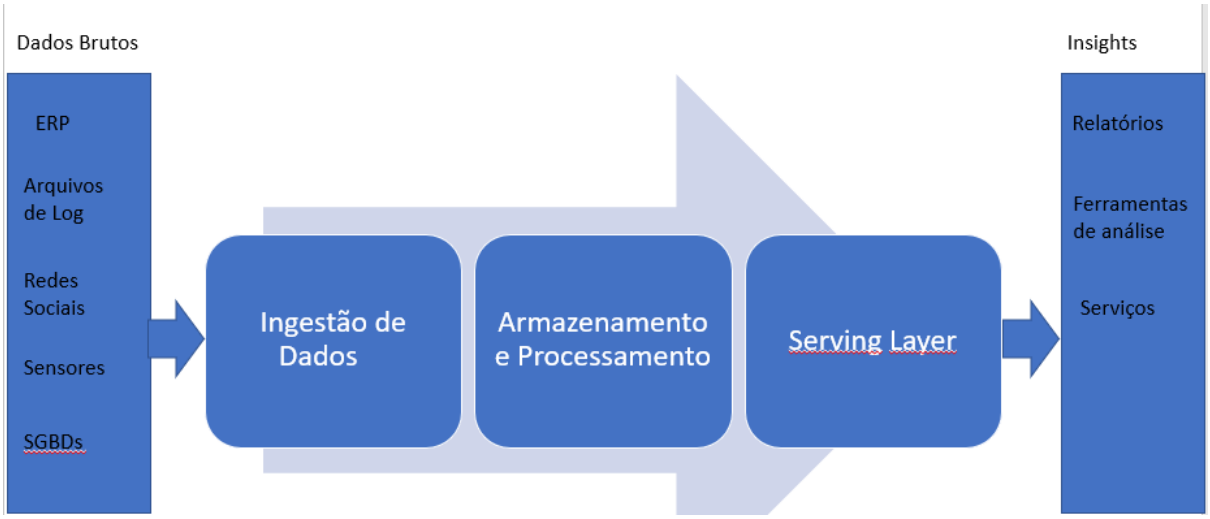
O termo *pipeline*, tradução conduto tabular, é usado em diversas áreas do conhecimento, tais como pipeline de vendas, pipeline de projetos, pipeline de marketing, entre outros. Na Tecnologia da Informação ele também é usado em mais de uma subárea do conhecimento, como por exemplo na arquitetura de computadores. Nesse material, o foco é o pipeline de dados.

O conceito de pipeline de dados não é um conceito tão bem definido. Para Jesse Anderson, especialista em Engenharia de Dados (https://www.jesse-anderson.com/), pipeline de dados pode ser definido como um "processo para transformar dados brutos em dados que possam ser utilizados por toda organização". Jamie Grier, ex-Engenheiro de Software da Data Artisans e do Twitter, define Pipeline de dados como sendo "uma cadeia de processos de dados em que a saída de um processo é a entrada de outro processo". Por fim, Tyler Akidau, escritora do O'Reilly e coautora do livro Streaming Systems, define o pipeline de dados como "qualquer sistema que recupera dados de diversas fontes e, de alguma forma, transforma esses dados antes de carregá-los em uma ou mais saídas".

O pipeline de dados pode ser de dois tipos: *batch* e *stream*. No processo via *batch*, os dados são consumidos periodicamente, como por exemplo em um processo ETL clássico. Por outro lado, no *streaming* os dados são consumidos pelo pipeline em tempo real, sendo esse tipo de *pipeline* aplicado quando existe a necessidade de ingestão de dados em tempo real. O *pipeline* em tempo real normalmente está associado ao *Big Data.* Um terceiro tipo de pipeline de dados é o híbrido, em que algumas etapas do *pipeline* são em lote (*batch)* e outras fases em tempo real.

De um modo geral, pode-se definir o pipeline de dados como uma sequência ordenada de estágios que inicia com a coleta dos dados até que os dados estejam disponíveis para análise. Durante esse processo, os dados brutos são transformados em *insights*. A Figura 49 exibe a estrutura genérica de *pipeline* de dados.

Figura 49 - Pipeline de dados.



Na etapa de Ingestão os dados são coletados de diversas fontes. Uma ferramenta comumente usada para a ingestão de dados *online* é o Apache Kafka[4]. Apache Kafka é uma plataforma de gerenciamento de mensagens pub-sub[5] que conecta fontes de dados a um ou mais coletores. É importante lembrar que a ingestão de dados pode ser realizada em *batch* em períodos definidos, nesse caso o processo é similar ao ETL clássico, que não deixa de ser um *pipeline* de dados.

