Analytics Inteligência Artificial

Tema da aula
Introdução ao R

LABDATA | FUNDAÇÃO INSTITUTO DE ADMINISTRAÇÃO BUSINESS SCHOOL

Analytics Inteligência Artificial

Professor:
Anderson França

Coordenadores:
Profª Drª Alessandra de Ávila Montini
Profª Dr. Adolpho Walter Pimazoni Canton

FIA Business School

NOSSOS DIFERENCIAIS | QUEM SOMOS

BUSINESS SCHOOL
Graduação, pós-graduação, MBA, Pós- MBA, Mestrado Profissional, Curso In *Company* e EAD

CONSULTING
Consultoria personalizada que oferece soluções baseadas em seu problema de negócio

RESEARCH
Atualização dos conhecimentos e do material didático oferecidos nas atividades de ensino

FUNDAÇÃO INSTITUTO DE ADMINISTRAÇÃO

Único curso de graduação em administração a receber as notas máximas

A primeira escola brasileira a ser finalista da maior competição de MBA do mundo

Única *Business School* brasileira a figurar no *ranking* LATAM

Signatária do Pacto Global da ONU

Membro fundador da ANAMBA - Associação Nacional MBAs

Credenciada pela AMBA - *Association of MBAs*

Credenciada ao *Executive MBA Council*

Filiada a AACSB - *Association to Advance Collegiate Schools of Business*

Filiada a EFMD - *European Foundation for Management Development*

Referência em cursos de MBA nas principais mídias de circulação

Líder em Educação Executiva, referência de ensino nos cursos de graduação, pós-graduação e MBA, tendo excelência nos programas de educação. Uma das principais escolas de negócio do mundo, possuindo convênios internacionais com Universidades nos EUA, Europa e Ásia. +8.000 projetos de consultorias em organizações públicas e privadas.

O **Laboratório de Análise de Dados** – LABDATA é um Centro de Excelência que atua nas áreas de ensino, pesquisa e consultoria em análise de informação utilizando técnicas de ***Big Data***, ***Analytics*** e **Inteligência Artificial**.

O LABDATA é um dos pioneiros no lançamento dos cursos de *Big Data* e *Analytics* no Brasil

Os diretores foram professores de grandes especialistas do mercado

+10 anos de atuação

+1000 alunos formados

Docentes

- Sólida formação acadêmica: doutores e mestres em sua maioria
- Larga experiência de mercado na resolução de *cases*
- Participação em Congressos Nacionais e Internacionais
- Professor assistente que acompanha o aluno durante todo o curso

Estrutura

- 100% das aulas realizadas em laboratórios
- Computadores para uso individual durante as aulas
- 5 laboratórios de alta qualidade (investimento +R$2MM)
- 2 Unidades próximas a estação de metrô (com estacionamento)

Corpo Diretivo

COORDENADORES DO LABDATA | ATUAÇÃO ACADÊMCIA E PROFISSIONAL

Profª Dra.
Alessandra Montini

Diretora do LABDATA-FIA, apaixonada por dados e pela arte de lecionar. Têm muito orgulho de ter criado na FIA cinco laboratórios para as aulas de Big Data e inteligência Artificial. Possui mais de 20 anos de trajetória nas áreas de Data Mining, Big Data, Inteligência Artificial e Analytics. Cientista de dados com carreira realizada na Universidade de São Paulo. Graduada e mestra em estatística aplicada pelo IME-USP e doutora pela FEA-USP. Com muita dedicação chegou ao cargo de professora e pesquisadora na FEA-USP, ganhou mais de 30 prêmios de excelência acadêmica pela FEA-USP e mais de 30 prêmios de excelência acadêmica como professora dos cursos de MBA da FIA. Orienta alunos de mestrado e de doutorado na FEA-USP. Membro do Conselho Curador da FIA, Coordenadora de Grupos de Pesquisa no CNPQ, Parecerista da FAPESP e Colunista de grandes Portais de Tecnologia.

linkedin.com/in/alessandramontini/

Prof. Dr.
Adolpho Walter Canton

Diretor do LABDATA-FIA. Consultor em Projetos de *Analytics*, *Big Data* e Inteligência Artificial. Professor FEA – USP. PhD em Estatística Aplicada pela *University of North Carolina at Chapel Hill*, Estados Unidos.

1. Introdução

Por quê R?

- De graça, open source, e está disponíveis nas principais plataformas.
- Muitos pacotes para modelagem estatística, machine learning, visualização, importação e manipulação de dados. Se você tem um problema, muitas pessoas já tiveram esse problema.
- Foi desenvolvido para conectar com linguagem de programação de alta performance como Fortran, C e C++.
- Existem muitos pacotes que já facilitam a geração de relatórios em pdf ou html e permite criar websites interativos.

