Apoema

Aquele que *enxerga longe!*

www.**apoema**cursos.com

Data Science

Módulo 2 – Linguagem R

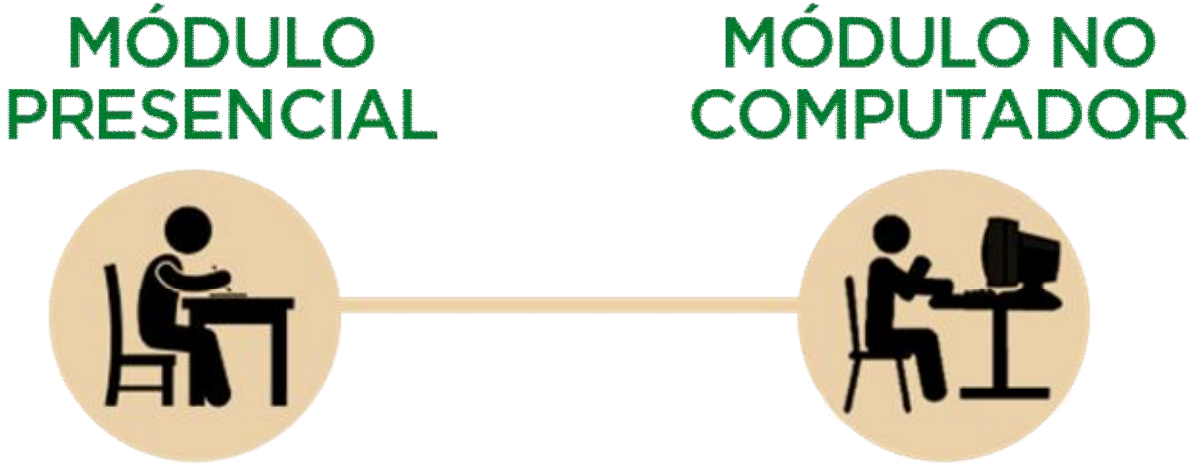
Ed. Enseada Trade Center
Rua Professor Almeida Cousin, 125, Sala 1003, 10º andar
Enseada do Suá, Vitória – ES, CEP: 29050-565