Na camada de Armazenamento e Processamento, os dados são transformados conforme as técnicas apresentadas nos capítulos anteriores e as *queries* executadas. Nesse estágio pode-se adotar como solução o MPP (*Massively Parallel Processing*) ou uma solução que desacople o processamento do armazenamento. No caso de uma solução que separe o processamento do

[4] https://kafka.apache.org/

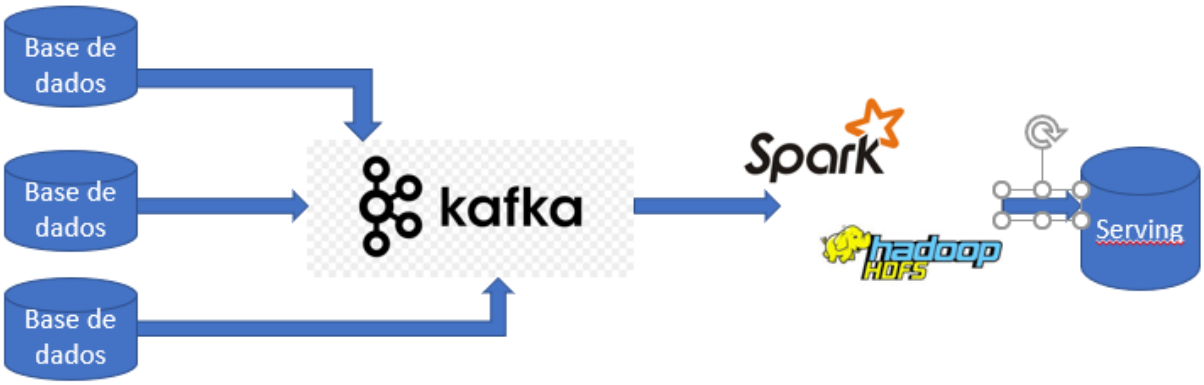
[5] Publish/subscribe (pub/sub) messaging é uma forma de comunicação, assíncrona, de serviço a serviço em arquiteturas sem servidores ou com microsserviços.

armazenamento é comum a adoção do Apache Spark[6] para processamento do *Hadoop* (capítulo anterior desse material), para armazenamento.

Na terceira camada, *Serving Stage*, os dados estão disponíveis para serem consumidos por analistas de dados, decisores que avaliam os relatórios ou mesmo uma aplicação que consulta os resultados desse processo. Normalmente nessa etapa é usado um banco NoSQL.

Diante do exposto, uma possível plataforma tecnológica para o pipeline de dados pode ser expressa na Figura 50.

Figura 50 - Exemplo de Tecnologias de um Pipeline.



Para construção de um *pipeline* de dados existem duas arquiteturas que normalmente podem ser seguidas: *Lambda*[7] e *Kappa*.

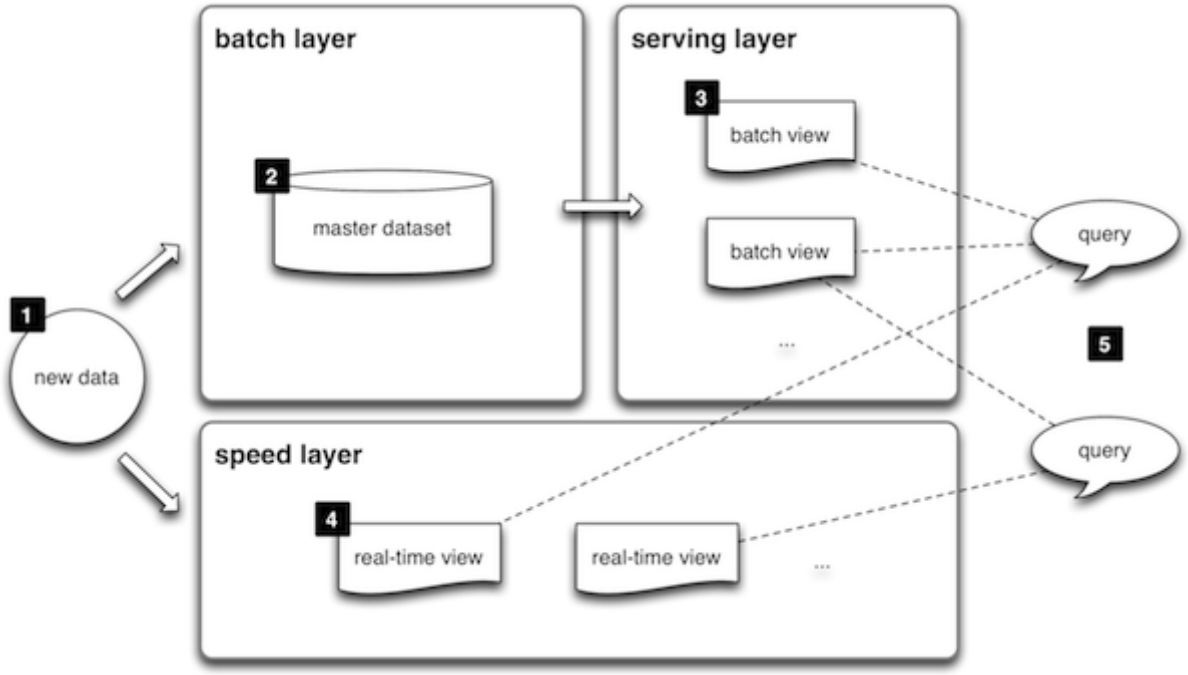
A arquitetura *Lambda* que foi proposta por Nathan Marz é uma arquitetura genérica que tem entre suas premissas ser uma arquitetura com robustez e tolerante a falhas, baixa latência para atualização e leitura de dados, e possuir uma boa escalabilidade. Essa arquitetura lida tanto com o processo *batch* quanto *online* e está

