Introdução ao Tratamento e Análise de Dados em R

Aula 3 - Acessando e Utilizando Bases de Dados

Sérgio Rivero

PPGE-UFPA

15 de maio de 2019

Banpará

Sumário

Banpará

Objetivos da Aula

Nesta Aula queremos:

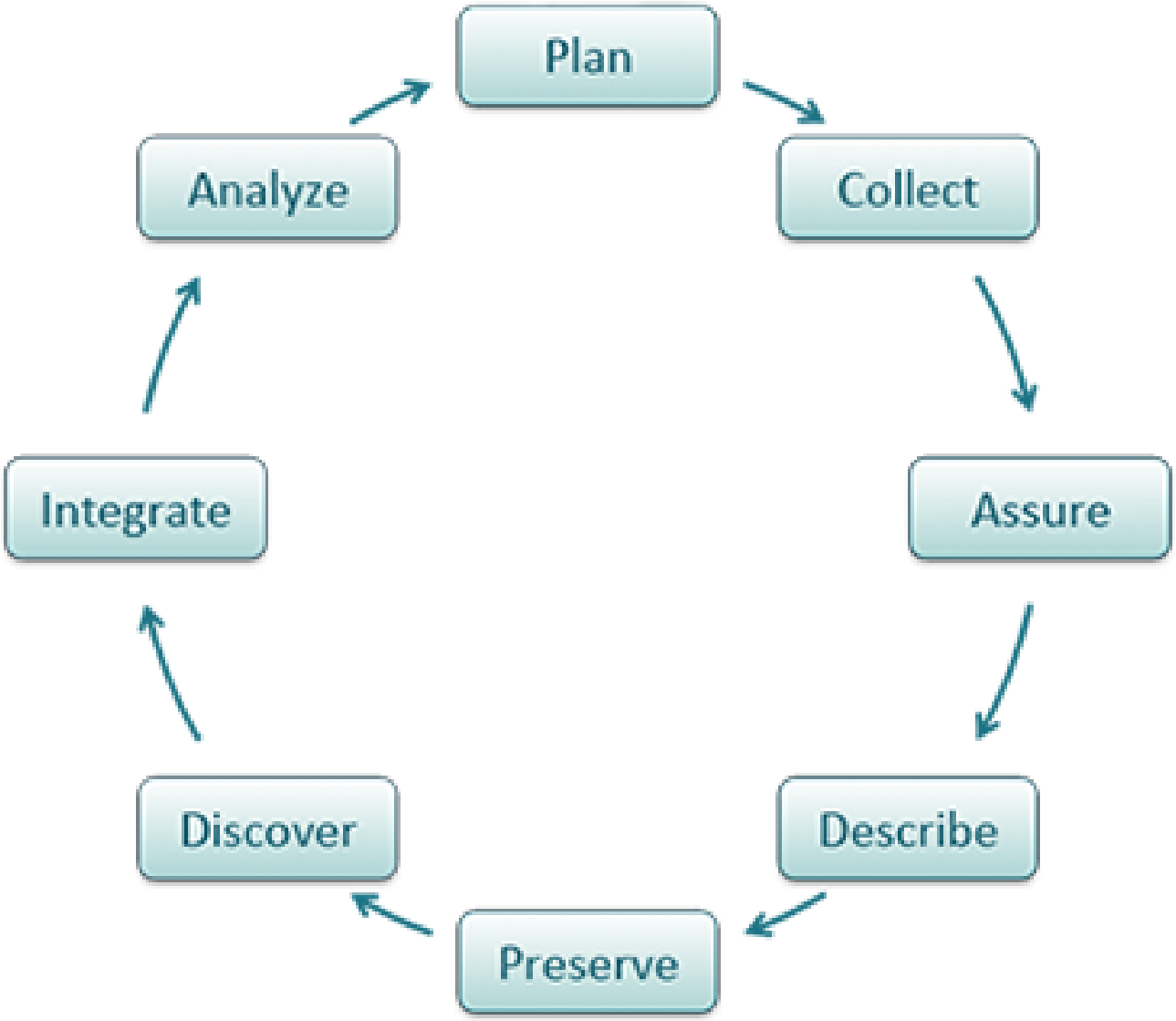
1. Apresentar o conceito de Dataframe;
2. Apresentar os tipos de dados utilizados no R;
3. Apresentar os principais comandos;

