LAMFO

Introdução ao R

Cayan Portela
cayanportela@hotmail.com

LAMFO

Laboratório de Aprendizado de Máquina em Finanças e Organizações

www.lamfo.unb.br

www.lamfo-unb.github.io

www.facebook.com/lamfounb/

Coordenador: Prof. Pedro Alburquerque

Escopo

- Rstudio IDE - Ferramentas básicas.
- O R como uma calculadora.
- Estruturas de Objetos.
- Vetores, Matrizes, Data Frames e Listas.
- Importação e Exportação de Dados.
- Operações básicas em uma tabela de dados.
- Funções, for (loop) e família apply.
- Gráficos

Escopo

- Rstudio IDE - Ferramentas básicas.
- O R como uma calculadora.
- Estruturas de Objetos.
- Vetores, Matrizes, Data Frames e Listas.
- Importação e Exportação de Dados.
- Operações básicas em uma tabela de dados.
- ~~Funções, for (loop) e família apply.~~
- Gráficos

Conceitos Básicos

- O que é o R?

1. Software livre.
2. Ambiente de programação estatística.

- Alguns pontos positivos.

1. Análise gráfica flexível e de qualidade.
2. Fronteira do conhecimento: métodos estatísticos mais recentes.
3. Comunidade aberta: foruns, stackoverflow, etc.

RStudio

- IDE (Ambiente de Desenvolvimento Integrado) para o software R.
- Funcionalidades e melhor organização para progrmação.

Divido em 4 partes (alteráveis):

1. Editor
2. Console
3. Help, Plots, Pacotes, etc.
4. Ambiente, histórico, conexões.

Comandos Básicos

Console: onde a programação do R “é exectuada”.

Símbolo “>” indica que o R está pronto para receber um comando.

Para executar um comando:

1. Na janela de console: Enter.
2. Na janela de editor:
    - ctrl + Enter (Linux)
    - ctrl + Enter ou ctrl + R (Windows)
3. Point-click: “Run”

Pacotes

- Pacotes são conjunto de funções externas.
- Havendo conexão com a internet, pacotes são facilmentes instalados.

  Geralmente, são instalados de duas maneiras:

1. Tools > Install Packages... > Nome_do_Pacote
2. Através do comando 'install.packages("Nome_do_Pacote")'

```
install.packages("dplyr")
library(dplyr)      # Carregando o pacote no ambiente.
?mutate             # Com '?' abrimos a documentação do pacote, para ajuda.
```

Programação Inicial

Inicialmente podemos pensar no R como uma calculadora.

```
2+2
[1] 4
```

Podemos indexar valores a um objeto. Assim, ele estará disponível no seu ambiente, representando seu valor.

```
imposto <- 0.12
salario <- 800
imposto*salario
[1] 96
```

Programação Inicial

Tambem podemos alocar strings (letras) em um objeto.

```
x <- "Ola R!"
print(x)
[1] "Ola R!"
```

É comum a necessidade de alocar um conjunto de valores em um único objeto. Para isso, eles são armazenados em um vetor, através da função 'c()'.

```
rendimentos <- c(354,400,877,230,820)
rendimentos
[1] 354 400 877 230 820
```

Programação Inicial

O R também trabalha de maneira vetorial. Por exemplo, caso queiramos multiplicar todos os valores do vetor 'rendimentos' por 5%, basta multiplicar o proprio vetor por 0.05. Não há necessidade de realizar uma operação para cada elemento

```
rendimentos*0.05
[1] 17.70 20.00 43.85 11.50 41.00
```

Programação Inicial

Se os vetores forem do mesmo tamanho, a operação é realizada elemento a elemento. Se forem de tamanhos diferentes, o R dá um aviso e recicla elementos.

```
c( 1 , 2 , 3 ) + c( 1 , 2 , 3 )
[1] 2 4 6
```

```
c( 1 , 2 , 3 ) + c( 1 , 2 , 3 , 4 )
Warning in c(1, 2, 3) + c(1, 2, 3, 4): comprimento do objeto maior não é
múltiplo do comprimento do objeto menor
[1] 2 4 6 5
```