[6] https://spark.apache.org/

[7] http://lambda-architecture.net/

dividida em três camadas: *Batch Layer*, *Speed Layer* e *Serving Layer*, conforme a Figura 51.

Figura 51 - Arquitetura Lambda.



Fonte: http://lambda-architecture.net/.

A arquitetura proposta por Marz pode ser dividida em dois fluxos representados pela numeração da Figura 51: i) sequência 1, 2, 3, 5 e ii) sequência 1, 4 e 5. No primeiro fluxo (1, 2, 3 e 5), o dado bruto, representado na figura pelo número 1, é armazenado no número 2 da figura, de forma não volátil, usando, por exemplo, Apache *Hadoop* para posterior processamento. Após o armazenamento, os dados são disponibilizados na *Serving Layer,* número 3 da Figura 51*,* para consultas *ad-hoc e* com alta latência. Uma plataforma que pode ser usada na camada Serving é o Apache Cassandra[8].

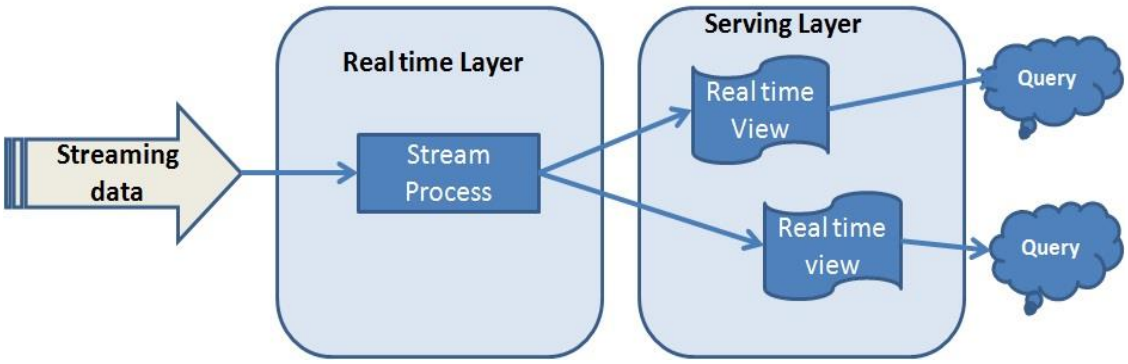
[8] http://cassandra.apache.org/

No segundo fluxo, a *Speed Layer*, número 4 da Figura 51, cria análises em tempo real para compensar a alta latência da *Serving Layer* e disponibiliza essa visão para consultas próximas ao tempo real.

Importante destacar que os dois fluxos, *batch e streaming*, são complementares, em outras palavras, uma camada não é mais importante do que a outra.

A arquitetura Kappa[9] proposta por Jay Kreps é uma simplificação da arquitetura Lambda em que o processo *batch, Batch Layer,* foi removido. Assim, conforme a Figura 52, a arquitetura Kappa é estruturada em duas camadas: *Real Time Layer* e *Seving Layer*.

Figura 52 - Arquitetura Kappa.



Fonte: https://www.whizlabs.com/blog/real-time-big-data-pipeline/.

Nessa arquitetura o dado é ingerido pela camada de tempo real, *Real Time Layer*, usando, por exemplo, Apache Kafka. Ainda na *Real Time Layer* o dado ingerido é processado. O dado resultante do processamento no primeiro estágio da arquitetura é enviado para a *Serving Layer* e disponibilizado para consultas.

[9] http://milinda.pathirage.org/kappa-architecture.com/.