Fonte: Advanced R by Hadley Wickham

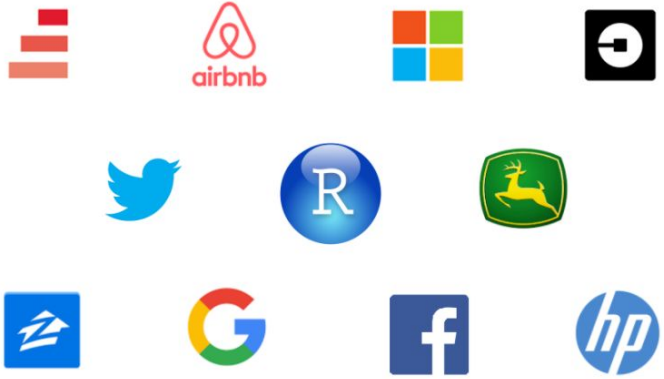
Desafios

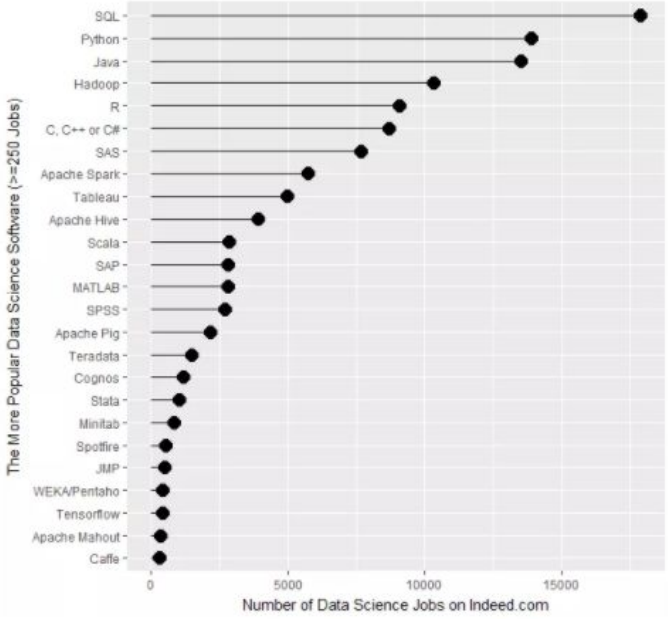
- Muitos códigos que encontramos no dia a dia foram escritos para resolver um determinado problema. E isso torna o código pouco elegante, não tão rápido ou mais complexo de ser entendido.
- Comparado aos outros programas, a comunidade R tende a ser mais focado no resultado ao invés de focar no processo. O conhecimento das melhores práticas de engenharia de software é irregular: por exemplo, não há programadores de R suficientes para usar o controle de código-fonte ou automação de testes.
- R não é uma linguagem de programação particularmente rápida, e o código R mal escrito pode ser terrivelmente lento. R também é um grande vilão da memória do seu computador.

Fonte: Advanced R by Hadley Wickham

Motivação

O R já está nas principais empresas do mundo todo
É possível criar relatórios profissionais em diversos formatos
Aplicativos interativos
Estatística: do básico ao avançado





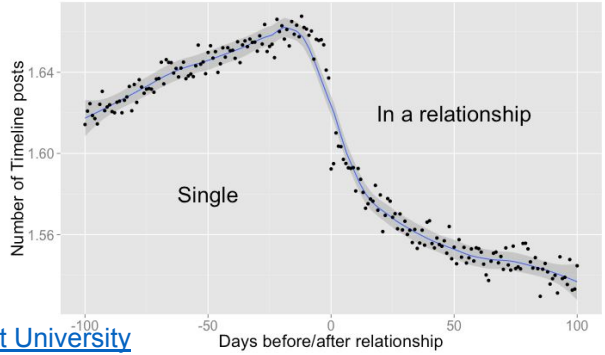
Uso nas empresas

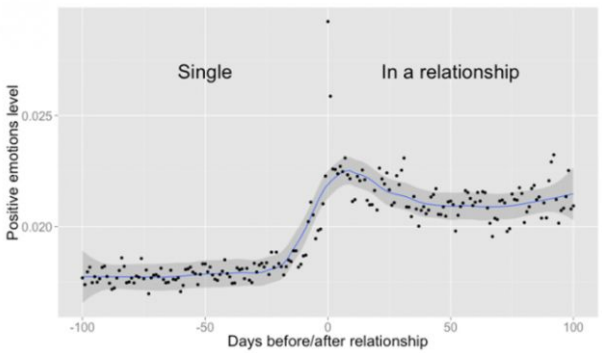
Airbnb

80% dos cientistas de dados utilizam o R e 64% utilizam como ferramenta primária de análise de dados. É muito utilizado pra prever a taxa de retorno utilizando as avaliações passadas para combinar hóspede com host.

Facebook

O Facebook usou R para realizar análises de comportamento com base em atualizações de status e de imagens de perfil. Um cientista de dados do Facebook, criou uma análise sobre a formação do amor, com base nas atualizações de status do relacionamento do Facebook. O que eles descobriram é que, à medida que um relacionamento é formado, o número de postagens da linha de tempo diminui. No entanto, a quantidade de emoções positivas nas postagens aumenta.

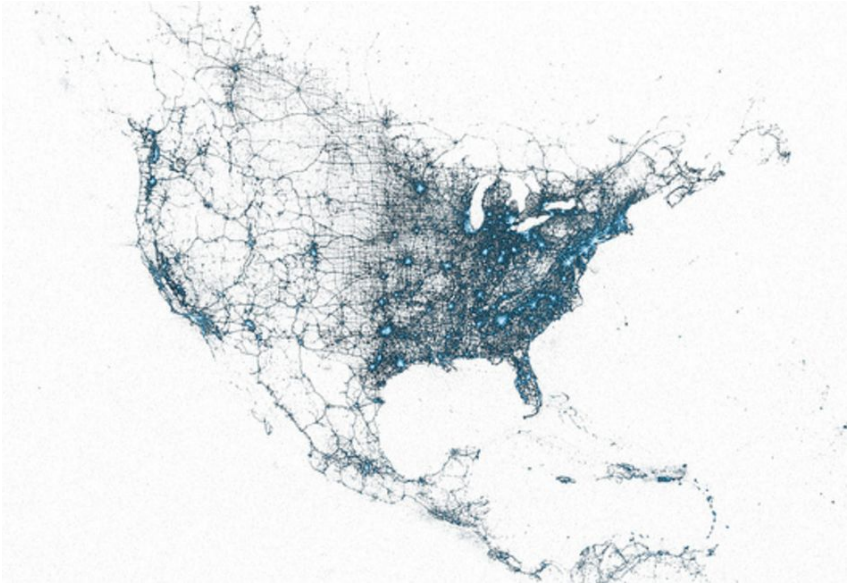




Source: [Northeast University]

Uso nas empresas

Twitter
Com o R, o Twitter criou alguns projetos bem interessantes. Abaixo estão os dados georreferenciados que representam todos os tweets nos Estados Unidos desde 2009. O Twitter também criou pacotes open-source para detecção de anomalia, o que ajudou a melhorar a experiência do cliente.