Programação Inicial

Vetores também podem ser compostos por strings. Como anteriormente, utilizaremos a funçao “c()”, porém os valores devem ser inseridos entre aspas.

```
alunos <- c("ana","bia","carol","joao")
alunos
[1] "ana"   "bia"   "carol" "joao"
```

Operadores Lógicos

Operadores logicos podem assumir dois valores: TRUE (verdadeiro) ou FALSE (falso). Esse tipo lógico é resultante de operações com símbolos de comparação.

```
10 > 5
[1] TRUE
10 < 5
[1] FALSE
```

```
10 == 5
[1] FALSE
10 == (5*2)
[1] TRUE
```

Operadores Lógicos

Como anteriormente, podemos fazer comparação vetorialmente.

```
rendimentos > 500
[1] FALSE FALSE  TRUE FALSE  TRUE
```

Valores logicos, como os resultantes de operações como esta, podem ser utilizados em operadores matemáticos. Nestas operações, valores TRUE são computados como valores iguais a 1 e valores FALSE como 0.

```
sum(rendimentos > 500)
[1] 2
mean(rendimentos > 500)
[1] 0.4
```

Indexação de Objetos

Para acessar um ou mais valores específicos de um elemento, utiliza-se colchetes '[ ]' após o nome do vetor, com a posição do elemento desejado.

```
rendimentos
[1] 354 400 877 230 820
```

```
rendimentos[1]
[1] 354
```

```
rendimentos[c(1,2)]
[1] 354 400
```

```
rendimentos[-c(1,3,5)]
[1] 400 230
rendimentos[c(TRUE,FALSE,TRUE,FALSE,TRUE,FALSE)]
[1] 354 877 820
```

Ordenando Vetores

Para ordenar um vetor, pode ser oneroso escrever a posicao dos elementos que represente a ordenação desejada. Ao invés disso, usaremos a função 'order()' ou 'sort()'.

```
rendimentos
[1] 354 400 877 230 820
```

```
order(rendimentos)
[1] 4 1 2 5 3
```

```
rendimentos[order(rendimentos)]
[1] 230 354 400 820 877
```

```
sort(rendimentos)
[1] 230 354 400 820 877
```

Matrizes

Extendendo o conceito de vetores, vamos considerar matrizes.

```
matriz <- matrix( c(1,3,5,7,11,13,17,19,23) , byrow = TRUE , ncol = 3)
matriz
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    7   11   13
[3,]   17   19   23
```

Agora, na indexação teremos duas posições: linhas e colunas. O primeiro (antes da virgula), corresponde à linha e o segundo (após a vírgula) à coluna.

```
matriz[2,3]
[1] 13
```

Matrizes

Caso o interesse seja em obter todos os elementos de uma respectiva linha ou coluna, basta deixar o espaço referente sem preenchimento.

```
# Todas as linhas da coluna 1
matriz[ , 1]
[1]  1  7 17
```

```
# Todas as colunas da linha 3
matriz[3 , ]
[1] 17 19 23
```

```
# Linhas 1 e 3 ; todas as colunas
matriz[c(1,3) , ]
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]   17   19   23
```

Matrizes e Vetores

Uma característica de matrizes e vetores, é que todos os elementos serão do mesmo tipo. Caso tente juntar elementos diferentes em um vetor, como visto anteriormente, o R converterá os elementos para que todos sejam do mesmo tipo (no caso, string).

- Matrizes: todos os elementos numéricos.
- Vetores: todos os elementos numéricos ou todos os elementos strings.

Em diversas aplicações é necessário reunir elementos de diferentes tipos em uma mesma tabela. No R, essas tabelas se chamam data frames.

Data Frames

Para exemplificar o uso de um data.frame vamos considerar as notas e outras informações de um conjunto de alunos.

```
nomes    <- c("ana", "bia", "caio", "dani", "edu")
nota1    <- c( 7,  4, 9, 3, 6)
nota2    <- c( 10, 5, 6, 2, 4)
entrega1 <- c(  T, F, T, F, F)
planilha <- data.frame(nomes , nota1 , nota2 , entrega1)
planilha
  nomes nota1 nota2 entrega1
1   ana     7    10     TRUE
2   bia     4     5    FALSE
3  caio     9     6     TRUE
4  dani     3     2    FALSE
5   edu     6     4    FALSE
```