A arquitetura Kappa possui quatro princípios que devem ser observados, são eles:

1. **Tudo é *Stream***: processamento em *batch* é um subconjunto do *stream*.
2. **Fontes de Dados Imutáveis**: os dados brutos são persistidos e, a partir desses dados, as visões são derivadas. Um novo estado dos dados pode ser extraído, pois os dados persistidos não sofrem alterações.
3. ***Frameworks* simples para análise de dados**: codificação, manutenção e atualizações são reduzidas.
4. **Reprodutibilidade**: possibilidade de reprodução de resultados a partir dos dados históricos.

Cada uma das arquiteturas apresentadas anteriormente, Lambda e Kappa, possuem suas vantagens e desvantagens. Por exemplo, a arquitetura possui um bom balanceamento entre confiabilidade e velocidade. Por outro lado, a modelagem de dados na arquitetura Lambda é difícil de ser migrada para outras arquiteturas. A arquitetura Kappa possui entre suas vantagens o consumo de poucos recursos, pois a análise de dados é realizada baseada em tempo real. Entre as desvantagens da arquitetura Kappa está a ausência da camada *batch* que pode implicar em erros durante o processamento.

Capítulo 8. Mineração de Texto

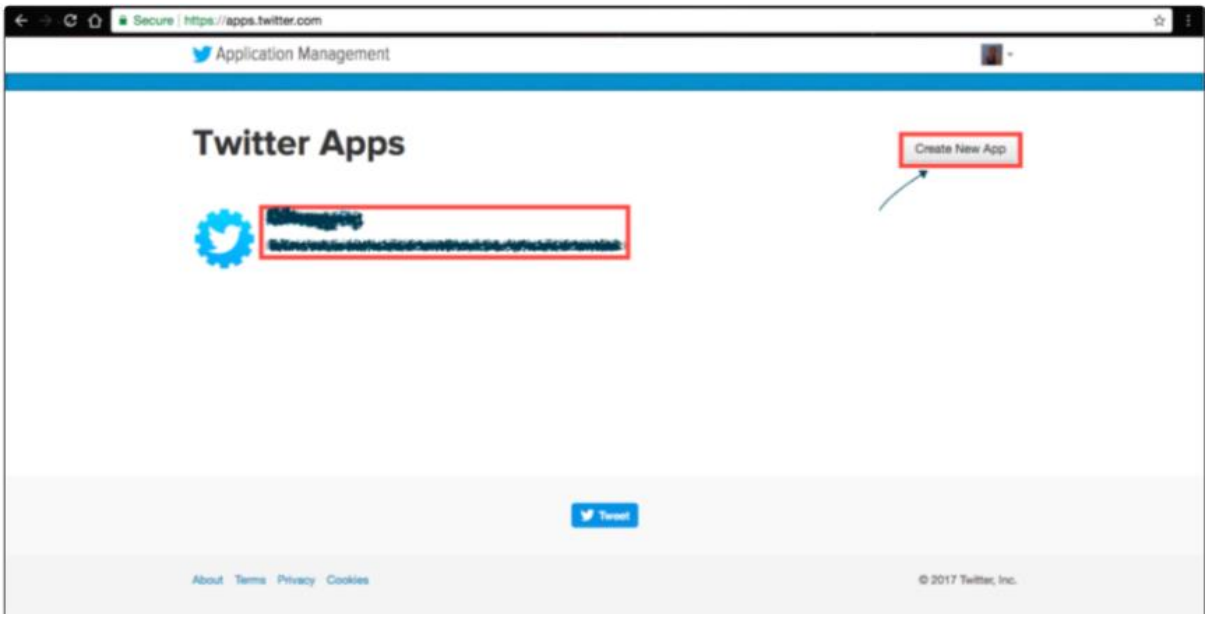
No Capítulo 2 deste material, foram apresentadas algumas técnicas e fontes para coleta de dados. Neste capítulo, será abordada a coleta de dados em Redes Sociais e a preparação de dados a partir de textos, a Mineração de Texto, *Text Mining*. O processo de mineração de textos é similar ao processo de mineração de dados, porém os dados a serem minerados são desestruturados, textos produzidos em linguagem natural. Portanto, a mineração de texto pode ser definida como o processo de extrair informação, previamente desconhecida, de maneira automática, ou semiautomática, a partir de bases de dados textuais.

Diversas são as aplicações possíveis usando a mineração de texto, tais como: análise de sentimentos, classificação de opiniões, análise de discursos, classificação de documentos, clusterização, entre outras tarefas de aprendizado de máquina.

Assim, como no processo de mineração de dados, é necessário coletar os textos a serem a minerados. Esses textos podem ser oriundos de diversas fontes, tais como Redes Sociais (Twitter, Instagram, Facebook), revisões de leitores sobre filmes, hospedagens, jornais e revistas, discursos, entre outras diversas fontes textuais.

Para performar a coleta de dados do Twitter, é necessária ter uma conta do Twitter e criar um App para ter acesso a opção de desenvolvedor. A Figura 53 exibe onde criar a App no Twitter.