Aula presencial

Aula virtual

Universidade.thescientist.com.br

Atividades



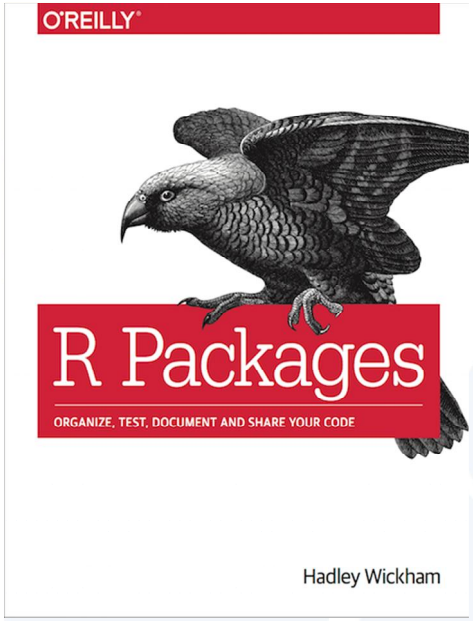
Statistical and Data Analysis

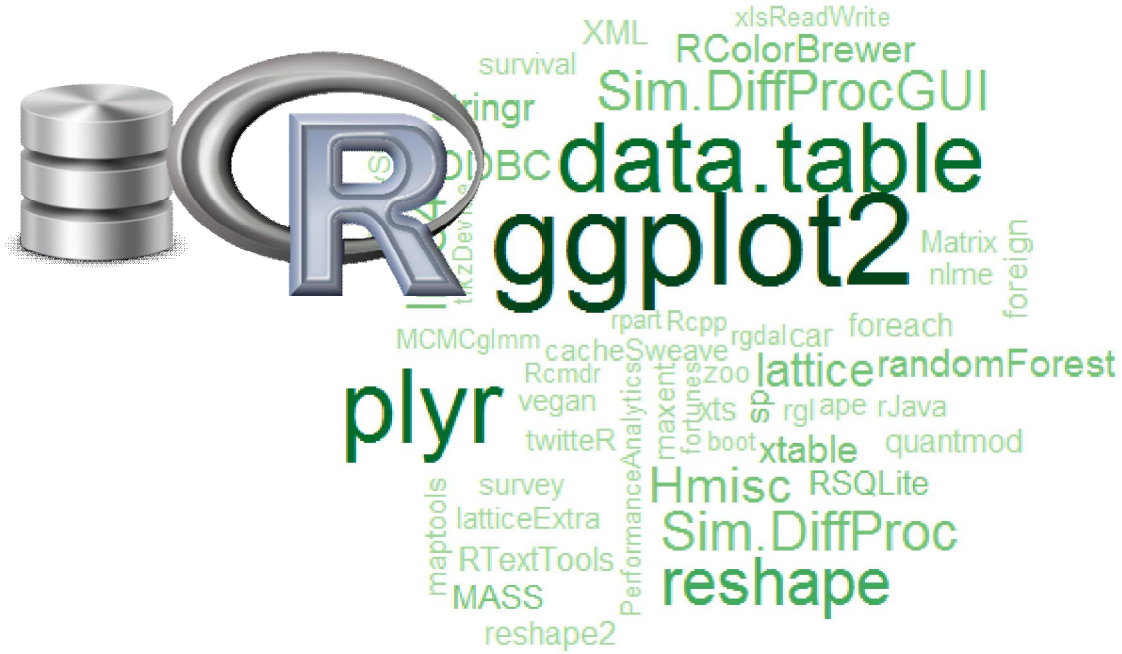
R & RStudio

A GUIDE TO INSTALLING R FOR WINDOWS, LINUX AND MAC OS X