Data Frames

Podemos ter colunas com variáveis de diferentes tipos, porém todos os elementos de cada coluna precisam ser do mesmo tipo, e as colunas precisam ter o mesmo tamanho (valores faltantes podem ser preenchidos com NA). O indexador funciona da mesma maneira que vimos para matrizes, mas também de outras formas.

```
planilha[3,1]
[1] caio
Levels: ana bia caio dani edu
planilha[2,3]
[1] 5
```

Data Frames

```
planilha$nota2
[1] 10  5  6  2  4

planilha[,'nota2']
[1] 10  5  6  2  4

## Criando nova variável

planilha$Presenca <- c(0.95,0.80,0.95,0.7,0.75)

planilha
  nomes nota1 nota2 entrega1 Presenca
1   ana     7    10     TRUE     0.95
2   bia     4     5    FALSE     0.80
3  caio     9     6     TRUE     0.95
4  dani     3     2    FALSE     0.70
5   edu     6     4    FALSE     0.75
```

LAMFO

Data Frames

O ordenamento também funciona da mesma forma. Comumente, é interessante ordenar o data frame por alguma medida crescente ou decrescente. Por exemplo, podemos ordenar o data frame de acordo com a nota1.

```
order(planilha$nota1)
[1] 4 2 5 1 3

planilha[ order(planilha$nota1) , ]
  nomes nota1 nota2 entrega1 Presenca
4  dani     3     2    FALSE     0.70
2   bia     4     5    FALSE     0.80
5   edu     6     4    FALSE     0.75
1   ana     7    10     TRUE     0.95
3  caio     9     6     TRUE     0.95
```

Data Frames

Para filtrar valores, podemos utilizar os operadores lógicos, segundo algum critério específico. Por exemplo, caso queiramos uma tabela onde constam apenas alunos com nota maior que 5 na primeira prova.

```
planilha$nota1 > 5
[1]  TRUE FALSE  TRUE FALSE  TRUE

planilha2 <- planilha[ planilha$nota1 > 5 , ]

planilha2
  nomes nota1 nota2 entrega1 Presenca
1   ana     7    10     TRUE     0.95
3  caio     9     6     TRUE     0.95
5   edu     6     4    FALSE     0.75
```

Data Frames

Para a mesma operação realizada anteriormente, pode-se utilizar a função 'subset()'.

```
planilha2b <- subset( planilha , nota1 > 5)

planilha2b
  nomes nota1 nota2 entrega1 Presenca
1   ana     7    10     TRUE     0.95
3  caio     9     6     TRUE     0.95
5   edu     6     4    FALSE     0.75
```

Para métricas que envolvam duas condições (e.g. nota1 maior que 5 e nota2 menor que 8) informamos duas condições com utilizando o operador '&'.

subset(planilha , nota1 > 5 & nota2 < 8).

Exportando Dados

Para exportar um objeto criado no R, utilizamos funções com write.table(...) , write.csv(...) , write.csv2(...) , a depender do tipo de separador desejado.

```
getwd()
[1] "/home/cayan"

write.csv(planilha2 , "planilha2.csv", row.names = FALSE)
```

Missing Values

Quando um elemento é desconhecido ou faltante, o mesmo é registrado nos dados como NA (not available). Em geral, operações que que envolvam NA retornam NA.

```
vetor <- c(1,2,3,4,NA)
vetor
[1]  1  2  3  4 NA

sum(vetor)
[1] NA
sum(vetor,na.rm=TRUE)
[1] 10
```

Missing Values

Para trabalhar com operadores lógicos com NA, usamos a função 'is.na()'.

```
vetor == 4
[1] FALSE FALSE FALSE  TRUE    NA

vetor == NA
[1] NA NA NA NA NA

is.na(vetor)
[1] FALSE FALSE FALSE FALSE  TRUE

sum(is.na(vetor))
[1] 1

mean(is.na(vetor))*100
[1] 20
```

Missing Values

Um outro tipo de “valor faltante”, são os resultantes de algumas computações numéricas, os NaN (Not a Number), causados por indeterminações.