Figura 53 - Configuração App Twitter.

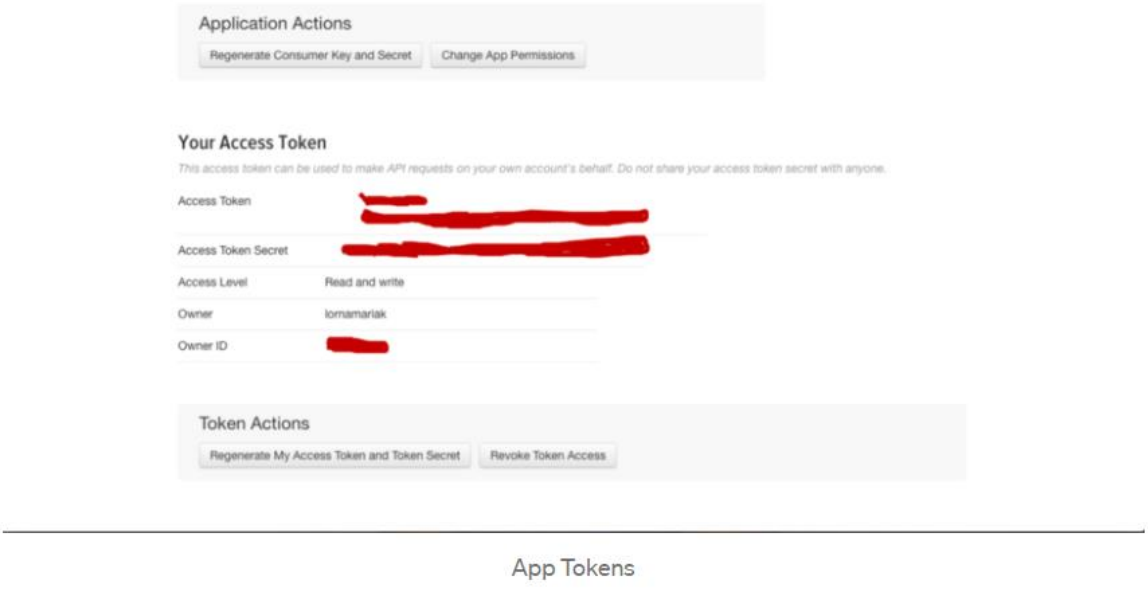


Fonte: apps.twitter.com.

Durante o processo de criação de App, são solicitadas várias informações relativas ao App que está sendo criada e informações sobre o *token* de acesso.

Usando a linguagem R, pacote *twitteR*, a função para conexão ao Twitter é *setup_twitter_oauth(consumer_key, consumer_secret, access_token, access_secret)*. Os parâmetros da função são preenchidos durante a criação do App, conforme a Figura 54.

Figura 54 - Configuração da conta desenvolvedor.



App Tokens

Fonte: apps.twitter.com.

Uma vez conectado ao Twitter, é possível coletar os dados desejados, através, por exemplo, da função searchTwitter. A sequência de comando abaixo retorna uma lista com 5000 (n= 5000) Tweets com a palavra-chave “cpicovid” escritas em português (lang=”PT”).

```
> tweets <- searchTwitter("cpicovid", n=5000, lang="pt")
> class(tweets)
[1] "list"
```

Uma vez coletado os textos a serem analisados, assim como na mineração de dados, é necessário realizar o pré-processamento, a preparação, do texto para posterior análise. Conforme Deler 2012, o pré-processamento de texto pode ser estruturado nas seguintes etapas:

a) Escopo do texto;

b) Tokenização – Análise Léxica;

c) Remoção das *stopwords*;

d) Stemmização;

e) Normalização de grafia;

f) Identificação de sentenças;

g) Padronização da capitalização.

A primeira etapa do pré-processamento, escopo, está relacionada à definição do fragmento do texto. Por exemplo, será usado o texto inteiro, a análise será por parágrafo do texto, serão analisadas postagens separadas, e-mail, entre outras opções. A definição do escopo depende da análise a ser realizada.

Na segunda etapa, *tokenização*, é realizada a análise léxica. Nessa etapa ocorre a divisão do texto em *tokens* ou palavras, através da eliminação de marcações, símbolos, caracteres especiais e espaços, sendo que a estratégia de *tokenização* depende do idioma do texto. Uma estratégia comum para textos em português e a identificação de pontuação e espaços em brancos. A seguir, um exemplo de tokenização, com o texto de entrada e os tokens identificados no texto.