O ciclo de tratamento e análise de dados

- Atividades que utilizam dados e fazem algum tratamento formal (estatístico, matemático) destes dados são comuns
- Muitas vezes, os dados correspondem a milhões de observações em diversas bases de dados.
- Estas bases de dados, muitas vezes têm estruturas complexas e diferentes formas de organização
- Tratar e analisar estes dados exige planejamento, avaliação, organização;
- Muitos dados podem ser analisados várias vezes,e podem ter um reuso para diferentes projetos e análises.
- Esta possibilidade de reuso torna ainda mais importante o seu tratamento e armazenamento sistematizado.
- Dados têm um *ciclo de vida* e este ciclo pode ser pensado como um conjunto de atividades que vão do planejamento de seu uso até a sua análise.

Banpará

O Ciclo

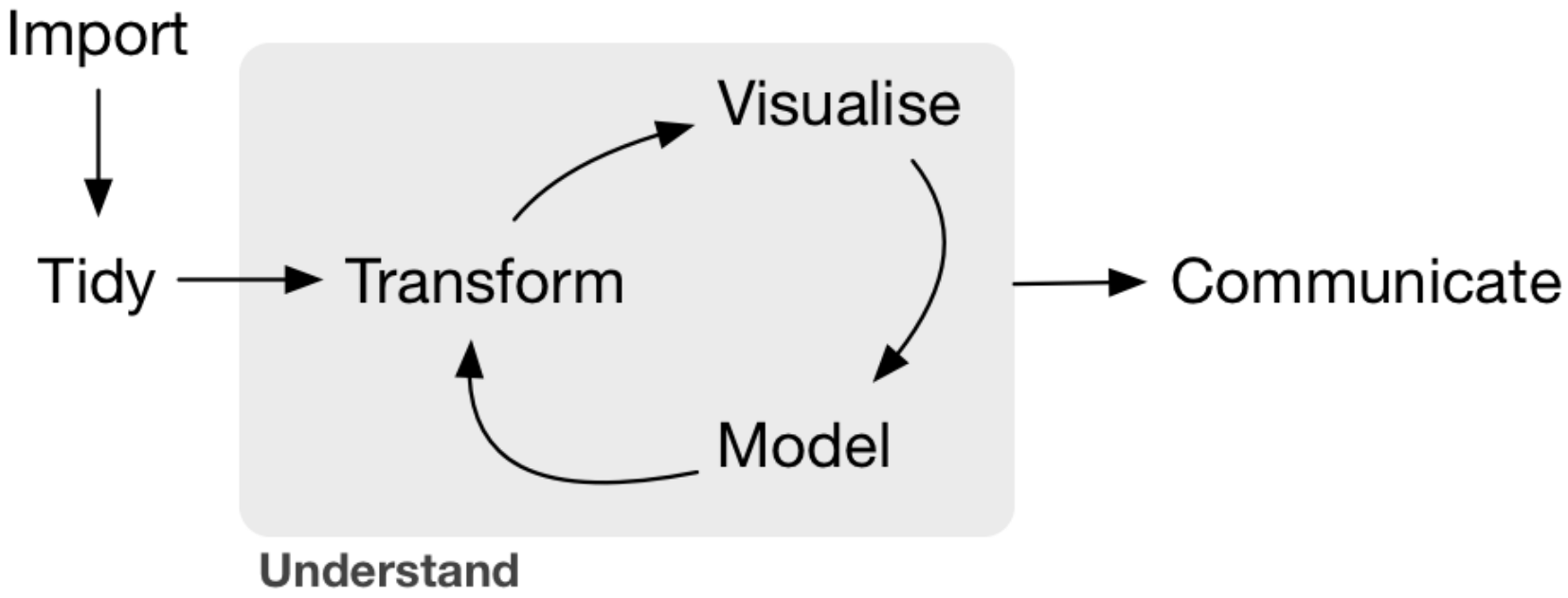


https://www.dataone.org/data-life-cycle

Detalhando o ciclo

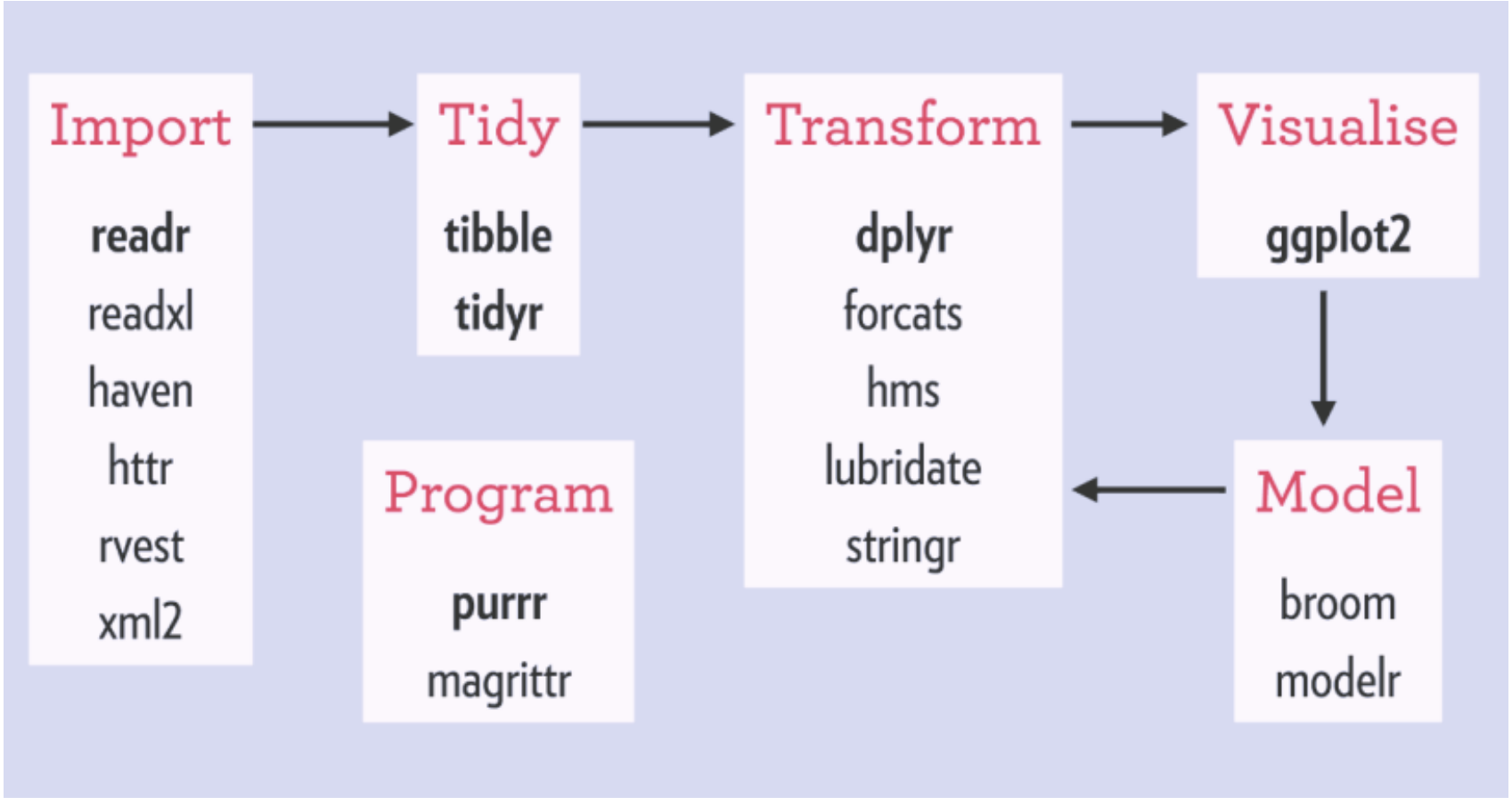
- Planejar: como dados que serão compilados gerenciados e disponibilizados ao longo de sua vida útil
- Coletar: como os dados serão adquiridos
- Validar: assegurar a qualidade dos dados
- Descrever: produzir os metadados apropriados
- Preservar: como os dados serão armazenados e preservados
- Descobrir: avaliar quais dados são relevantes para a análise
- Integrar: combinar os dados, gerar novos dados
- Analisar: os dados são analisados