Similarmente, usa-se a função ‘is.nan()’ para operadores lógicos.

```
0/0
[1] NaN

Inf - Inf
[1] NaN

is.nan( c(1 , 2 , 3 , 0/0) )
[1] FALSE FALSE FALSE  TRUE
```

Exercícios

Considerando o data frame ‘planilha’ faça:

1. Calcule as médias dos vetores ‘nota1’ e ‘nota2’
2. Crie uma nova variável com a média das duas provas para cada aluno.
3. Crie uma nova variável que recebe “Aprovado” caso a média do aluno seja maior ou igual a 5 e “Reprovado” caso contrário.
4. Ordene a tabela pelo vetor correspondente à média das notas.
5. Crie uma segunda tabela, apenas com alunos aprovados.

Leitura de Dados Externos

Existem diversas maneiras para carregar dados de fontes externas no R.

Muitas vezes a função utilizada é dependente da estrutura dos dados (csv, excel, txt, jSON, SAS, stata, etc). https://raw.githubusercontent.com/Cayan-Portela/cursoR/master/salario.txt

```
dados <- read.table("/home/cayan/salario.txt",
                    header = TRUE, sep = "" , dec = ".")

head(dados)
  Salario      Sexo Posicao  Exp
1     148 Masculino       7 16.7
2     165 Masculino       7  6.7
3     145 Masculino       5 14.8
4     139  Feminino       7 13.9
5     142  Feminino       6  6.4
6     144 Masculino       5  9.1
```

Leitura de Dados Externos

Existem outras funções para carregar dados de outros formatos. Abaixo, seguem alguns exemplos.

- Para dados em formato csv: read.csv(...)
- Para dados em formato excel: read_excel(..) - pacote 'readxl'
- Para dados em formato SAS: read_sas(...) - pacote 'haven'
- Para dados em formato stata: read_stata(...) - pacote 'haven'

Explorando Data Frames

Algumas função são bastante úteis quando estamos analisando uma base de dados real. Com o objeto 'dados' exploraremos algumas destas funções.

```
# Olhando a estrutura das variáveis dos dados

str(dados)
'data.frame':   220 obs. of  4 variables:
 $ Salario: int  148 165 145 139 142 144 128 143 157 150 ...
 $ Sexo   : Factor w/ 2 levels "Feminino","Masculino": 2 2 2 1 1 2 1 2 2 2 ...
 $ Posicao: int  7 7 5 7 6 5 3 6 7 7 ...
 $ Exp    : num  16.7 6.7 14.8 13.9 6.4 9.1 8.5 18.2 13 21.6 ...

# Olhando a dimensão da tabela (linhas x colunas)

dim(dados)
[1] 220   4
```

Explorando Data Frames

Também é interessante calcular algumas medidas descritivas, como média e variância.

```
# Calculando a média da variavel salario
sum( dados$Salario ) / length( dados$Salario )
[1] 142.8682

# Calculando a média da variavel salario
mean(dados$Salario)
[1] 142.8682

# Calculando a variância da variavel salario
sum((dados$Salario - mean(dados$Salario))^2) / (length(dados$Salario) - 1)
[1] 156.7634

# Calculando a variância da variavel salario
var(dados$Salario)
[1] 156.7634
```

Explorando Data Frames

- Função 'summary()' fornece medidas descritivas gerais do data frame.
- Função 'View()' abre os dados em um painel de visualização. (No painel de ambiente, clicar no objeto).

```
summary(dados)
    Salario            Sexo       Posicao          Exp        
 Min.   :110.0   Feminino : 75   Min.   :1.000   Min.   : 1.70  
 1st Qu.:133.0   Masculino:145   1st Qu.:4.000   1st Qu.: 6.60  
 Median :143.5                   Median :5.000   Median : 9.50  
 Mean   :142.9                   Mean   :5.068   Mean   :10.48  
 3rd Qu.:151.0                   3rd Qu.:6.000   3rd Qu.:13.40  
 Max.   :172.0                   Max.   :9.000   Max.   :26.10  

# View(dados)
```