Texto de Entrada	Tokens identificados
“Nessa etapa ocorre a divisão do texto em *tokens* ou palavras, através da eliminação de marcações, símbolos, caracteres especiais e espaços”	“Nessa” “etapa” “ocorre” “a” “divisão” “do” “texto” “em” “tokens” “ou” “palavras” “através” “da” “eliminação” “de” “marcações” “símbolos” “caracteres” “especiais” “e” “espaços”

Na linguagem R pode ser usado o pacote *tokenizers* para gerar os tokens usando a função *tokenize_words*. A Figura 55 exibe um exemplo de análise léxica usando o R.

Figura 55 - Exemplo do processo de tokenização.

```
> library(tokenizers)
> texto_token = paste0("Nessa etapa ocorre a divisão do texto em tokens ou palavras,
+                      através da eliminação de marcações, símbolos, caracteres especiais e espaços")
> tokenize_words(texto_token)
[[1]]
 [1] "nessa"      "etapa"      "ocorre"     "a"          "divisão"    "do"         "texto"      "em"
 [9] "tokens"     "ou"         "palavras"   "através"    "da"         "eliminação" "de"         "marcações"
[17] "símbolos"   "caracteres" "especiais"  "e"          "espaços"
```

Após a *tokenização*, é realizada a remoção das *stopwords*. *Stopwords* são Termos Irregulares que normalmente possuem alta frequência nos textos e não agregam conhecimento na análise do texto. As *stopwords* normalmente são os termos mais frequentes na coleção, constituídos de verbos auxiliares, artigos, preposições, entre outros termos que devem ser removidos antes da análise.

A Figura 56 exibe alguns exemplos de *Stopwords*, sendo que durante o processo de preparação do texto novas palavras podem ser adicionadas.

Figura 56 - Exemplo de stopwords.

```
> stopwords("pt")
  [1] "de"           "a"            "o"            "que"          "e"            "do"
  [7] "da"           "em"           "um"           "para"         "com"          "não"
 [13] "uma"          "os"           "no"           "se"           "na"           "por"
 [19] "mais"         "as"           "dos"          "como"         "mas"          "ao"
 [25] "ele"          "das"          "à"            "seu"          "sua"          "ou"
 [31] "quando"       "muito"        "nos"          "já"           "eu"           "também"
 [37] "só"           "pelo"         "pela"         "até"          "isso"         "ela"
 [43] "entre"        "depois"       "sem"          "mesmo"        "aos"          "seus"
 [49] "quem"         "nas"          "me"           "esse"         "eles"         "você"
 [55] "essa"         "num"          "nem"          "suas"         "meu"          "às"
 [61] "minha"        "numa"         "pelos"        "elas"         "qual"         "nós"
 [67] "lhe"          "deles"        "essas"        "esses"        "pelas"        "este"
 [73] "dele"         "tu"           "te"           "vocês"        "vos"          "lhes"
 [79] "meus"         "minhas"       "teu"          "tua"          "teus"         "tuas"
 [85] "nosso"        "nossa"        "nossos"       "nossas"       "dela"         "delas"
 [91] "esta"         "estes"        "estas"        "aquele"       "aquela"       "aqueles"
 [97] "aquelas"      "isto"         "aquilo"       "estou"        "está"         "estamos"
[103] "estão"        "estive"       "esteve"       "estivemos"    "estiveram"    "estava"
[109] "estávamos"    "estavam"      "estivera"     "estivéramos"  "esteja"       "estejamos"
[115] "estejam"      "estivesse"    "estivéssemos" "estivessem"   "estiver"      "estivermos"
[121] "estiverem"    "hei"          "há"           "havemos"      "hão"          "houve"
[127] "houvemos"     "houveram"     "houvera"      "houvéramos"   "haja"         "hajamos"
[133] "hajam"        "houvesse"     "houvéssemos"  "houvessem"    "houver"       "houvermos"
[139] "houverem"     "houverei"     "houverá"      "houveremos"   "houverão"     "houveria"
[145] "houveríamos"  "houveriam"    "sou"          "somos"        "são"          "era"
[151] "éramos"       "eram"         "fui"          "foi"          "fomos"        "foram"
[157] "fora"         "fôramos"      "seja"         "sejamos"      "sejam"        "fosse"
[163] "fôssemos"     "fossem"       "for"          "formos"       "forem"        "serei"
[169] "será"         "seremos"      "serão"        "seria"        "seríamos"     "seriam"
```

Após a remoção dos termos irregulares, *stopwords,* é realizada a normalização morfológica, *stemming*, dos termos. O processo de *stemming* consiste na redução das palavras ao seu radical eliminando prefixos, sufixos, plurais e inflexão de gêneros.

A Figura 57 exibe a normalização morfológica usando a função *stemDocument* da biblioteca tm da linguagem R.