O ciclo de tratamento de dados e modelagem



Tratamento de Dados Estatísticos

Ferramentas para tratamento de dados em R



Ferramentas de Tratamento de Dados em R

Pacotes para importação de dados em R

- *readr* - genérico para dados em formato matricial (*retangular data*), tais como arquivos csv, tsv, fwf.
- *haven* - para importar arquivos SPSS, Stata, e SAS;
- *readxl* e *xlsx* - arquivos excel (.xls e .xlsx);
- *DBI* - arquivos de bancos de dados;
- *jsonlite* - arquivos *json*;
- *xml2* - arquivos padrão *XML*
- *httr* - *APIs* na *Web*

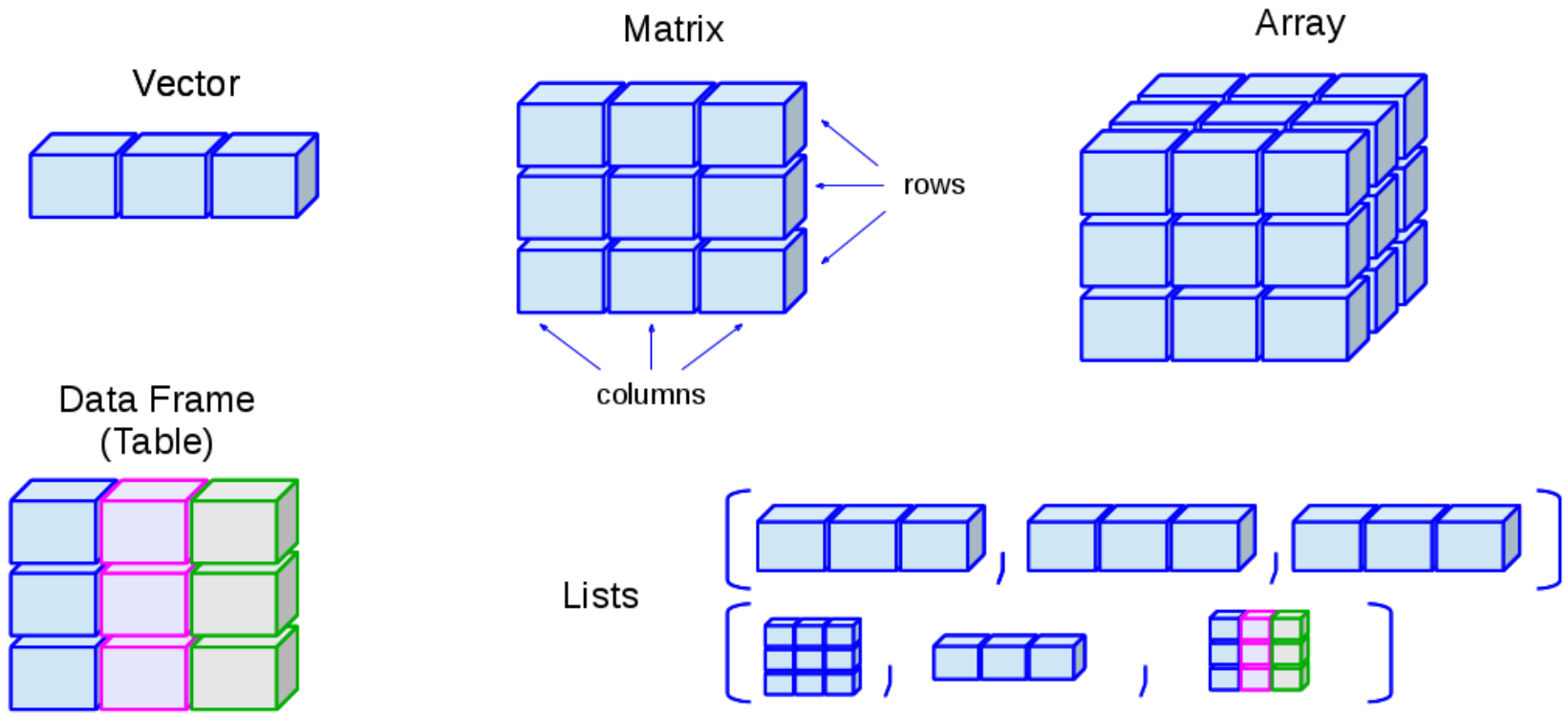
O *tidyverse* tem pacotes que fornecem funcionalidades para executar eficazmente as atividades de tratamentos e análise de dados, reduzindo o esforço de codificação e a possibilidade de erros.

O que são dados limpos (tidy data)?