Explorando Data Frames

Duas funções bastante utilizadas no R, são as funções 'cbind()' e 'rbind()', utilizadas para "juntar" data frames por colunas (columns) e linhas (rows) respectivamente. Por exemplo, voltando ao data frame 'planilha', suponha que agora estão disponíveis as mesmas informações para outros alunos, e deseja-se juntar as duas tabelas em uma só.

```
planilha3 <- data.frame(nomes    = c("joao","manu","tais"),
                        nota1    = c( 2 , 9 , 4),
                        nota2    = c( 5 , 5 , 6),
                        entrega1 = c( T , T , T),
                        Presenca = c( 0.85 , 0.95 , 0.90) )
```

Explorando Data Frames

```
head(planilha)
  nomes nota1 nota2 entrega1 Presenca
1   ana     7    10     TRUE     0.95
2   bia     4     5    FALSE     0.80
3  caio     9     6     TRUE     0.95
4  dani     3     2    FALSE     0.70
5   edu     6     4    FALSE     0.75

head(planilha3)
  nomes nota1 nota2 entrega1 Presenca
1  joao     2     5     TRUE     0.85
2  manu     9     5     TRUE     0.95
3  tais     4     6     TRUE     0.90
```

Explorando Data Frames

```
planilha4 <- rbind(planilha , planilha3)

planilha4
  nomes nota1 nota2 entrega1 Presenca
1   ana     7    10     TRUE     0.95
2   bia     4     5    FALSE     0.80
3  caio     9     6     TRUE     0.95
4  dani     3     2    FALSE     0.70
5   edu     6     4    FALSE     0.75
6  joao     2     5     TRUE     0.85
7  manu     9     5     TRUE     0.95
8  tais     4     6     TRUE     0.90
```

Caso a outra tabela fosse referente à outras variáveis dos mesmos alunos, o procedimento seria análogo, utilizando o 'cbind()'. O resultado seria um novo data frame com o mesmo número de linhas e com mais colunas.

Exercícios

Considere os seguintes data frames:

```
d1 <- data.frame( Peso   = runif( n = 50 , min = 40   , max = 110),
                  Altura = runif( n = 50 , min = 1.5 , max = 2))

d2 <- data.frame( Altura = runif( n = 50 , min = 1.5 , max = 2),
                  Peso   = runif( n = 50 , min = 40   , max = 110))
```

1. Crie a variável IMC nos dois data frames (IMC = Peso / Altura^2)
2. Crie um novo data frame correspondente aos dois data frames (100 linhas , 3 colunas)
3. Neste novo data frame (100x3), exclua observações com IMC menores que 15 e maiores que 40

Listas

Listas possuem uma estrutura poderosa e são bastantes utilizadas no R. Nas listas, é possível armazenar um conjunto de objetos de diferentes estruturas (vetores, strings, funções, data.frames, matrizes, etc.), permitindo que cada elemento seja de um tipo, ou seja, são heterogêneas.

```
lista <- list()

class(lista)
[1] "list"
```

Listas

Podemos armazenar alguns objetos de diferentes tipos que trabalhamos até aqui. A indexação ocorre de maneira um pouco diferente, sendo necessário dois colchetes '[[ ]]' a um elemento da lista.

```
lista[[1]] <- dados

lista[[2]] <- planilha4

lista[[3]] <- matriz

lista[[4]] <- "Qualquer string"
```

Listas

Desta maneira, criamos uma lista com 4 elementos. O comando str(...) nos dá a estrutura da lista e de seus elementos.

```
str(lista)
List of 4
 $ :'data.frame':   220 obs. of  4 variables:
  ..$ Salario: int [1:220] 148 165 145 139 142 144 128 143 157 150 ...
  ..$ Sexo   : Factor w/ 2 levels "Feminino","Masculino": 2 2 2 1 1 2 1 2 2 2
  ..$ Posicao: int [1:220] 7 7 5 7 6 5 3 6 7 7 ...
  ..$ Exp    : num [1:220] 16.7 6.7 14.8 13.9 6.4 9.1 8.5 18.2 13 21.6 ...
 $ :'data.frame':   8 obs. of  5 variables:
  ..$ nomes   : Factor w/ 8 levels "ana","bia","caio",..: 1 2 3 4 5 6 7 8
  ..$ nota1   : num [1:8] 7 4 9 3 6 2 9 4
  ..$ nota2   : num [1:8] 10 5 6 2 4 5 5 6
  ..$ entrega1: logi [1:8] TRUE FALSE TRUE FALSE FALSE TRUE ...
  ..$ Presenca: num [1:8] 0.95 0.8 0.95 0.7 0.75 0.85 0.95 0.9
 $ : num [1:3, 1:3] 1 7 17 3 11 19 5 13 23
 $ : chr "Qualquer string"
```