Figura 57 - Exemplo de stemmização.

```
> stemDocument(texto_token, language = "portuguese")
[1] "Ness etap ocorr a divisã do text em tokens ou palavras, através da elimin de marcações, símbolos, caract es
pec e espac"
>
```

Observe na Figura 57 que a palavra “Nessa” foi reduzida a Ness, pois “Nessa”, “Nesse” “nessas”, “nesses” possuem o mesmo radical “Ness”. O processo de *stemming* reduz significativamente o número de termos.

Conforme Deler (2012), na etapa de padronização de grafia ocorrem procedimentos de correções de grafia. Na sequência, ocorre a identificação de sentenças a partir da pontuação.

Na última etapa proposta por Deler (2012), ocorre padronização da capitalização (maiúscula ou minúscula) das letras.

Na sequência, será apresentado a implementação de um processo de coleta e preparação de dados de Tweets. O pacote do R utilizado para apoiar nesses processos foi o “tm”. A Figura 58 exibe a conexão com o Twitter e a busca por 5000 tweets, sendo “cpicovid” o termo de busca utilizado. Após a busca, a relação de Tweets é transformada em DataFrame, função *twListToDF*. Observe que o DataFrame possui 5000 linhas, 5000 tweets e 16 colunas. As colunas do DataFrame contêm, além dos textos dos tweets, outras informações, tais como, o apelido de quem tuitou, a quantidade de *retweets* daquele tweet, data e hora da postagem, entre outros dados para análise.

Figura 58 - Conexão e busca no Twitter.

```
> setup_twitter_oauth(consumer_key, consumer_secret, access_token, access_secret)
[1] "Using direct authentication"
> tweets <- searchTwitter("cpicovid", n=5000, lang="pt")
> dados <- twListToDF(tweets)
> dim(dados)
[1] 5000   16
>
```

A Figura 59 exibe as linhas de código da preparação do Tweets. Na linha 10 foi criado o Corpus. Corpus é um conjunto de textos, nesse caso um conjunto de 5000 tweets. Na linha 11 os textos são convertidos para minúsculas, na 12 as pontuações são removidas, na linha 13 são removidos os espaços extras, na 14 são removidos números e na linha 15 são removidas as palavras irregulares que não agregam valor na análise.

Figura 59 - Exemplo de Código para pré-processamento.

```
corpus <- Corpus(VectorSource(dados$text))
corpus <- tm_map(corpus, tolower)
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, removeWords, stopwords('pt'))
tdm <- as.matrix((TermDocumentMatrix(corpus, control=list(stemming = TRUE))))
```

Na linha 16, usando a função TermDocumentMatrix, é criada uma matriz com os termos contidos em cada documento. Os termos estão representados nas linhas e os documentos, tweets, nas colunas. Assim, a matriz “tdm” possui 5000 colunas, quantidade de tweets, e 3550 colunas representando os termos que foram identificados. Observe que na linha 16 foi realizado a stemmização através do parâmetro stemming = TRUE. A título de ilustração, caso o parâmetro *stemming* fosse setado como FALSE, o número de termos seria um pouco maior, neste exemplo, seriam 3684 termos.

A partir deste ponto, a base está pronta para ser analisada. Naturalmente, a partir de uma análise prévia, novos ajustes podem ser necessários como, por exemplo, a inclusão de outras palavras na relação de stopwords, a opção por não performar a stemmização, entre outras decisões de projeto.

A Figura 60 exibe um pequeno trecho de código que realiza a análise dos tweets. Inicialmente, foram somadas as linhas para contar a quantidade de vezes que os termos aparecem nos Tweets. Observe que os seis primeiros termos são: “brasil – 2206”, “cpi – 1668”, “cloroquina – 1433”, “capitã – 1251”, “diz – 1015” e “contra – 987”. A partir desses termos, pode-se observar que um termo que poderia ser removido é o termo “diz”, pois não agrega à análise.

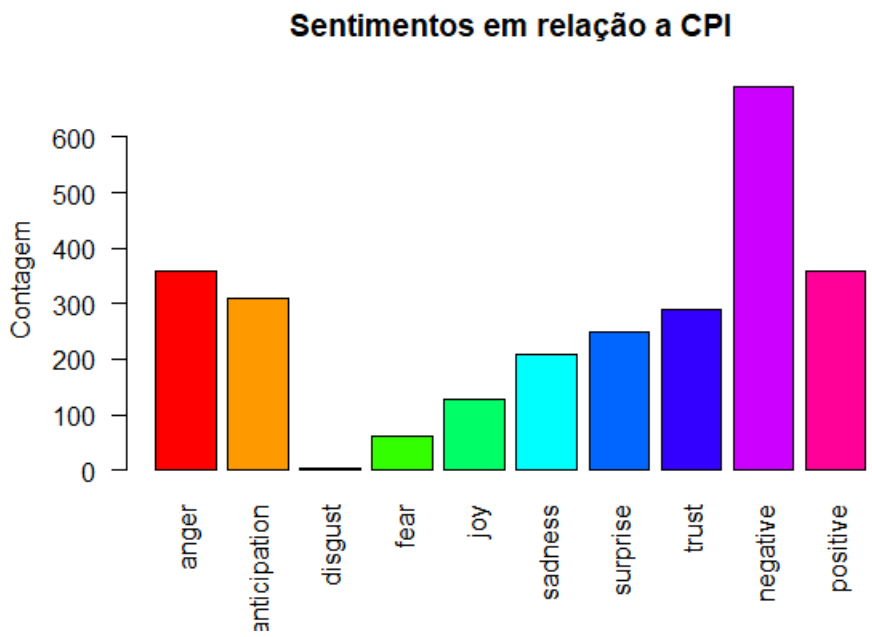
Figura 60 - Exemplo de análise de texto.

```
> fre <- rowSums(tdm)
> fre <- sort(fre,decreasing = TRUE)
> head(fre)
    brasil        cpi cloroquina     capitã       diz     contra
      2206       1668       1433       1251       1015        987
> wordcloud(corpus, min.freq = 30, max.words = 30, random.order = FALSE,rot.per = 0.35, colors=brewer.pal(8,"Dar
k2"))
> s<- get_nrc_sentiment(dados$text)
> barplot(colSums(s), las=2, col = rainbow(10), ylab= "Contagem",
+         main = "Sentimentos em relação a CPI" )
> |
```

A função *worlcloud* da Figura 60 cria uma nuvem de palavra com as palavras mais citadas, Figura 61. A função *get_nrc_sentiment* faz a análise dos sentimentos dos tweets e a função barplot gera um gráfico de barras, conforme Figura 62.

Figura 61 - Nuvem de palavras.

Figura 62 - Gráficos de barras exibindo a classificação dos Tweets.



Como foi possível observar nesse pequeno exemplo, após a preparação do texto, novos ajustes podem ser necessários antes de partir para a fase de análise. Outras análises de textos podem ser realizadas como clusterização, classificação, entre outras tarefas de mineração.

Referências

BRYSON, S. *et al.* Visually exploring gigabyte data sets in real time. *In*: *Communications of the ACM*. USA, New York: Communications of the ACM, v. 42, n. 8, p. 82–90. 1999. ISSN 0001-0782. Disponível em: http://doi.acm.org/10.1145/310930. Acesso em: 01 nov. 2021.

DOAN, A.; DOMINGOS, P. HALEVY, A. *Reconciling Schemas of Disparate Data Sources*: A Machine-Learning Approach. 2001. Disponível em: https://homes.cs.washington.edu/~pedrod/papers/sigmod01.pdf. Acesso em: 01 nov. 2021.

ELMAGARMID, A. K.; IPEIROTIS, P. G.; VERYKIOS, V. S. *Duplicate record detection*: A survey. IEEE Transactions on Knowledge and Data Engineering, v. 19, n. 1, p. 1–16, 2007.

GARCÍA, Salvador; LUENGO, Julián; HERRERA, Francisco. *Data Preprocessing in Data Mining*. Springer International Publishing. v. 73. n. 1, 2015.

INMON, W. H. *Building the Data Warehouse*. USA, New York: John Wiley & Sons, Inc., 1992.

KIMBALL, R. *The Data Warehouse Toolkit*: Practical Techniques for Building Dimensional Data Warehouses. 1. ed. USA, New York: John Wiley & Sons, Inc., 1996.

MINGOTI, S. *Análise de Dados Através de Métodos de Estatística Multivariada*: Uma Abordagem Aplicada. 1. ed. UFMG, 2007.

NOGUEIRA, R. R. *Newsminer*: um sistema de data warehouse baseado em texto de notícias. 2017. Dissertação (Programa de Pós-graduação em Ciência da Computação (Campus SOROCABA)) - Universidade Federal de São Carlos - UFSCar, 2017.

REHMAN, M. H. *et al.* Big data reduction framework for value creation in sustainable enterprises. In: *International Journal of Information Management*. v. 36. n. 6. p. 917-928. 2016. Disponível em:

https://www.sciencedirect.com/science/article/pii/S0268401216303097?via%3Dihub#!. Acesso em: 01 nov. 2021.

SEYVET, N.; VIELA, I. M. *Applying the Kappa architecture in the telco industry*. 2016. Disponível em: https://www.oreilly.com/ideas/applying-the-kappa-architecture-in-the-telco-industry. Acesso em: 01 nov. 2021.

TRIGUERO, I. *et al*. Transforming big data into smart data: An insight on the use of the k-nearest neighbors algorithm to obtain quality data. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*. v. 9. n. 2. p. 1289. 2018. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1289. Acesso em: 01 nov. 2021.