Os dados limpos têm algumas características fundamentais:

1. Cada coluna da sua base de dados tem apenas um tipo de dado e corresponde a apenas uma variável;
2. Cada linha de sua base de dados corresponde a apenas uma observação;
3. Cada elemento da base corresponde a um dado de uma variável em uma observação.
4. Preferencialmente, o número de observações faltantes é mínimo.

Tipos de dados em R



Uma representação gráfica dos tipos de dados em R.
Fonte: `http://venus.ifca.unican.es/Rintro/index.html`

Escalares

Determinada variável pode ser um escalar, ou seja, simplesmente um número:
Exemplo de escalares:

```
> a <- 2
> b <- 2*a
> a
```

```
[1] 2
```

```
> b
```

```
[1] 4
```

Banpará

Vetores

O vetor é um objeto matemático caracterizado em um conjunto de segmentos orientados de reta que possuem o mesmo módulo, direção e sentido. Ele contêm elementos de classes diferentes, conforme apresentado abaixo:

- *a <- c((1,3500,5.3,543,-2,4000)*
- *b <- c("taxa de juros","taxa de câmbio","reservas bancárias")*
- *c <- c(FALSE,TRUE,FALSE,TRUE)*

Matrizes

```
matriz <- matrix(data=1:16,nrow=4,ncol=4)
matriz
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

Onde:

- data: parâmetro que representa os dados usados para criar matriz;
- nrow: parâmetro para número de linhas;
- ncol: parâmetro para número de colunas.

Array

Essa estrutura de dados possui três dimensões, as linhas, as colunas e as camadas.

cubo <- array(data = 1:27, dim=c(3,3,3))

cubo

```
, , 1

     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

, , 2

     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18

, , 3

     [,1] [,2] [,3]
[1,]   19   22   25
[2,]   20   23   26
[3,]   21   24   27
```

data: parâmetro que representa os dados usados para criar matriz;

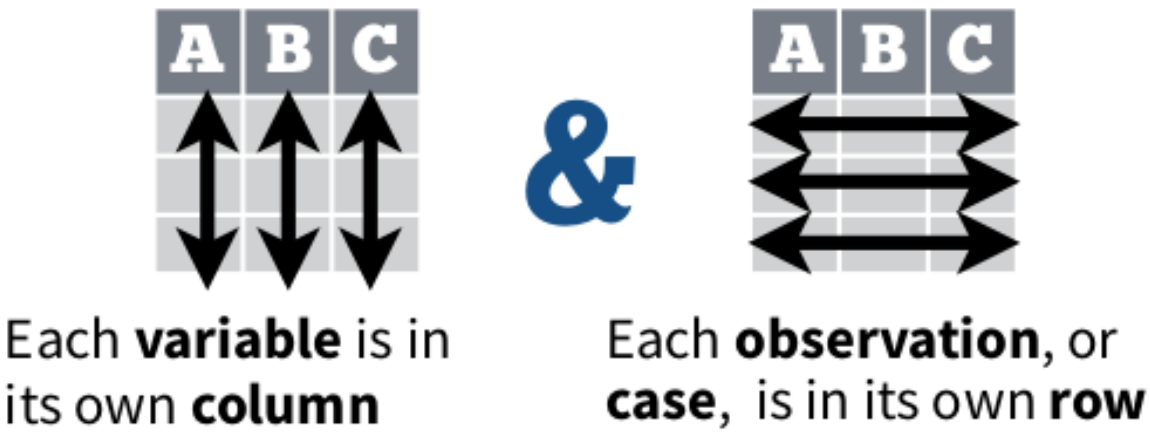
dim: parâmetro para determinar as dimensões do array, sendo **dim** um vetor;

Banpará

Dataframes

Dataframe é uma formatação de tabela presente no R que comporta duas dimensões.

- A primeira dimensão compreende as linhas(Observações)
- segunda dimensão compreende as colunas(variáveis).



Variaveis e observações num Dataframe

Estrutura de um Dataframe

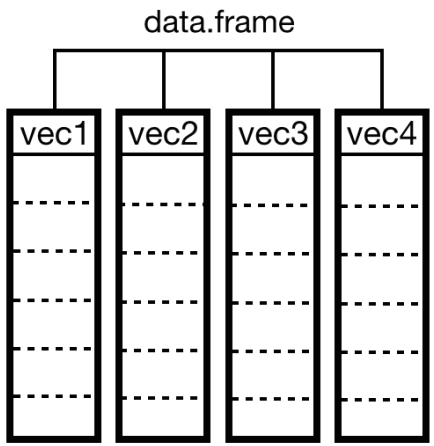


table1

country	year	cases	pop
Afghan	1999	745	19987071
Afghan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Criando um Dataframe

Vetor dos meses:

```
> DATA <- c("ago/2018", "set/2018", "out/2018",
+           "nov/2018", "dez/2018", "jan/2019")
> DATA
```

```
[1] "ago/2018" "set/2018" "out/2018" "nov/2018" "dez/2018" "jan/2019"
```

Vetor do IPCA para os respectivos meses do vetor DATA:

```
> IPCA <- c(-0.09, 0.48, 0.45, -0.21, 0.15, 0.32)
> IPCA
```

```
[1] -0.09  0.48  0.45 -0.21  0.15  0.32
```

Vetor do Pib mensal em milhões(R$) para os respectivos meses do vetor DATA:

```
> PIBmensalMilhoes <- c(583011.3, 551215.6, 597218.7,
+                       604073.9, 624464.1, 591715.7)
> PIBmensalMilhoes
```

```
[1] 583011.3 551215.6 597218.7 604073.9 624464.1 591715.7
```

Banpará

O Dataframe

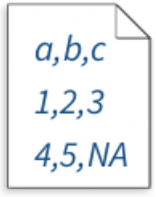
```
> Dados <- data.frame(cbind(DATA, IPCA, PIBmensalMilhoes))
> Dados
```

	DATA	IPCA	PIBmensalMilhoes
1	ago/2018	-0.09	583011.3
2	set/2018	0.48	551215.6
3	out/2018	0.45	597218.7
4	nov/2018	-0.21	604073.9
5	dez/2018	0.15	624464.1
6	jan/2019	0.32	591715.7

Banpará

Importando arquivos tabulares - *readr*

read_*(file, col_names = TRUE, col_types = NULL, locale = default_locale(), na = c("", "NA"), quoted_na = TRUE, comment = "", trim_ws = TRUE, skip = 0, n_max = Inf, guess_max = min(1000, n_max), progress = interactive())



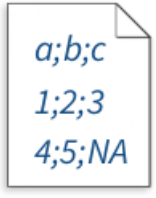
A	B	C
1	2	3
4	5	NA

Comma Delimited Files

read_csv("file.csv"**)**

To make file.csv run:

write_file(x = "a,b,c\n1,2,3\n4,5,NA", path = "file.csv")

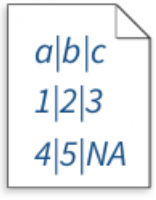


A	B	C
1	2	3
4	5	NA

Semi-colon Delimited Files

read_csv2("file2.csv"**)**

write_file(x = "a;b;c\n1;2;3\n4;5;NA", path = "file2.csv")

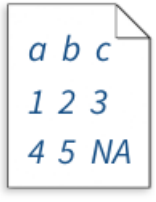


A	B	C
1	2	3
4	5	NA

Files with Any Delimiter

read_delim("file.txt", delim = "|"**)**

write_file(x = "a|b|c\n1|2|3\n4|5|NA", path = "file.txt")



A	B	C
1	2	3
4	5	NA

Fixed Width Files

read_fwf("file.fwf", col_positions = c(1, 3, 5)**)**

write_file(x = "a b c\n1 2 3\n4 5 NA", path = "file.fwf")

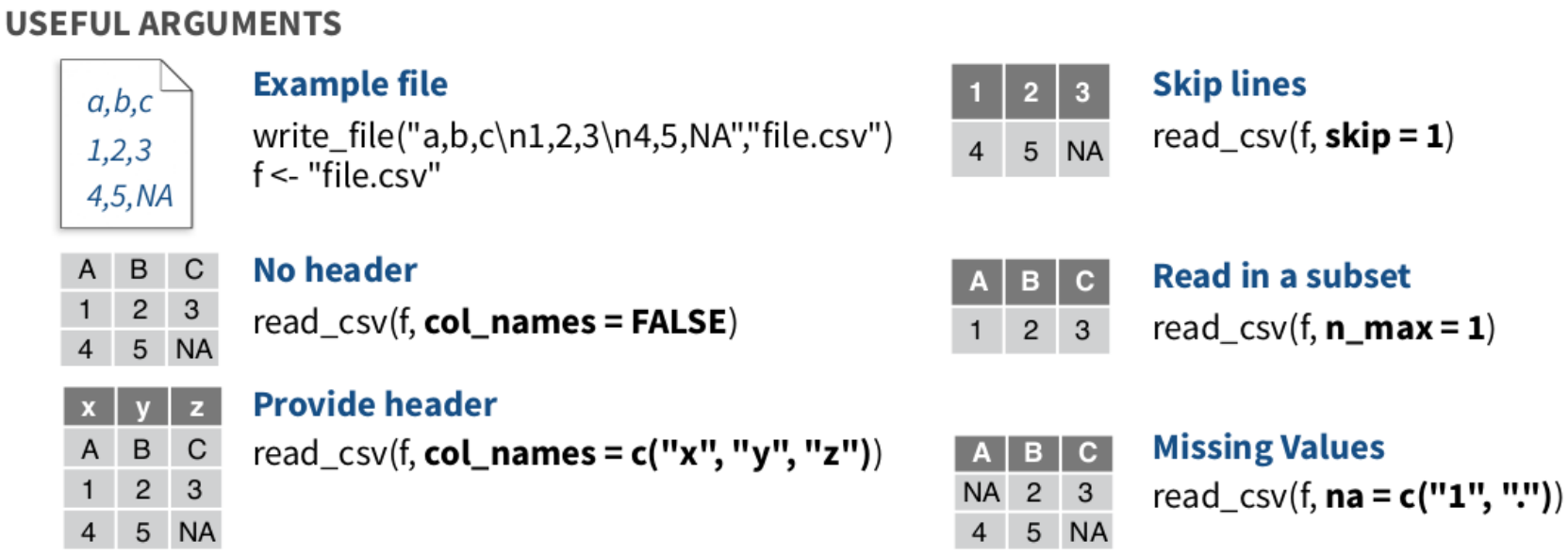
Tab Delimited Files

read_tsv("file.tsv"**)** Also **read_table().**

write_file(x = "a\tb\tc\n1\t2\t3\n4\t5\tNA", path = "file.tsv")

Banpará

Algumas opções úteis



https://github.com/rstudio/cheatsheets/raw/master/data-import.pdf

Lendo Arquivos xlsx

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

file: o caminho para o arquivo

sheetIndex: O número da folha da planilha

header: Um valor lógico. Se verdadeiro, a primeira linha é o nome das variáveis.

colClasses: um vetor de caracteres representando a classe de cada coluna

```
library(xlsx)
file <- system.file("tests", "test_import.xlsx", package = "xlsx")
res <- read.xlsx(file, 1)  # read first sheet
head(res[, 1:6])
```

Banpará

APIs

- O acesso a APIs em R, pode ser feito com o pacote **httr** - `http://hadley.nz/`;
- este pacote atua criando chamadas de APIs e lidando com autenticação destas;
- O pacote **jsonlite**, realiza suporte ao trabalho com dados JASON, para traduzir as estruturas de dados aninhadas do JSON em objetos R;
- O pacote **lubridate**, atua na transformação e extração de datas, funções úteis para trabalhar com datas, fusos horá¡rios e operações aritméticas com datas;

Para obtê-los:
install.packages(c("httr", "jsonlite", "lubridate"))

Banpará

Utilizando APIs

Argumentos de consulta:

- GET(): Recupera o arquivo;
- POST(): Adiciona um arquivo;
- DELETE(): Remove um arquivo;

1. As API's acessam dados na internet. Detalhamos os elementos do comando abaixo:
   1. Verbo HTTP (GET, POST, DELETE, etc.);
   2. O URL base da API;
   3. O caminho da URL ou o endpoint;
   4. Argumentos de consulta de URL (por exemplo, ?foo=bar);
   5. Cabeçalhos opcionais;
   6. Um corpo de solicitação opcional;
2. **download.file()** baixa um arquivo para o seu computador;
3. Se os dados estiverem em **JSON** é necessário utilizar o pacote *jasolite*.
4. a função **fromJSON()** pode ser usada para importar esses dados para um objeto data.frame.

Agradecimentos

Obrigado!

Banpará

Exercícios

Banpará