Listas

Após indicar o elemento a ser acessado, já é possível realizar as demais operações de sua natureza, como anteriormente. Por exemplo, sabemos que o primeiro elemento do nosso objeto 'lista' é um data frame, logo, realizar as seguintes operações.

```
summary( lista[[1]] )
    Salario               Sexo        Posicao           Exp
 Min.   :110.0   Feminino : 75   Min.   :1.000   Min.   : 1.70
 1st Qu.:133.0   Masculino:145   1st Qu.:4.000   1st Qu.: 6.60
 Median :143.5                   Median :5.000   Median : 9.50
 Mean   :142.9                   Mean   :5.068   Mean   :10.48
 3rd Qu.:151.0                   3rd Qu.:6.000   3rd Qu.:13.40
 Max.   :172.0                   Max.   :9.000   Max.   :26.10
```

Listas

```
names(lista[[1]])
[1] "Salario" "Sexo"    "Posicao" "Exp"

mean( lista[[1]]$Salario )
[1] 142.8682

lista[[1]]$Salario[10]
[1] 150

lista[[1]][10,1]
[1] 150

class(lista[[1]])
[1] "data.frame"
```

Se atribuirmos o elemento lista[[1]] à algum objeto, os procedimentos ocorrem da mesma maneira.

Listas

O mesmo ocorre com os demais elementos.

```
head( lista[[2]] )   # data.frame
  nomes nota1 nota2 entrega1 Presenca
1   ana     7    10     TRUE     0.95
2   bia     4     5    FALSE     0.80
3  caio     9     6     TRUE     0.95
4  dani     3     2    FALSE     0.70
5   edu     6     4    FALSE     0.75
6  joao     2     5     TRUE     0.85

dim( lista[[3]] )    # matriz
[1] 3 3

nchar(lista[[4]] )   # string
[1] 15
```

Listas

Listas são estruturas muito fortes, permitindo até ter lista dentro de lista

```
teste <- list(x1 = 105,
              x2 = list(2883 , 1790),
              x3 = c(103, 119))

str(teste)
List of 3
 $ x1: num 105
 $ x2:List of 2
  ..$ : num 2883
  ..$ : num 1790
 $ x3: num [1:2] 103 119
```

Listas

Pode ser interessante alocarmos todos os objetos da lista em um só vetor.

Para isso, utilizamos a função 'unlist(...)'. Assim, é possível manipular o vetor de forma usual.

```
unlist(teste)
  x1  x21  x22  x31  x32
 105 2883 1790  103  119

mean( unlist(teste) )
[1] 1000

var( unlist(teste) )
[1] 1637896

unlist(teste)[1]
 x1
105
```

Gráficos

Como dito anteriormente, a análise visual é uma poderosa ferramenta do R.

Diversas plataformas foram desenvolvidas e/ou adaptadas para este cenário no R.

- ggplot2 - Gráficos mais refinados e flexíveis (‘Gramática dos gráficos’).
- plotly - Gráficos interativos.
- Shiny - Visualização interativa, web pages, dashboards.
- leflet - Mapas interativos no R.