Apoema

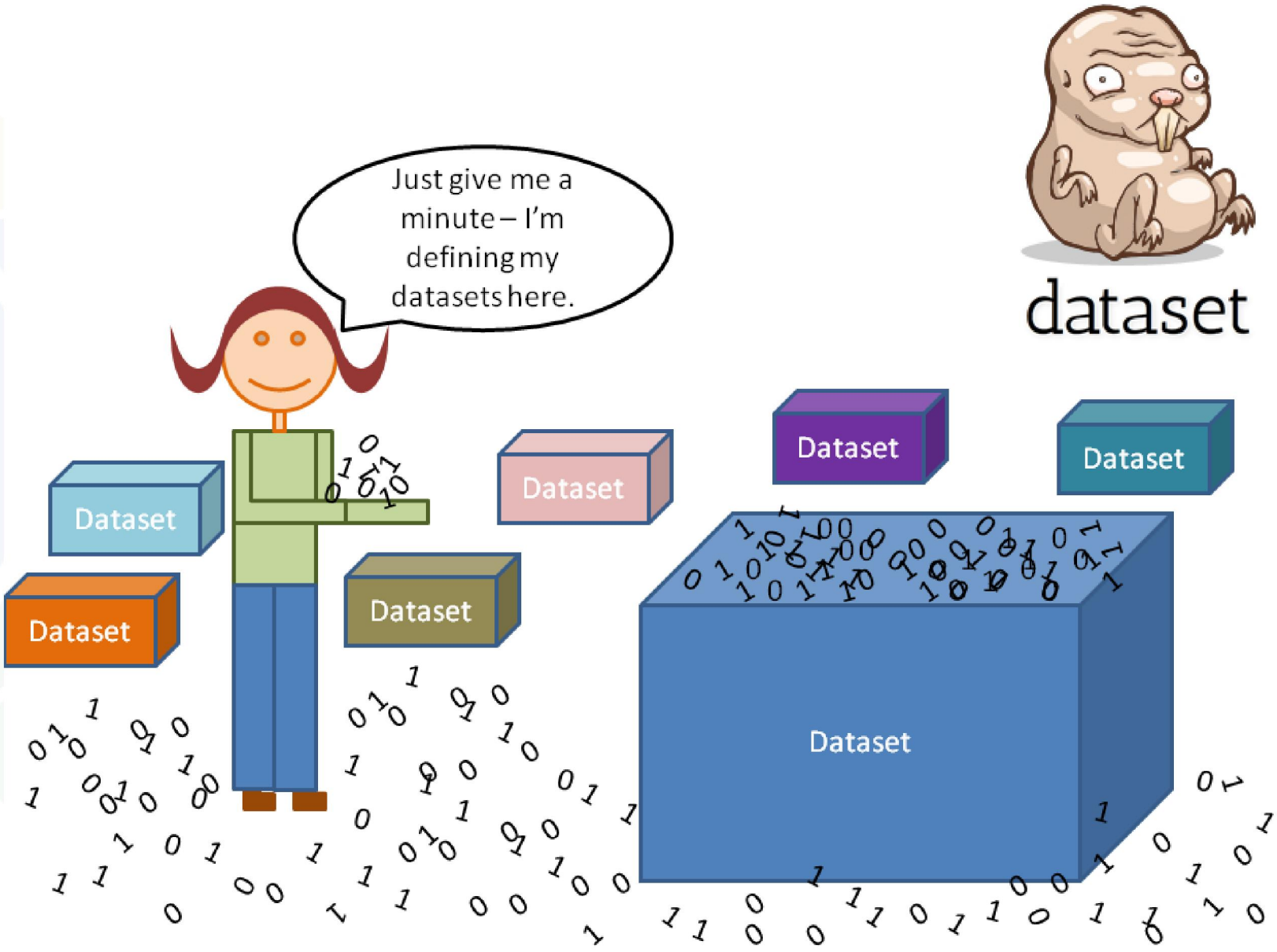
Apoema



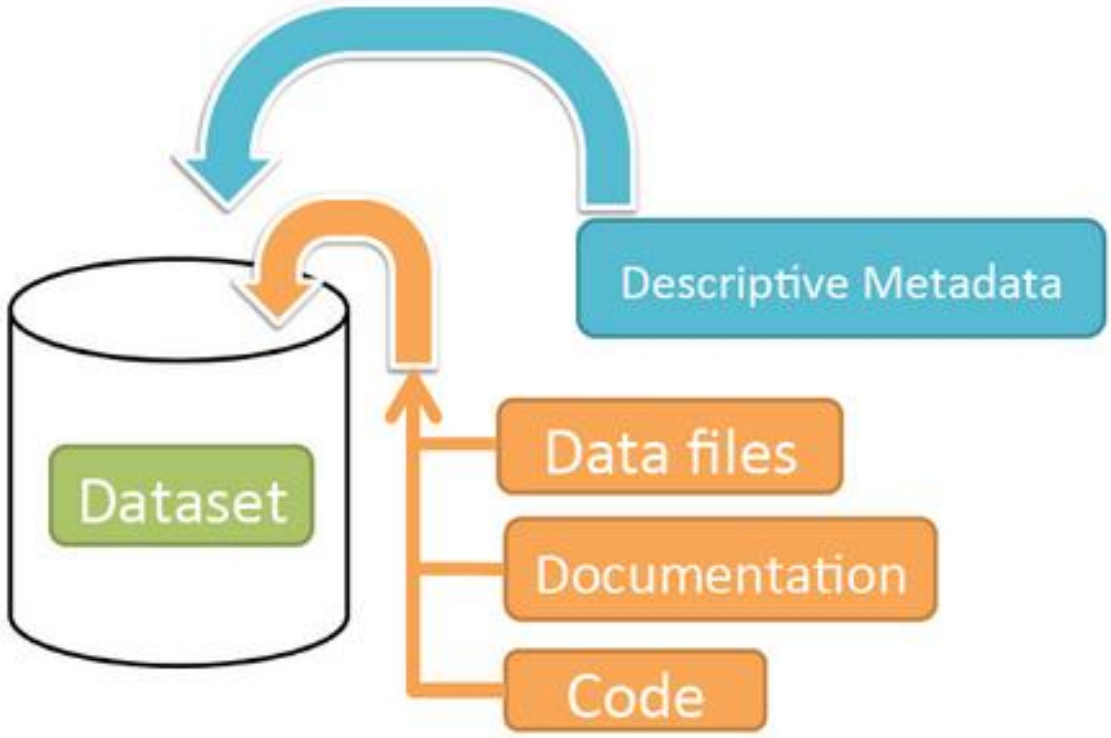



```
a <- available.packages()
head(rownames(a), 3)       ## Show the names of the first few packages
```

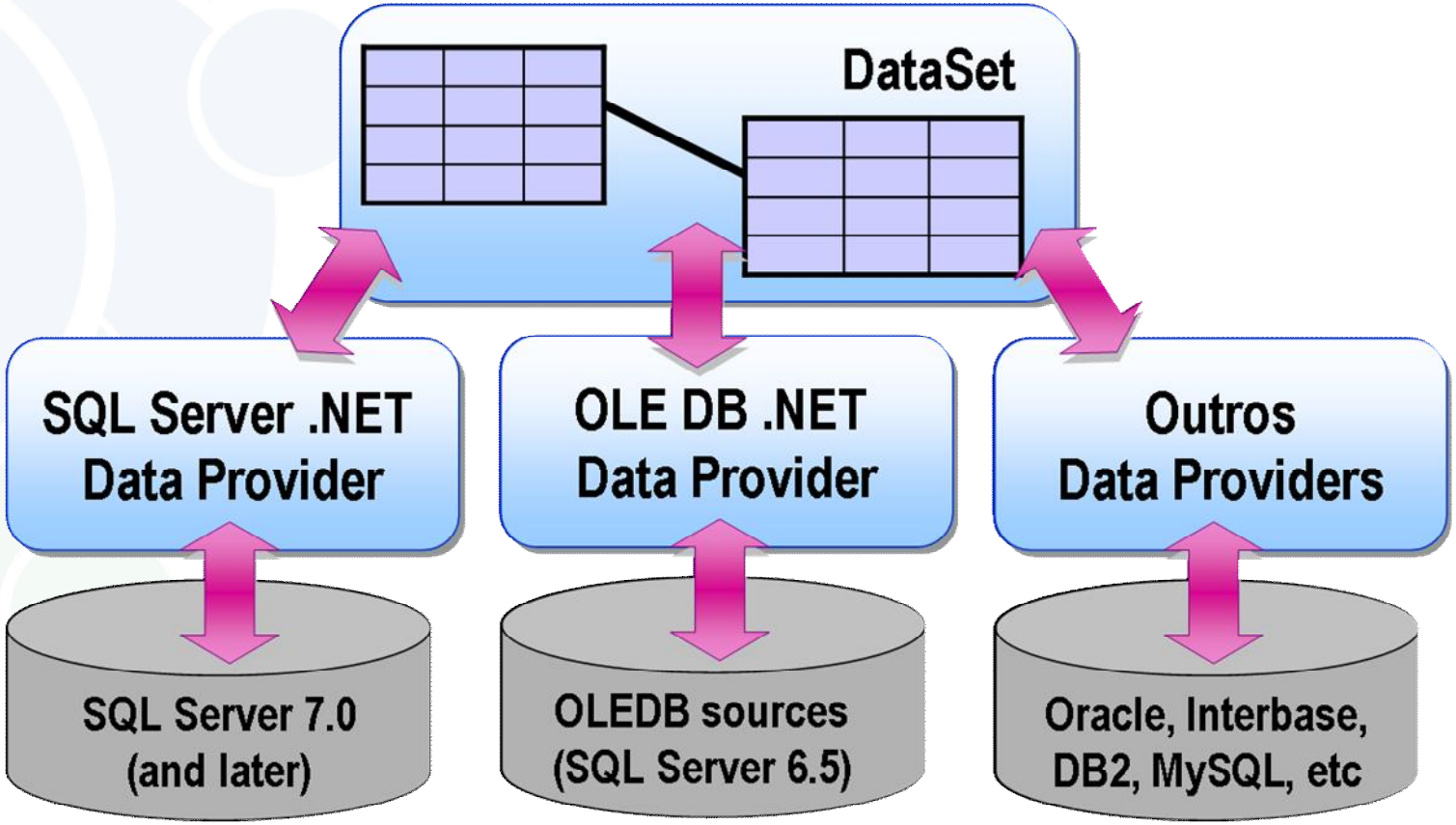
```
## [1] "A3" "abc" "abcdeFBA"
```

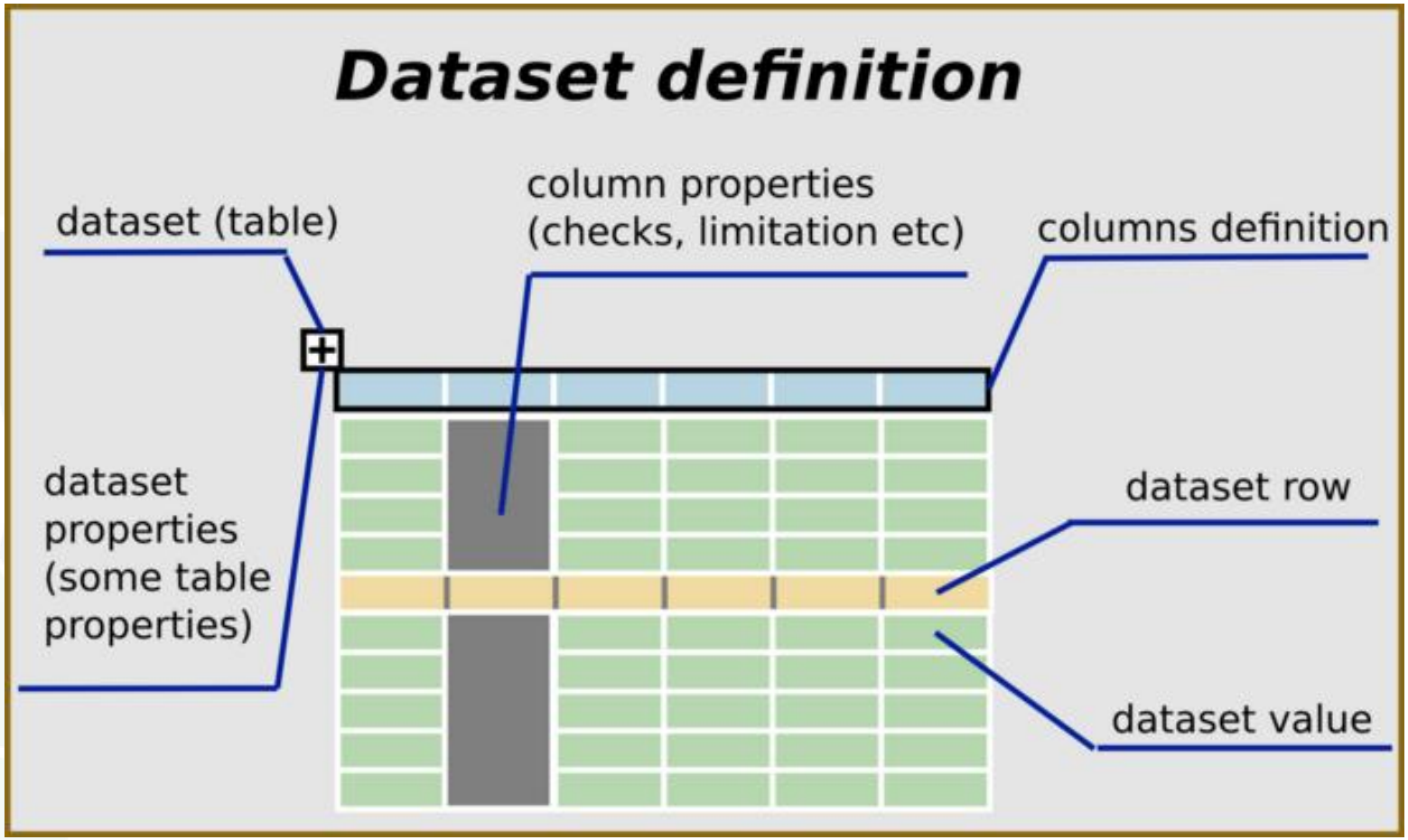

Just give me a minute – I'm defining my datasets here.
Dataset
Dataset
Dataset
Dataset
Dataset
Dataset
Dataset
dataset

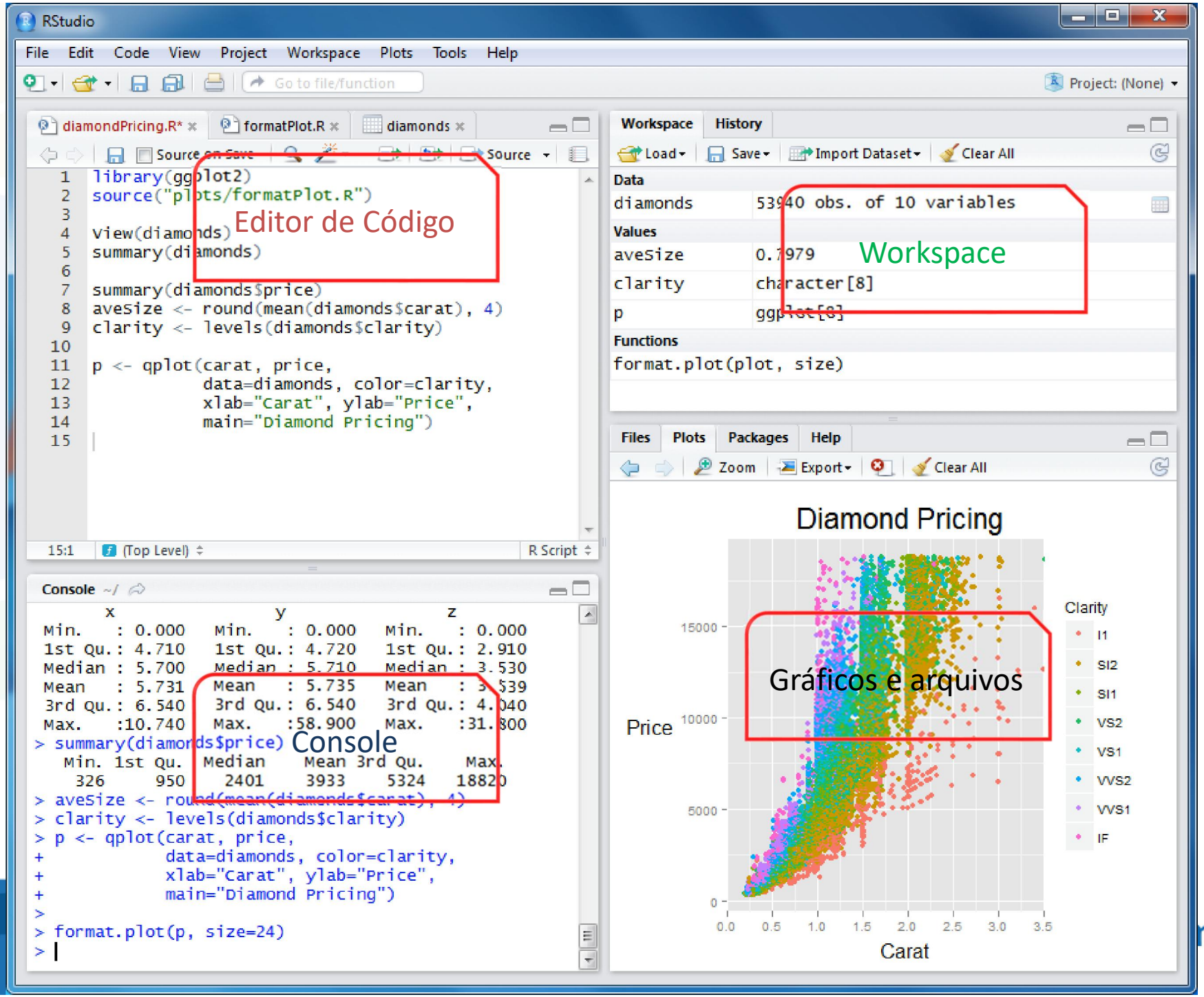
Schematic Diagram of a **Dataset**



Container for your data, documentation, and code.


DataSet
SQL Server .NET
Data Provider
OLE DB .NET
Data Provider
Outros
Data Providers
SQL Server 7.0
(and later)
OLEDB sources
(SQL Server 6.5)
Oracle, Interbase,
DB2, MySQL, etc


Dataset definition
dataset (table)
column properties
(checks, limitation etc)
columns definition
dataset
properties
(some table
properties)
dataset row
dataset value


RStudio
File Edit Code View Project Workspace Plots Tools Help
Go to file/function
Project: (None)
diamondPricing.R*
formatPlot.R
diamonds
Source on Save
Source
1 library(ggplot2)
2 source("plots/formatPlot.R")
3
4 View(diamonds)
5 summary(diamonds)
6
7 summary(diamonds$price)
8 aveSize <- round(mean(diamonds$carat), 4)
9 clarity <- levels(diamonds$clarity)
10
11 p <- qplot(carat, price,
12 data=diamonds, color=clarity,
13 xlab="Carat", ylab="Price",
14 main="Diamond Pricing")
15
Editor de Código
15:1 (Top Level) R Script
Workspace History
Load Save Import Dataset Clear All
Data
diamonds 53940 obs. of 10 variables
Values
aveSize 0.7979
clarity character[8]
p ggplot[8]
Functions
format.plot(plot, size)
Workspace
Files Plots Packages Help
Zoom Export Clear All
Diamond Pricing
Price
Carat
Clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF
Gráficos e arquivos
Console ~/
x y z
Min. : 0.000 Min. : 0.000 Min. : 0.000
1st Qu.: 4.710 1st Qu.: 4.720 1st Qu.: 2.910
Median : 5.700 Median : 5.710 Median : 3.530
Mean : 5.731 Mean : 5.735 Mean : 3.539
3rd Qu.: 6.540 3rd Qu.: 6.540 3rd Qu.: 4.040
Max. :10.740 Max. :58.900 Max. :31.800
> summary(diamonds$price)
Min. 1st Qu. Median Mean 3rd Qu. Max.
326 950 2401 3933 5324 18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+ data=diamonds, color=clarity,
+ xlab="Carat", ylab="Price",
+ main="Diamond Pricing")
>
> format.plot(p, size=24)
>
Console

Objetos em R

- characteres
- númerico (numeros reais)
- inteiros
- complexos
- logicos (True/False)

Objetos em R

O objeto mais básico é um vetor

- Um vetor só pode conter objetos da mesma classe
- MAS: A única exceção é uma * lista *, que é representada como um vetor
- Mas pode conter objetos de diferentes classes (na verdade, isso é geralmente porque os usamos)
- Vetores vazios podem ser criados com a função vector ().

Números

- Os números em R a geralmente tratados como objetos numéricos (ou seja, Números reais de precisão)
- Se você quiser explicitamente um inteiro, você precisa especificar o `` `L`` como sufixo
  - Ex: Introduzir `` `1``` dá-lhe um objecto numérico; Entrando `` `1L`` Explicitamente dá-lhe um número inteiro.

- Existe também um número especial `` `Inf``` que representa o infinito;
  - por exemplo. `` `1 / 0```; `` `Inf``` pode ser usado em cálculos comuns;
  - por exemplo. `` `1 / Inf``` é 0

- O valor `` `NaN``` representa um valor indefinido ("not a number");
  - por exemplo. 0/0; `` `NaN``` também pode ser pensado como um valor em falta (Mais sobre isso mais tarde)

Atributos em R

- Objetos em R podem ter atributos, como exemplo:
- nomes, dimnames
- dimensões (por exemplo, matrizes, matrizes)
- classe
- comprimento
- outros atributos / metadados definidos pelo usuário

- Os atributos de um objeto podem ser acessados usando `attributes ()`
- função.
  - pode ser pensado como um valor em falta (Mais sobre isso mais tarde)

Criando vetores

A função `c ()` pode ser usada para criar vetores de objetos.

```
X <- c (0.5, 0.6) ## numerico
X <- c (TRUE, FALSE) ## lógico
X <- c (T, F) ## lógico
X <- c ("a", "b", "c") ## caractere
X <- 9:29 ## inteiro
X <- c (1 + 0i, 2 + 4i) ## complexo
```

Usando a função vector()

```
x <- vector("numeric", length = 10)


print(x)
 [1] 0 0 0 0 0 0 0 0 0 0
```

Misturando objetos

O que acontece depois?

y <- c(1.7, "a") ## caractere
y <- c(TRUE, 2) ## numérico
y <- c("a", TRUE) ## caractere

```
y <- c(1.7, "a") ## caractere
y
[1] "1.7" "a"
```

Coerção explicita

Os objetos podem ser explicitamente coagidos de uma classe para outra usando as funções `as. *`, Se disponível.

```
X <- 0: 6
Classe (x)
[1] "integer"

As.numeric (x)
[1] 0 1 2 3 4 5 6

As.logical (x)
[1] FALSO VERDADEIRO VERDADEIRO VERDADEIRO VERDADEIRO VERDADEIRO

As.character (x)
[1] "0" "1" "2" "3" "4" "5" "6"
```

Coerção explicita

A coerção absurda ou sem sentido (Nonsensical) resulta em `NA`s.

X <- c ("a", "b", "c")
As.numeric (x)
[1] NA NA NA

Mensagem de aviso:
NAs introduzidas pela coerção

As.logical (x)
[1] NA NA NA
As.complex (x)
[1] NA NA NA

Mensagem de aviso:
NAs introduzidas pela coerção[1] "0" "1" "2" "3" "4" "5" "6"

Matrizes

As matrizes são vetores com um atributo _dimension_. O atributo de dimensão é em si um vetor inteiro de comprimento 2 (nrow, ncol)

```
m <- matriz (nrow = 2, ncol = 3)
m
[,1] [,2] [,3]
[1,] NA NA NA
[2,] NA NA NA

 Dim (m)
[1] 2 3
attributes(m)
$dim
[1] 2 3
```

Matrizes

As matrizes são construídas _column-wise_, então as entradas podem ser pensadas de começar no canto "superior esquerdo" e correr pelas colunas.

```
m <- matriz (1: 6, nrow = 2, ncol = 3)
m
     [, 1] [, 2] [, 3]
[1,] 1      3     5
[2,] 2      4     6
```

Matrizes

Matrizes também podem ser criadas diretamente a partir de vetores adicionando um atributo de dimensão.

```
m <- 1:10
m
[1] 1 2 3 4 5 6 7 8 9 10

Dim (m) <- c (2, 5)
m
     [, 1] [, 2] [, 3] [, 4] [, 5]
[1,]   1    3     5     7    9
[2,]   2    4     6     8   10
```

cbind-ing e rbind-ing

Matrizes podem ser criadas por _column-binding_ ou _row-binding_ com `cbind ()` e `rbind ()`.

```
X <- 1: 3
Y <- 10:12
cbind (x, y)
      x   y
[1,] 1 10
[2,] 2 11
[3,] 3 12

Rbind (x, y)
   [, 1] [, 2] [, 3]
X    1    2      3
Y  10  11     12
```

Listas

As listas são um tipo especial de vetor que pode conter elementos de classes diferentes. Um tipo de dados muito importante em R e você deve conhecê-los bem.

```
X <- list (1, "a", TRUE, 1 + 4i)
X
[[1]]
[1] 1

[[2]]
[1] "a"

[[3]]
[1] VERDADEIRO

[[4]]
[1] 1 + 4i
```

Factors

Os fatores são usados para representar dados categóricos.
Os fatores podem ser desordenados ou ordenados. Pode-se pensar em um fator como um vetor inteiro onde cada inteiro tem um _label_.

Fatores são tratados especialmente por funções de modelagem como `lm ()` e `glm ()`

Usar fatores com rótulos é melhor do que usar números inteiros porque os fatores são auto-descreventes ou seja: **Ter uma variável que tem valores "Masculino" e "Feminino" é melhor do que uma variável que tem valores 1 e 2.**

Factors

```
x <- factor(c("yes", "yes", "no", "yes", "no"))
x
[1] yes yes no yes no
Levels: no yes

table(x)
x
no yes
 2   3

unclass(x)
[1] 2 2 1 2 1

attr(,"levels")
[1] "no"  "yes"
```

Factors

A ordem dos níveis pode ser definida usando o argumento `levels` para` factor () `. Isso pode ser importante na modelagem linear porque o primeiro nível é usado como o nível de linha de base.

```
x <- factor(c("yes", "yes", "no", "yes", "no"), levels = c("yes", "no"))
x
[1] yes yes no yes no
Levels: yes no
```

Missing Values

Os valores faltantes (Missing Values) são denotados por `NA` ou` NaN` para operações matemáticas indefinidas.

- `is.na ()` é usado para testar objetos se forem `NA` (not available)

- `is.nan ()` é usado para testar `NaN` (Not a Number)

- Os valores de `NA` têm uma classe também, portanto, há um inteiro` NA`, caractere `NA`, etc.

- Um valor `NaN` também é` NA`, mas o inverso não é verdadeiro

```
?is.nan
?is.na
?NA
?NaN
```

Tente isso!

Data Frames

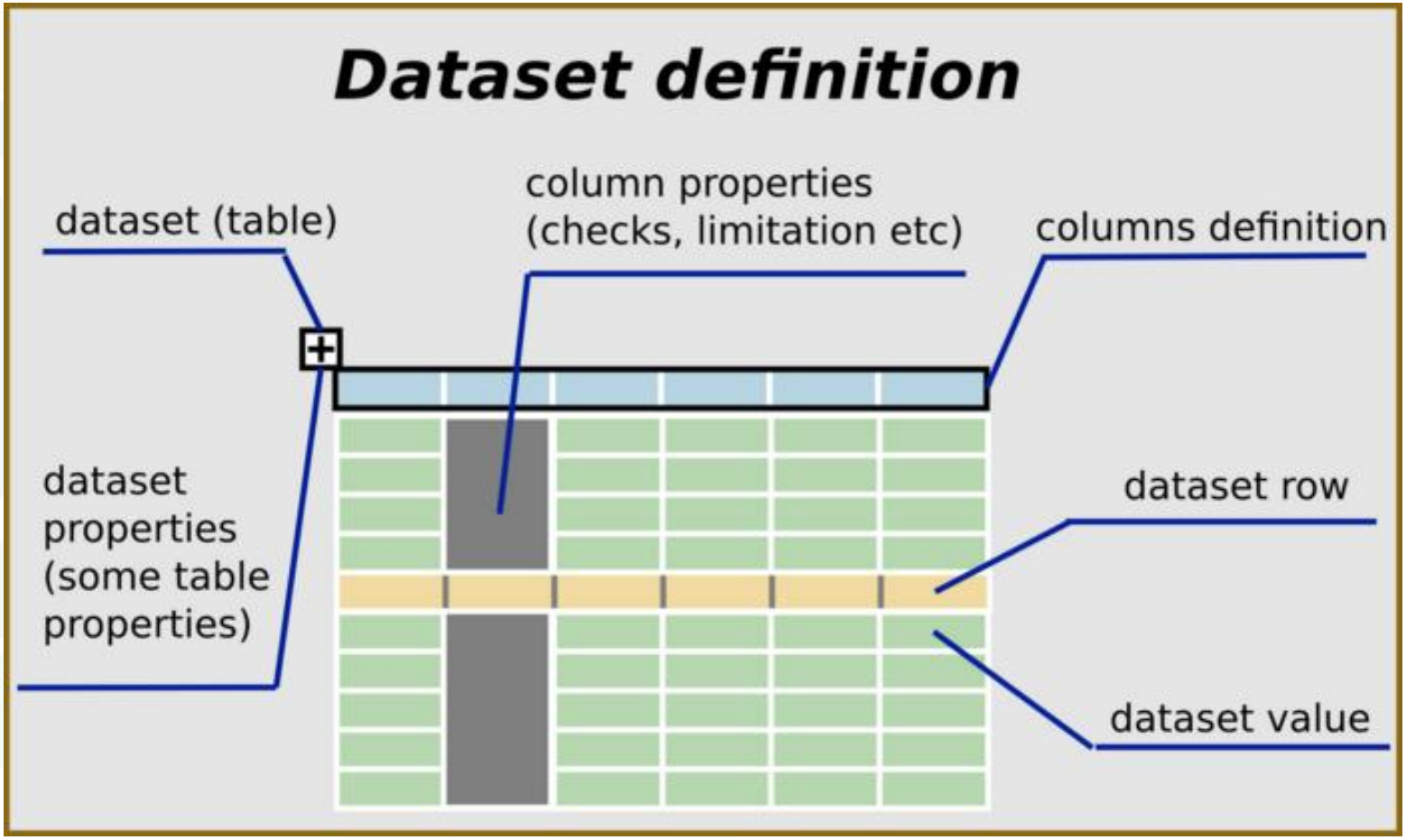
Os Data Frames são usados para armazenar dados tabulares

Eles são representados como um tipo especial de lista onde cada elemento da lista deve ter o mesmo comprimento.

Cada elemento da lista pode ser pensado como uma coluna e o comprimento de cada elemento da lista é o número de linhas.

Ao contrário das matrizes, os Data Frames podem armazenar diferentes classes de objetos em cada coluna (como listas).

- Matrizes devem ter todos os elementos da mesma classe
- Os Data Frames também têm um atributo especial chamado `row.names`
- Os Data Frames são normalmente criados chamando `read.table ()` ou `read.csv ()`
- Pode ser convertido em uma matriz chamando `data.matrix ()`
- Um valor `NaN` também é` NA`, mas o inverso não é verdadeiro


Dataset definition
dataset (table)
column properties
(checks, limitation etc)
columns definition
dataset
properties
(some table
properties)
dataset row
dataset value

Data Frame

```
x <- data.frame(foo = 1:4, bar = c(T, T, F, F))

x
  foo   bar
1   1  TRUE
2   2  TRUE
3   3 FALSE
4   4 FALSE

nrow(x)
[1] 4

ncol(x)
[1] 2
```

Names

Os objetos R também podem ter nomes, o que é muito útil para escrever códigos legíveis e objetos auto-descreventes.

```
x <- 1:3
names(x)
NULL

names(x) <- c("foo", "bar", "norf")
x
foo bar norf
  1   2    3

names(x)
[1] "foo"  "bar"  "norf"
```

Names

As listas também podem ter nomes

```
x <- list(a = 1, b = 2, c = 3)
x
$a
[1] 1

$b
[1] 2

$c
[1] 3
```

Names

E matrices

```
m <- matrix(1:4, nrow = 2, ncol = 2)
dimnames(m) <- list(c("a", "b"), c("c", "d"))

m
  c d
a 1 3
b 2 4
```