Source: Northeast University

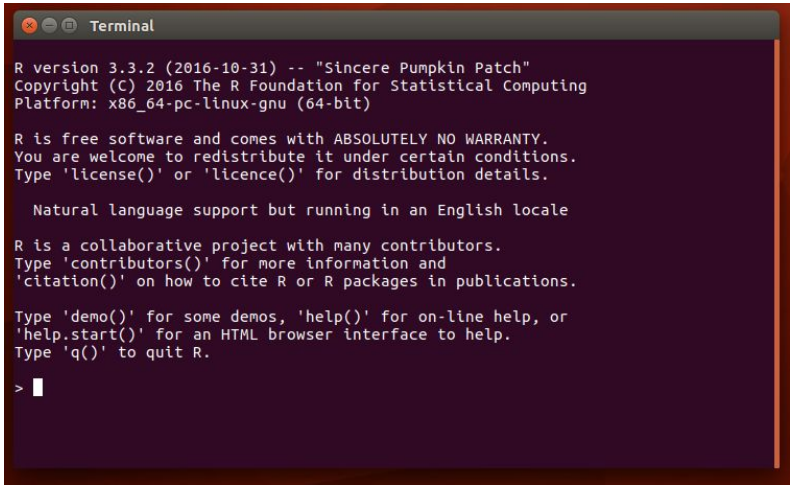
Instalação

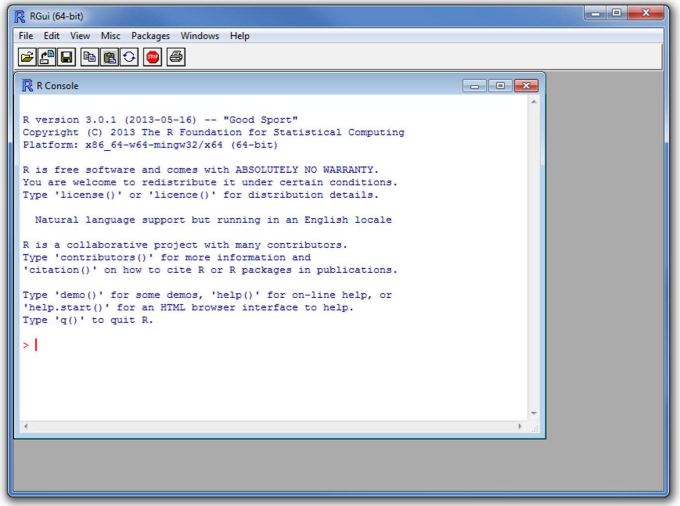
https://www.r-project.org/

https://www.rstudio.com/

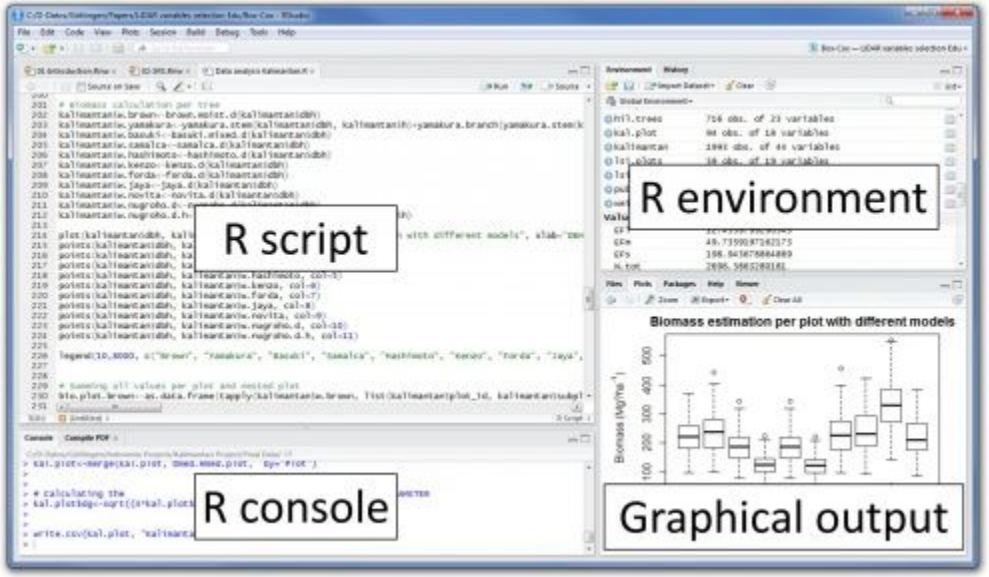
https://mran.microsoft.com/open/

R GUI


Terminal
R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>


RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
R version 3.0.1 (2013-05-16) -- "Good Sport"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>

O Rstudio


R script
R environment
R console
Graphical output
Biomass estimation per plot with different models

Preciso de ajuda

- help(median)
- ?median

Para recursos que contenham caracteres especiais, utiliza-se aspas simples ou duplas, transformando em caractere de texto.

- help(“[[”)
- Google it
- https://stackoverflow.com/
- https://cran.r-project.org/

Primeiros passos

getwd(): consulta o diretório de trabalho atual

setwd(): altera o diretório de trabalho (o caminho dos arquivos pode conter apenas \\ ou /)

install.packages(“<nome_do_pacote>”): Instalar os pacotes

library(<nome_do_pacote>): carrega o pacote contendo as funções utilizadas

rm( ): remover objetos

Pacotes

Mais de 6.000 pacotes (conjunto de funções reutilizáveis) disponíveis.
Exemplos:

Pacote	Descrição
Caret	Conjunto de funções para machine learning padronizando o uso de diferentes pacotes.
ddply e reshape	Funções para manipulação de base de dados
arm	Modelos de regressão hierárquicos
ggplot2	Construção de diversos tipos de gráficos mais elaborados.
igraph	Análise de redes sociais
tm	Coleção de funções para text mining em R.
XML e RJSONIO	Funções para facilitar o manuseio de dados em XML e JSON respectivamente.

O R é *case sensitive*

A é diferente de **a**.

R como calculadora

Operadores aritméticos

Operador	Descrição
x + y	Adição de x com y
x - y	Subtração de y em x
x * y	Multiplicação de x e y
x / y	Divisão de x por y
x^y ou x**y	x elevado a y-ésima potência
x%%y	Resto da divisão de x por y (módulo)
x%/%y	Parte inteira da divisão de x por y

Vamos tentar

```
> 6 * 6
[ 1 ] 36

> 7 / 2
[ 1 ] 3.5

>(4*(6+2))/4
[ 1 ] 8

> 4 / 0
[ 1 ] Inf
```

Valores especiais

Inf / -Inf = divisões por 0, valores da ordem de 10^308. Não retorna erro, mas infinito.

```
> pi / 0
> 1 / 0 + 1 / 0
```

NaN = Indeterminações matemáticas, como 0/0 e log(-1)

```
> 0 / 0
> 1 / 0 - 1 / 0
```

R como calculadora

As operações e suas precedências são mantidas como na matemática, ou seja, divisão e multiplicação são calculadas antes da adição e subtração.

Falando em matemática, notaram que no último exemplo apareceu a função pi?

```
> pi
```

```
[ 1 ] 3.141593
```

Existem funções incorporadas em R que podemos chamar para fazer as coisas de forma mais simples

```
> (sin(pi/2) + 2)*3
```

```
[ 1 ] 9
```

sin(), cos(), tan(), asin(), acos(), atan(), atan2(), log(), log10(), exp(), sqrt(), factorial()…
E muito mais: Short R reference card

Operadores lógicos

Operadores lógicos retornarão sempre ou `TRUE` ou `FALSE`. Eles definem perguntas que aceitam apenas verdadeiro e falso como resposta.

Operador	Descrição
x < y	x é menor que y?
x <= y	x menor ou igual a y?
x > y	x maior que y?
x >= y	x maior ou igual a y?
x == y	x igual a y
x != y	x diferente de y?
!x	Negativa de x?
x & y	x e y são verdadeiros?
x | y	x ou y são verdadeiros?
xor(x,y)	x ou y são verdadeiros? (apenas um deles)

Vamos tentar

Qual o valor esperado para as entradas abaixo?

```
> 1 < 1                          [1] FALSE
> 1 <= 1                         [1] TRUE
> 1 == 0.999                     [1] FALSE
> 13.5 != 13.5                   [1] FALSE
> !TRUE                          [1] FALSE
> FALSE                          [1] FALSE
> TRUE & FALSE                   [1] FALSE
> TRUE & TRUE                    [1] TRUE
> xor(TRUE, TRUE)                [1] FALSE
> xor(TRUE, FALSE)               [1] TRUE
> TRUE == 1                      [1] TRUE
> TRUE == 2                      [1] FALSE
> FALSE == 0                     [1] TRUE
```

Números complexos

O R manipula números complexos de forma fácil e intuitiva da mesma forma que trata números reais. Além disso, existem funções úteis para eles, como módulo, conjugado e argumento.

Operador	Descrição
Re(z)	Parte real de z
im(z)	Parte imaginária de z
Mod(z)	Módulo de z
Arg(z)	Argumento de z
Conj(z)	Complexo conjugado de z

Definir variável

Para definir uma variável, basta escolher um nome (*lembre-se das regras de nomes no R*) e atribuir a ela um valor:

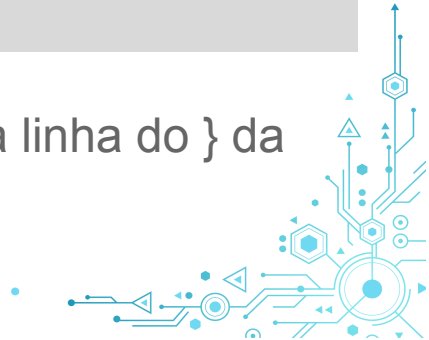
```
minha.variavel = 1
```

If, else e else if

Estrutura

```
if(<condição1>) {
 Faça algo
} else if (<condição2>) {
  # caso a condição1 seja falsa e a condição2 seja verdadeira...
 Faça outra coisa
} else {
  # faz coisas necessárias caso todas as condições
  # anteriores falhem
}
```

Ponto de atenção: O else e o else if têm que estar na mesma linha do } da expressão anterior, senão não rodará!



If, else e else if

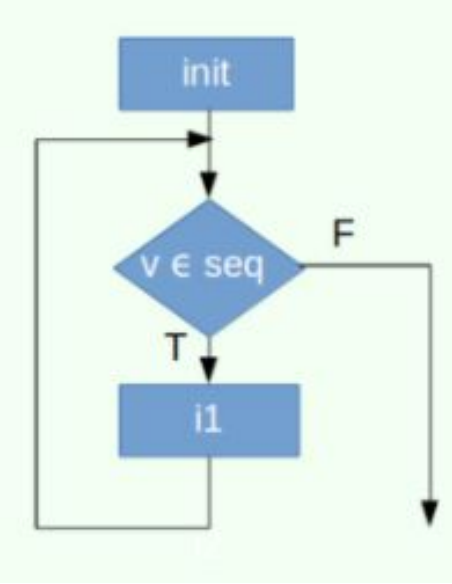
```
# Certo
if(1 == 2) {
  "Resultado esperado"
} else { # <----- Mesma linha
  "1 é diferente de 2"
}


# ERRADO!!!
if(1 == 2) {
  "Resultado esperado"
}
else { # <----- Na linha abaixo do "}"
  "1 é diferente de 2"
}
```

If, else e else if

- Além de TRUE e FALSE, o R aceita 1 e 0
- Objetos character, NA, NaN e list não são interpretáveis como lógicos
- Caso seja passado um array, vector ou matrix, será utilizado apenas o primeiro elemento (evitar!)
- else e else if são opcionais.

for

O **for** é um tipo "laço" (loop, em inglês) que aplica um bloco de código para um número fixo de iterações. Geralmente, um **for** percorre um vetor e utiliza um elemento diferente deste vetor em cada iteração.




```
for (i in 1:4){
j <- i + 10
     print(j)
}
```

Iterar um vetor de palavras

```
checklist <- c("laticinios", "frutas", "produtos
de limpeza")

for(check in checklist) {
  print(paste(check, " - ok", sep = ""))
}
```

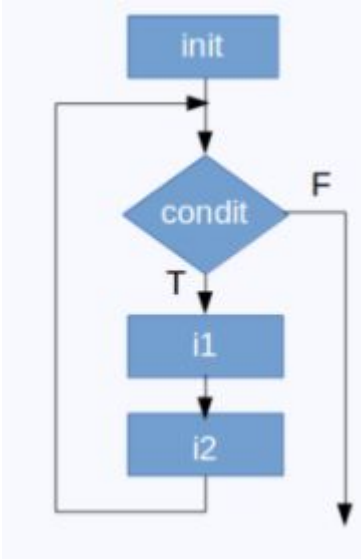
for

- Frequentemente é vantagem iterar sobre índices do vetor em vez dos valores propriamente ditos

- O for é especialmente ineficiente no R. Ao contrário das outras linguagens, o uso do for será menos frequente conforme você vai se aprimorando no R

while

O laço **while** é especialmente útil quando não sabemos quando iremos parar (a condição para o laço deixar de rodar pode envolver o acaso ou convergência, por exemplo)..




```
while (i < 5){
      print(i)
      i <- i + 1
}
```

```
proba <- 0.2
chances <- 0
while(runif(1) > proba) {
  chances <- chances + 1
}
chances
```

Funções

Uma das maiores vantagens de R é a capacidade do usuário para adicionar funções. Na verdade, muitas das funções em R são, na verdade, funções de funções. A estrutura de uma função é dada abaixo.

```
Nome_funcao <- function(var){
     Faça algo!
     return(nova_variavel)
}
```

```
quadrado <- function(x){
     quadrado <- x*x
     return(quadrado)
}
```

Exercícios

1. Utilizando o que aprendemos até agora, vamos construir uma função que calcule o valor que você irá receber de salário baseado na quantidade de horas de trabalho

a) A função deve receber o número de horas trabalhadas (horas) e o preço negociado por hora (pnh)

Exercícios

2. Agora suponhamos que seu negócio deu muito certo, e alguns clientes trouxeram muito mais trabalho e para deixá-lo feliz, você decidiu dar um desconto de 10% por hora para projetos que envolvam mais de 100 horas de trabalho. Você precisa ajustar sua função para calcular o novo preço.

a) A função deve receber o número de horas trabalhadas (horas) e o preço negociado por hora (pnh).

Dica: utilize o if dentro de sua função

Estrutura de dados

A estrutura dos dados podem ser organizadas por sua dimensionalidade (1d, 2d, ou nd) e eles são homogêneos (todos os conteúdos devem ser do mesmo tipo) ou heterogêneo (os conteúdos podem ser de diferentes tipos). Isso dá origem a 5 tipos de dados usados com mais frequência em análise de dados:

	Homogêneo	Heterogêneo
1d	Vetor atômico	Lista
2d	Matriz	Data Frame
nd	Array	

Vetores e atribuições

O R opera em estruturas de dados nomeados. A estrutura mais simples é o vetor, que é uma entidade única constituída por uma coleção ordenada de números ou palavras e armazenam os dados a serem analisados.

$$f([x_1, \ldots, x_n]) = [f(x_1), \ldots, f(x_n)]$$

Exemplo:

```
> vetor1 <- 1:3
> vetor1
[1] 1 2 3
>2*vetor1
[1] 2 4 6
> sin(vetor1)
[1] 0.84144710 0.9092974 0.1411200
```

Eficiência

Implementar funções vetorizadas seria bem simples, não fosse a lentidão dos *loops.* Porém, as funções mais úteis do R são implementadas em linguagem de baixo nível, como Fortran, C ou C++.
O exemplo abaixo mostra o cálculo da raiz quadrada de cada elemento de um vetor de números.

```
> x <- 1:1000000 # sequência de inteiros de 1 a 1.000.000
> # função para calcular a raiz quadrada de cada elemento de um
vetor de numerico
> minha_raiz <- function(numeros) {
+ resp <- numeric(length(numeros))
+ for(i in seq_along(numeros)) {
+   resp[i] <- sqrt(numeros[i])
+ }
+ return(resp)
+ }
```

```
> system.time(loop <- minha_raiz(x))
      Usuário       Sistema       decorrido
         1.64          0.03            1.69
```

```
> system.time(vetor <- sqrt(x))
      Usuário       Sistema       decorrido
         0.02          0,00            0,02
```

Criando Vetores

Para agrupar elementos em um vetor utilizamos a função concatenar c().

```
> c(1, 3, 5)
[1] 1 3 5
```

Uma sequência de valores.

```
> 1:5
[1] 1 2 3 4 5
```

Uma sequência de valores complexos.

```
> seq(1, 5, by=0.5)
 [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

Uma sequência de valores complexos.

```
> rep(1:2, times=3 )
[1] 1 2 1 2 1 2
```

Funções de vetores

Vamos criar um objeto chamado **x** e atribuir os valores **5, 4, 9, 6, 8, 9, 23, 9, 9**

Retorna o vetor de x ordenado

```
> sort(x)
```

Visualizar a contagem dos valores

```
> table(x)
```

Retornar x invertido

```
> rev(x)
```

Visualizar valores únicos

```
> unique(x)
```

Seleção de vetores

Por posição

```
> x[4]          #Seleciona o elemento 4
> x[-4]         #Todos os elementos menos o 4
> x[2:4]        #Elementos de 2 a 4
> x[-(2:4)]     #Todos os elementos exceto 2 a 4
> x[c(1,5)]     #Elementos 1 e 5
```

Seleção de vetores

Por valor

```
> x[x == 10]              #Elementos igual a 10
> x[x < 0]                #Todos os elementos menor que 0
> x[x %in% c(1,2,5)]      #Elementos em 1, 2, 5
```

Por valor

```
> x['banana']             #Elemento chamado 'banana'
```

Matrizes

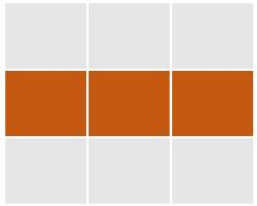
Criar uma matriz

```
> matriz <- matrix(1:12, nrow = 3, ncol = 4) #criar uma matriz
> matriz
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

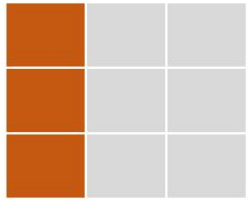
Se o argumento `data` tem menos elementos do que a matriz, eles são repetidos até preenchê-la

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

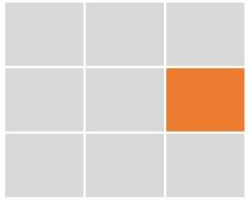
Seleção de matrizes



```
#selecionar uma linha
> matriz[2,]
```



```
#selecionar uma coluna
> matriz[,1]
```



```
#selecionar um elemento
> matriz[2,3]
```

```
#Transpor matriz
> t(matriz)
```

```
#Multiplicação de matriz
> matriz1 %*% matriz2
```

Listas

Uma lista é um conjunto de elementos que pode conter quaisquer tipos de dados.

```
#Criar uma lista
> lista <- list(x = 1:5, y = c("a","b"))
> lista
$x
[1] 1 2 3 4 5
$y
[1] "a" "b"
```

Seleção de listas

```
> lista[[2]]  #Segundo elemento de lista

> lista[2]    #nova lista com somente o primeiro elemento

> lista$x     #Elemento chamado x

> lista['y']  #nova lista com somente o elemento chamado y
```

Data Frames

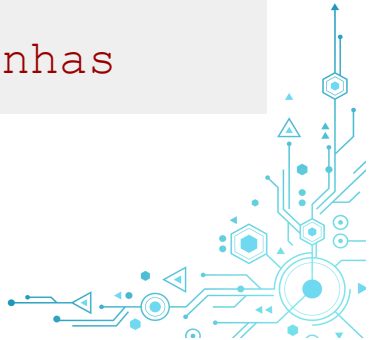
Com a função `data.frame` reunimos vetores de mesmo comprimento em um só objeto:

```
> df <- data.frame(x = 1:3, y = c("a", "b", "c"))
```

x	y
1	a
2	b
3	c

```
> View(df) #visualizar data frame
```

```
> head(df) #visualizar 6 primeiras linhas
```



Selecionar Data Frames

x	y
1	a
2	b
3	c

```
#selecionar uma coluna
> df[,2]
```

x	y
1	a
2	b
3	c

```
#selecionar uma linha
> df[2,]
```

x	y
1	a
2	b
3	c

```
#selecionar um elemento
> df[2,2]
```

```
#número de linhas
> nrow(df)
```

```
#número de colunas
> ncol(df)
```

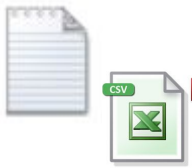
```
#número de coluna e linha
> dim(df)
```

Alterando valores

Para selecionar um subconjunto de um objeto no R, utilizamos um método chamado subsetting.

```
> x <- c(17,21,8,4,9,10,11)
> x
[1] 17 21 8 4 9 10 11

> x[x >= 10] <- 0
> x
[1] 0 0 8 4 9 0 0
```

Ler e escrever dados



Arquivos Texto, CSV, XML, JSON, ...

Banco de dado relacional

Banco de dados distribuidos

Ler e escrever dados

Input	Ouput
df <- read.table('file.txt')	write.table(df, 'file.txt')
df <- read.csv('file.csv')	write.csv(df, 'file.csv')
load('file.RData')	save(df, file = 'file.Rdata')

```
#Excel
library(xlsx)
dados <- read.xlsx("c:\meuexcel.xlsx", 1)
```

```
#Spss
#transforme o arquivo em .por
Get file = 'c:\meuexcel.sav'
export outfile= c:\meuexcel.por'

library(Hmisc)
dados <- spss.get("c:/meuexcel.por",
                  use.value.labels=TRUE)
```

```
#SAS
#transforme o arquivo em .xpt
libname out xport 'c:/meuexcel.xpt';
data out.dados;
set sasuser .dados;
run;

library(Hmisc)
dados <- sasxport.get("c:/meuexcel.xpt")
```

Exercícios

1. Verificar o tipo de dados da base de cartões type()
2. Selecionar apenas as linhas que contenham segmento C
3. Selecionar apenas as linhas em que o Segmento seja A e atribuir uma variável chamada Segmento.A
4 . Calcular a média, mediana, variância e desvio padrão da nova variável Segmento.A

Gráficos

O R é uma poderosa ferramenta para gerar gráficos.

```
x <- rnorm(100)
y <- x*3 +rnorm(100,0,2.8)

par(mfrow = c(1,2))

plot(x,y, pch=16, main="Gráfico de dispersão X vs Y",
     xlab = "Variável X", ylab = "Variável Y")

hist(x, main="Histograma de X",
     xlab = "Variável X", ylab = "Frequência")
```


SIBLE CITY
anean migrant deaths by route
Western Mediterran
Apulia & Calabria
Eastern Mediterranean
Central Mediterranean 2,447
646
2,267
2,373
to 24 August 2015
deaths in the Mediterranean by month
2015
Feb Mar Apr May Jun Jul Aug Sep Oct
grant route to Germany
Hamburg
GERMANY
Frankfurt
Vienna
Munich
HUNGARY
AUSTRIA
Belgrade
SERBIA
MACEDONIA
GREECE
Athens
HOW MANY YEARS LEFT
July 1979
November 1994
July 1996
August 2000
Not all Egyptian Twitter users in the Middle East Network designated themselves as Egyptian—but their slowest tweets indicate their location.
Messages from many users outside Egypt increased, possibly in response to the service interruption.
Surgical
Nonsurgical
£115.85bn
+5.5%
Vegetables Shipped to U.S. Di
April
4,852
shipments to U.S.
from Central America to Miami, Fla.
FEEDSTOCK
Petro Products
Transportation
FUEL/ENERGY
Gay rights by type
GERMANY
Vienna
Munich
AUSTRIA
HUNGARY
Belgrade
SERBIA
GREECE
SAHARA
AFRICA
ALGERIA
MALI
NIGERIA
SUDAN
ETHIOPIA
SOUTH SUDAN
KENYA
CONGO
TANZANIA
ANGOLA
ZAMBIA
SAUDI ARABIA
Number of asylum claims
No data

Vamos tentar

Base simulada com 150 observações e 5 variáv

- Gastos no cartão em reais
- Idade
- Renda
- Pagamento de impostos
- Segmento

Objetivo:
Analisar a distribuição de cada variável;
Verificar a relação entre as variáveis

Análise Descritiva

- Medidas resumo das variáveis
- Distribuição das variáveis
- Análise de missings e outliers
- Correlações de Pearson
- Relação bivariada entre variáveis
- Análise multivariada

Medida de resumo das variáveis

Carregando a base de dados:

```
1  ### Descriptive analysis
2
3  setwd("C:/Users                                    16")
4  dados <- read.table("base_gastos_cartao.csv", sep=",", header=T)
5  head(dados)
6
```

5:12 (Top Level)

Console

```
> setwd("C:/Users/                                    ")
> dados <- read.table("base_gastos_cartao.csv", sep=",", header=T)
> head(dados)
  Gastos_Cartao Idade Renda Impostos Segmento
1           510    35  1120       60        C
2           490    30  1120       60        C
3           470    32  1040       60        C
4           460    31  1200       60        C
5           500    36  1120       60        C
6           540    39  1360      120        C
```

Medida de resumo das variáveis

Sumarizando os dados:

```
7  summary(dados)
8  table(dados$Segmento)
9:1  (Top Level)

Console
> summary(dados)
 Gastos_Cartao       Idade           Renda          Impostos      Segmento
 Min.   :430.0   Min.   :20.00   Min.   : 800   Min.   : 30.0   A:50
 1st Qu.:510.0   1st Qu.:28.00   1st Qu.:1280   1st Qu.: 90.0   B:50
 Median :580.0   Median :30.00   Median :3480   Median :390.0   C:50
 Mean   :584.3   Mean   :30.57   Mean   :3006   Mean   :359.8
 3rd Qu.:640.0   3rd Qu.:33.00   3rd Qu.:4080   3rd Qu.:540.0
 Max.   :790.0   Max.   :44.00   Max.   :5520   Max.   :750.0
> table(dados$Segmento)

 A  B  C
50 50 50
```

Medida de resumo das variáveis

Quantil:

```
 9  quantile(dados$Gastos_Cartao,
10           probs = c(0.01, 0.05, 0.1, 0.9, 0.95, 0.99))
11  quantile(dados$Idade,
12           probs = c(0.01, 0.05, 0.1, 0.9, 0.95, 0.99))
13  quantile(dados$Renda,
14           probs = c(0.01, 0.05, 0.1, 0.9, 0.95, 0.99))
15  quantile(dados$Impostos,
16           probs = c(0.01, 0.05, 0.1, 0.9, 0.95, 0.99))
17
14:11  (Top Level)
```

Console

```
> quantile(dados$Gastos_Cartao,
+          probs = c(0.01, 0.05, 0.1, 0.9, 0.95, 0.99))
   1%    5%   10%   90%   95%   99%
440.0 460.0 480.0 690.0 725.5 770.0
> quantile(dados$Idade,
+          probs = c(0.01, 0.05, 0.1, 0.9, 0.95, 0.99))
   1%    5%   10%   90%   95%   99%
22.00 23.45 25.00 36.10 38.00 41.51
> quantile(dados$Renda,
+          probs = c(0.01, 0.05, 0.1, 0.9, 0.95, 0.99))
    1%     5%    10%    90%    95%    99%
 919.2 1040.0 1120.0 4640.0 4880.0 5360.0
> quantile(dados$Impostos,
+          probs = c(0.01, 0.05, 0.1, 0.9, 0.95, 0.99))
 1%  5% 10% 90% 95% 99%
 30  60  60 660 690 750
```

Medida de resumo das variáveis

Distribuição das variáveis:

```
18  vars <- names(dados)[1:4]
19  for(i in vars){
20    par(mfrow=c(2,1))
21    hist(dados[,i], breaks = 20, main=paste0("Histograma - ", i),
22         xlab=i, ylab="Frequência", col="dark blue")
23    boxplot(dados[,i], main=paste0("Boxplot - ", i),
24            ylab=i, col="dark red")
25  }
```

Medida de resumo das variáveis

Missings e Outliers:

```
27  vars <- names(dados)
28  for(i in vars){
29    cat(paste0(i, " - número de observaçôes missing: ",
30               sum(is.na(dados[,i]))),"\n")
31  }
32
```

31:2 (Top Level)

Console

```
> vars <- names(dados)
> for(i in vars){
+   cat(paste0(i, " - número de observaçôes missing: ",
+              sum(is.na(dados[,i]))),"\n")
+ }
Gastos_Cartao - número de observaçôes missing: 0
Idade - número de observaçôes missing: 0
Renda - número de observaçôes missing: 0
Impostos - número de observaçôes missing: 0
Segmento - número de observaçôes missing: 0
```

Medida de resumo das variáveis

Correlações de Pearson:

```
33  vars <- names(dados)[1:4]
34  cor(dados[,vars])
35
35:1  (Top Level)
```

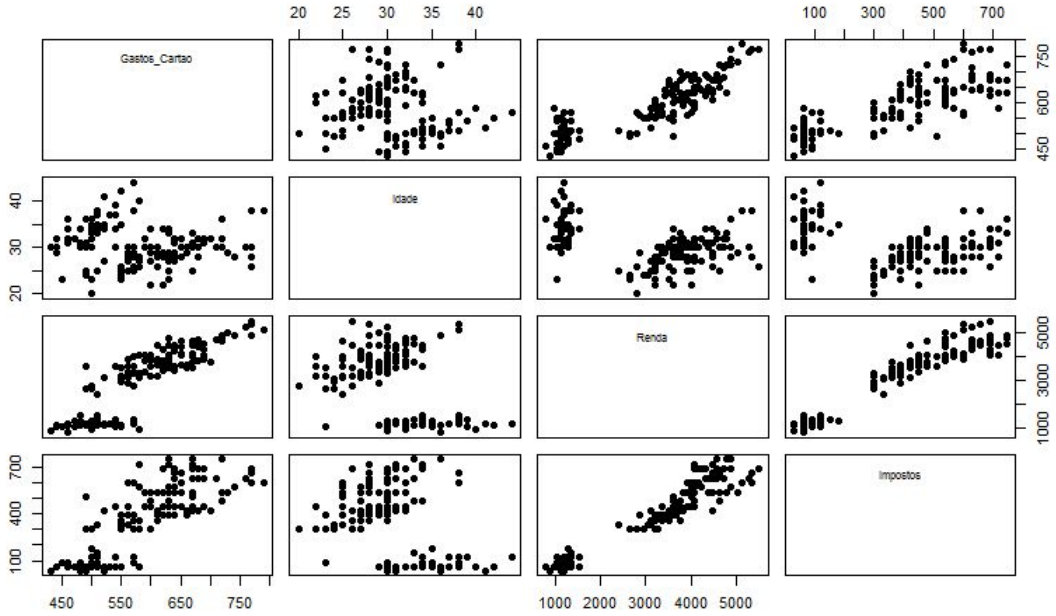
Console

```
> vars <- names(dados)[1:4]
> cor(dados[,vars])
             Gastos_Cartao      Idade      Renda   Impostos
Gastos_Cartao     1.0000000 -0.1175698  0.8717538  0.8179411
Idade            -0.1175698  1.0000000 -0.4284401 -0.3661259
Renda             0.8717538 -0.4284401  1.0000000  0.9628654
Impostos          0.8179411 -0.3661259  0.9628654  1.0000000
```

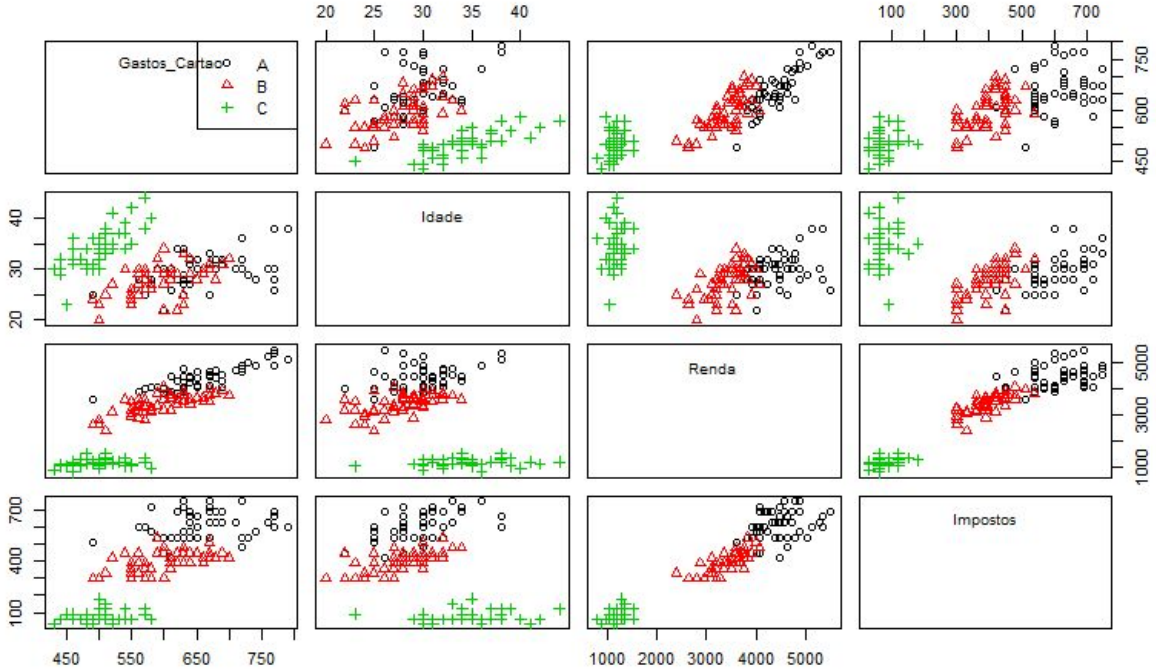
Análise Bivariada

```
36  library(car)
37  scatterplotMatrix(~ Gastos_Cartao + Idade + Renda +
38                      Impostos, data=dados, smooth=FALSE,
39                    reg.line=FALSE, ellipse=FALSE,
40                    diagonal="none", pch=16)
41
42  scatterplotMatrix(~ Gastos_Cartao + Idade + Renda +
43                      Impostos, data=dados, smooth=FALSE,
44                    reg.line=FALSE, ellipse=FALSE,
45                    groups=as.factor(dados$Segmento), diagonal="none")
```

Análise Bivariada

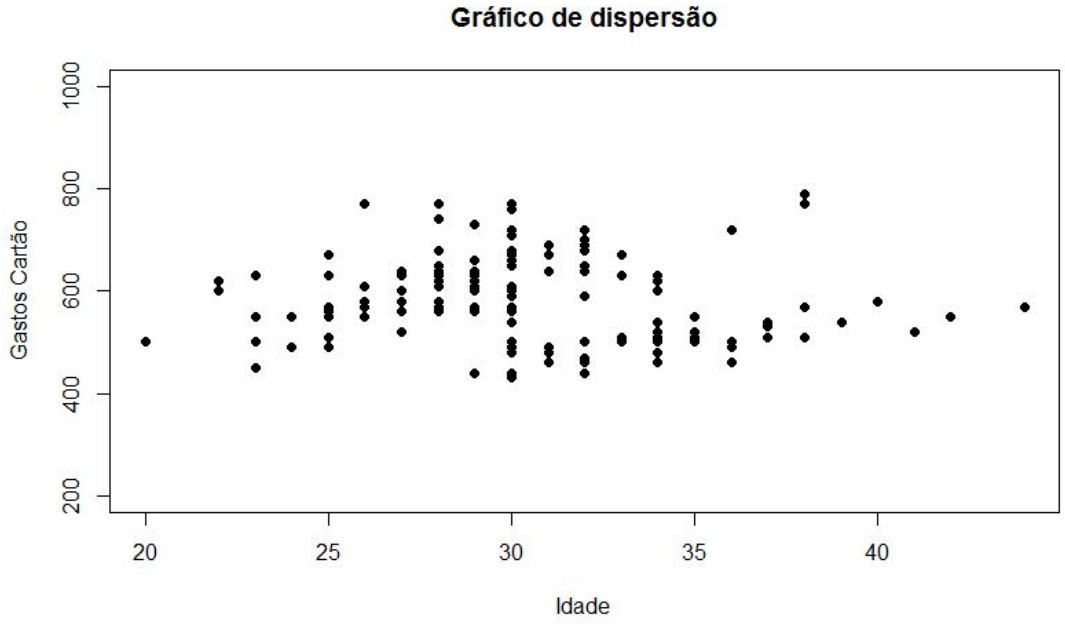


Análise Bivariada



Análise Bivariada

```
47  plot(dados$Idade, dados$Gastos_Cartao, pch=16, main="Gráfico de dispersão",
48       xlab = "Idade", ylab="Gastos Cartão", ylim=c(200,1000))
```



OBRIGADO

Anderson França
Email: contato@andersonfranca.me
LinkedIn: /andersonfrancal/