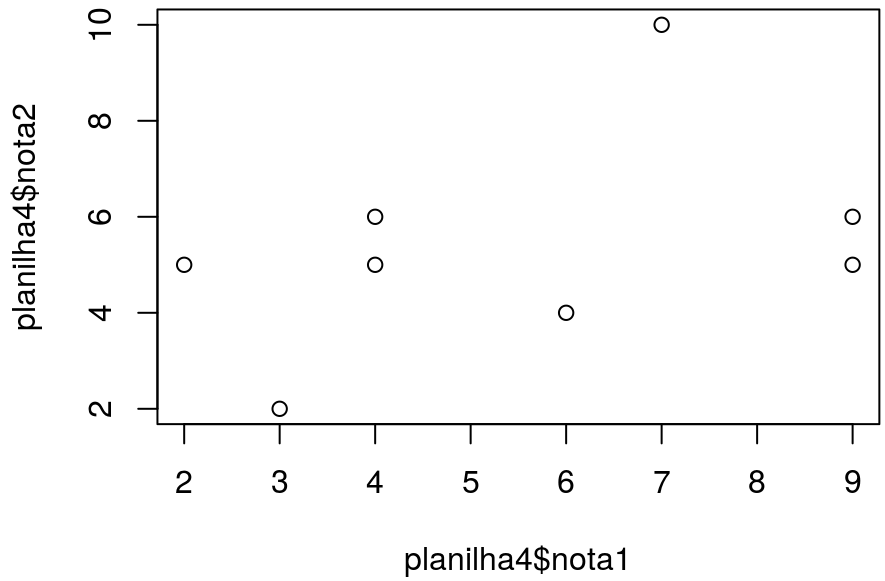
Gráficos

Aqui, focaremos nos recursos básicos do R para gráficos.

- Função plot(x, y, ...)
  - x = coordenada de pontos do eixo X (horizontal).
  - y = coordenada de pontos do eixo Y (vertical).
  - ... = demais argumentos gráficos (?par)
- Outras funções
  - barplot
  - boxplot
  - hist

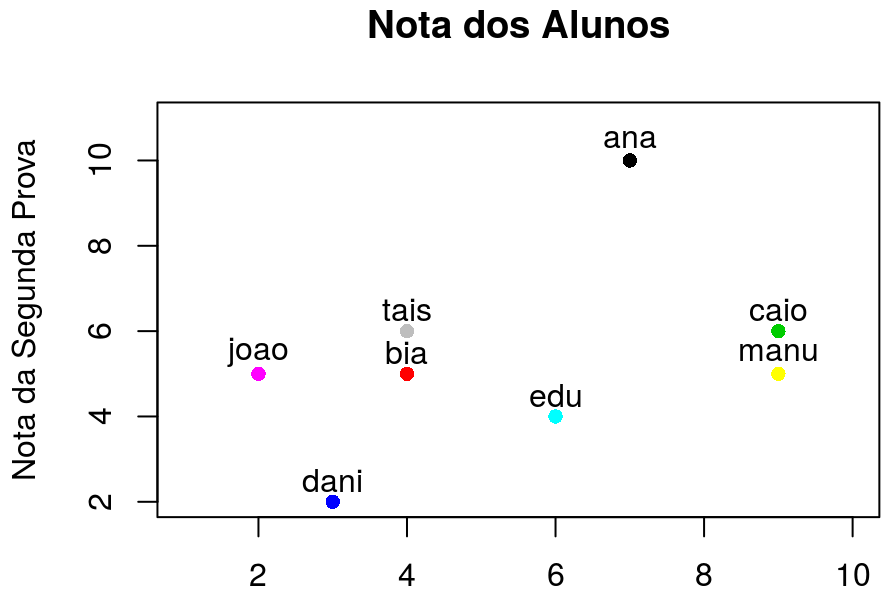
Gráficos

```
# Utilizando o objeto planilha4
plot( x = planilha4$nota1 , y = planilha4$nota2)
```



Gráficos

```
plot( x = planilha4$nota1 , y = planilha4$nota2 , col = planilha4$nomes,
      pch = 16 , xlim = c(1,10) , ylim = c(2,11) ,
      xlab = "Nota da Primeira Prova",   ylab = "Nota da Segunda Prova" ,
      main = "Nota dos Alunos")
text(planilha4$nota1,planilha4$nota2+0.5,planilha4$nomes)
```



Gráficos

Alguns comandos úteis:

- type: “p” para pontos, “l” para linhas.. (ver ?plot)
- lty: tipo de linha; 1 = sólida, 2 = pontilhada .. (ver ?par)
- lwd: largura da linha (default = 1)
- pch: símbolo a ser representado em um gráfico de pontos (ver ?points)
- col: cores
- legenda: função própria (ver ?legend)
- Para outros argumentos, ver: ?par, ?points, ?plot

Exercício

A partir do objeto 'dados', reproduza o gráfico a seguir: