Introdução ao R e RStudio

Aula Estágio de Docência Teoria do Risco

Luiz Carlos Leão

16/05/2024

Um pouco da história do R

- R é uma linguagem de programação estatística e gráfica abrangente e é derivada da linguagem S:
  - 1988 - S2: RA Becker, JM Chambers, A Wilks
  - 1992 - S3: JM Chambers, TJ Hastie
  - 1998 - S4: JM Chambers
- R: inicialmente escrito por Ross Ihaka e Robert Gentleman no Dep. de Estatísticas da Universidade de Auckland, Nova Zelândia, durante a década de 1990
- Desde 1997: equipe internacional “R-core” de 15 pessoas com acesso ao arquivo fonte

Ross Ihaka

Robert Gentleman

Por que R?

- É um software livre
- Roda em uma variedade de plataformas, incluindo Windows, Unix e MacOS
- Fornece uma plataforma para programar novos métodos estatísticos de maneira fácil e direta
- Contém rotinas estatísticas avançadas ainda não disponíveis em outros pacotes
- Possui múltiplos recursos gráficos

R possui uma curva de aprendizado

- Embora existam muitos tutoriais introdutórios (cobrindo tipos de dados, comandos básicos, interface), nenhum por si só é abrangente
- Em parte, isso ocorre porque grande parte das funcionalidades avançadas do R vem de centenas de pacotes contribuídos por usuários
- Procurar o que você deseja pode consumir muito tempo e pode ser difícil obter uma visão clara de quais procedimentos estão disponíveis

O paradigma de R é diferente

- Em vez de configurar uma análise completa de uma só vez, o processo é altamente interativo
- Você executa um comando (digamos, ajustar um modelo), pega os resultados e os processa por meio de outro comando (digamos, um conjunto de gráficos de diagnóstico), pega esses resultados e os processa por meio de outro comando (digamos, validação cruzada), etc. incluem a transformação dos dados e o retorno de todo o processo
- Você para quando percebe que analisou totalmente os dados

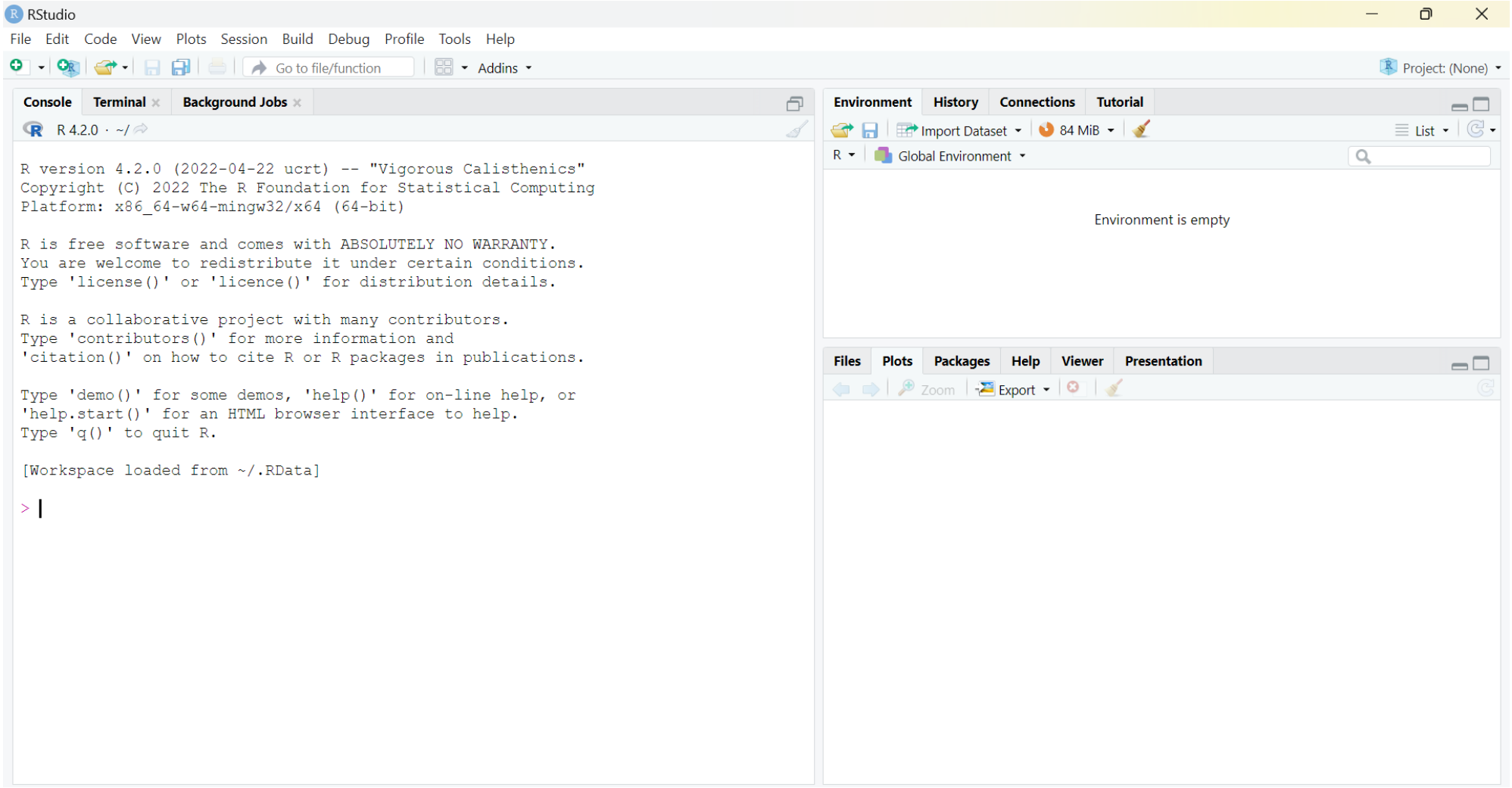
Como baixar e instalar o R?

- Para instalar R no Windows:
  - 1 Baixe o arquivo binário para R https://cran.
    - r-project.org/bin/windows/base/R-x.y.z-win.exe
  - 2 Abra o arquivo .exe baixado e instale R
- Para instalar R no Mac:
  - 1 Baixe a versão apropriada do arquivo .pkg
    - https://cran.r-project.org/bin/macosx/
  - 2 Abra o arquivo .pkg baixado e instale R

RStudio

- RStudio é uma interface de usuário gratuita para R.
  - 1 Instale a versão apropriada do RStudio https:
    - https://posit.co/download/rstudio-desktop/
  - 2 Execute-o para instalar o RStudio

Tela inicial do RStudio



Visão geral do R

- Você pode inserir comandos um de cada vez no prompt de comando (>) ou executar um conjunto de comandos a partir de um arquivo de origem.
- Existe uma grande variedade de tipos de dados, incluindo vetores (numéricos, caracteres, lógicos), matrizes, quadros de dados e listas.
- Para sair do R, use
  - >q()

Visão geral do R

- A maior parte das funcionalidades do R são fornecidas por meio de funções integradas e criadas pelo usuário e todos os objetos de dados são mantidos na memória durante uma sessão aberta
- As funções básicas estão disponíveis por padrão
- Outras funções estão contidas em pacotes que podem ser instalados e chamados a uma sessão atual conforme necessário

Visão geral do R

- Uma habilidade fundamental para usar R de maneira eficaz é aprender como usar o sistema de ajuda integrado
- Outras seções descrevem o ambiente de trabalho, entrada de programas e saída de resultados, instalação de novas funcionalidades através de pacotes e etc
- Uma característica fundamental do projeto R é que a saída da maioria das funções pode ser usada como entrada para outras funções
- Isso é descrito como reutilização de resultados

Introdução ao R

- Os resultados dos cálculos podem ser armazenados em objetos usando os operadores de atribuição:
  - Uma seta (<-) formada por um caractere menor que e um hífen sem espaço!
  - O caractere igual (=).

Introdução ao R

- Esses objetos podem então ser usados em outros cálculos. Para imprimir o objeto basta digitar o nome do objeto. Existem algumas restrições ao dar um nome a um objeto:
  - Os nomes dos objetos não podem conter símbolos 'estranhos' como!, +, -, #
  - São permitidos um ponto (.) e um sublinhado (_), bem como um nome que comece com um ponto
  - Os nomes dos objetos podem conter um número, mas não podem começar com um número
  - R é *case sensitive* ou seja diferencia maiúsculas de minúsculas, X e x são dois objetos diferentes, assim como temp e temP

Exemplo

```
> x <- c(1:10)
> x[(x>8) | (x<5)]
# 1 2 3 4 9 10
> x <- c(1:10)
> x
# 1 2 3 4 5 6 7 8 9 10
> x > 8
# F F F F F F F F T T
```

```
> x < 5
# T T T T F F F F F F
> x > 8 | x < 5
#T T T T F F F F T T
> x[c(T,T,T,T,F,F,F,F,T,T)]
# 1 2 3 4 9 10
```

Introdução ao R

- Para listar os objetos que você possui em sua sessão R atual, use a função ls ou os objetos de função

```
> ls()
# [1] "x“
```

Introdução ao R

- Se você atribuir um valor a um objeto que já existe, o conteúdo do objeto será substituído pelo novo valor (sem aviso!).
- Use a função rm para remover um ou mais objetos da sua sessão.

```
> rm(x)
> x
# Error: object 'x' not found
```

Introdução ao R

- Vamos criar dois pequenos vetores com dados e um diagrama de dispersão

```
> z2 <- c(1,2,3,4,5,6)
> z3 <- c(6,8,3,5,7,1)
> plot(z2,z3)
> title("Meu primeiro diagrama de dispersao")
```

Atenção!

- Lembre-se sempre: R é case sensitive
- SCOL, Scol e scol são três objetos diferentes

Introdução ao R

```
> x = sin(9)/75
> y = log(x) + x^2
> x
# [1] 0.005494913
Y
# [1] -5.203902
m <- matrix(c(1,2,4,1), ncol=2)
M
# [,1] [,2]
# [1,] 1 4
# [2,] 2 1
```

R Workspace

- Os objetos que você cria durante uma sessão R são mantidos na memória;
- a coleção de objetos que você possui atualmente é chamada de workspace
- Este workspace não é salvo no disco, a menos que você diga ao R para fazer isso
- Isso significa que seus objetos poderão ser perdidos se você fecha o R e não os salva, ou pior, quando o R ou seu sistema trava durante uma sessão

R Workspace

- Ao fechar a janela do RStudio ou do console R, o sistema perguntará se você deseja salvar a imagem do espaço de trabalho
- Se você optar por salvar a imagem do espaço de trabalho, todos os objetos em sua sessão R atual serão salvos em um arquivo .RData.
- Este é um arquivo binário localizado no diretório de trabalho do R, que é por padrão o diretório de instalação do R.

R Workspace

- Durante sua sessão R você também pode salvar explicitamente a imagem do espaço de trabalho
- Vá para o menu `Arquivo' e selecione `Salvar espaço de trabalho...', ou use a função save.image

como salvar no diretório de trabalho atual

> save.image()

comando para verificar qual é o diretório de trabalho atual

> getwd()

como salvar em um arquivo e local específico

- save.image(“path do diretorio")

R Workspace

- Se você salvou uma imagem do espaço de trabalho e iniciar R na próxima vez, ele restaurará o espaço de trabalho
- Assim, todos os seus objetos salvos anteriormente estarão disponíveis novamente
- Você também pode carregar explicitamente um espaço de trabalho salvo, que pode ser a imagem do espaço de trabalho de outra pessoa
- Vá ao menu `Arquivo' e selecione `Carregar espaço de trabalho...'

R Workspace

- Os comandos são inseridos interativamente no prompt do usuário R
- As teclas de seta para cima e para baixo percorrem seu histórico de comandos.
- Você provavelmente desejará manter projetos diferentes em diretórios físicos diferentes

R Workspace

- R fica confuso se você usar um caminho em seu código como:

> "c:\meusdocumentos\meuarquivo.txt"

- Isso ocorre porque R vê "\" como um caractere de escape
- Em vez disso, use

> "c:\\meus documentos\\meuarquivo.txt"

ou

> "c:/meusdocumentos/meuarquivo.txt"

R Workspace

> getwd() # imprime o diretório de trabalho atual

> ls() # lista os objetos no espaço de trabalho atual

> setwd("meudiretório")   # altera do diretório de trabalho para meudiretório

> setwd("c:/docs/meudir")

R Datasets

- R vem com vários datasets de amostra que você pode utilizar:

> data( )

> help(nome do dataset)

- para obter detalhes sobre um dataset específico

Pacotes do R

- Um dos pontos fortes do R é que o sistema pode ser facilmente estendido
- O sistema permite que você escreva novas funções e empacote essas funções em um chamado `pacote R' (ou `biblioteca R’).
- O pacote R também pode conter outros objetos R, por exemplo, conjuntos de dados ou documentação
- Há uma comunidade de usuários R ativa e muitos pacotes R foram escritos e disponibilizados no CRAN para outros usuários
- Apenas alguns exemplos, existem pacotes para otimização de portfólio, desenho de mapas, exportação de objetos para html, análise de séries temporais, estatísticas espaciais e a lista é vasta e cresce diariamente

Pacotes do R

- Quando você baixa o R, vários (cerca de 30) pacotes também são baixados
- Para usar uma função em um pacote R, esse pacote deve estar anexado ao sistema
- Quando você inicia o R, nem todos os pacotes baixados são anexados, apenas sete pacotes são anexados ao sistema por padrão
- Você pode usar a função search() para ver uma lista de pacotes que estão atualmente anexados ao sistema. Essa lista também é chamada de search path

```
> search()
# [1] ".GlobalEnv" "package:stats" "package:graphics“
# [4] "package:grDevices" "package:datasets" "package:utils“
# [7] "package:methods" "Autoloads" "package:base"
```

Pacotes do R

Para anexar outro pacote ao sistema você pode usar o menu ou a função library()

Selecione o menu `Pacotes' e selecione `Carregar pacote...', uma lista de pacotes disponíveis em seu sistema será exibida.

Selecione um e clique em `OK', o pacote agora está anexado à sua sessão R atual.

Através da função de library():

```
> library(MASS)
> shoes
# $A
# [1] 13,2 8,2 10,9 14,3 10,7 6,6 9,5 10,8 8,8 13,3
# $ B
# [1] 14,0 8,8 11,2 14,2 11,8 6,4 9,8 11,3 9,3 13,6
```

Pacotes do R

- A função library também pode ser usada para listar todas as bibliotecas disponíveis em seu sistema com uma breve descrição.
- Execute a função library sem argumentos

> library()

```
Packages in library 'C:/Users/plb2/AppData/Local/Programs/R/R-4.2.0/library':

ade4                    Analysis of Ecological Data: Exploratory and
                        Euclidean Methods in Environmental Sciences
adehabitatHR            Home Range Estimation
adehabitatLT            Analysis of Animal Movements
adehabitatMA            Tools to Deal with Raster Maps
base                    The R Base Package
boot                    Bootstrap Functions (Originally by Angelo
                        Canty for S)
cachem                  Cache R Objects with Automatic Pruning
CircStats               Circular Statistics, from "Topics in Circular
                        Statistics" (2001)
class                   Functions for Classification
classInt                Choose Univariate Class Intervals
cli                     Helpers for Developing Command Line
                        Interfaces
cluster                 "Finding Groups in Data": Cluster Analysis
                        Extended Rousseeuw et al.
```

Pacotes do R

- Como instalar novos pacotes?
- Utilize o comando install.packages(“nome do pacote”)

Reutilizando resultados

- Um dos recursos mais úteis do R é que o resultado das análises pode ser facilmente salvo e usado como entrada para análises adicionais

Exemplo 1

> lm(mpg~wt, data=mtcars)

- Isso executará uma regressão linear simples de milhas por galão no peso do carro usando o dataframe mtcars
- Os resultados são enviados para a tela
- Nada é salvo

Reutilizando resultados

Exemplo 3

> fit <- lm(mpg~wt, data=mtcars)

- Desta vez, a mesma regressão é realizada, mas os resultados são salvos com o nome fit.
- Nenhuma saída é enviada para a tela. No entanto, agora você pode manipular os resultados.

> summary(fit)

apresenta um sumário com os resultados principais da regressão

Reutilizando resultados

plotar resíduos por valores ajustados

> plot(fit$residuals, fit$fitted.values)

- Para ver o que uma função retorna, consulte a seção de valores da ajuda on-line dessa função: help(lm).
- Os resultados também podem ser usados por uma ampla gama de outras funções.

produz gráficos de diagnóstico da regressão

> plot(fit)

Input de dados no R

Tipos de dados no R

- R possui uma ampla variedade de tipos de dados, incluindo escalares, vetores (numéricos, caracteres, lógicos), matrizes, dataframes e listas.

Vetores

Vetores

```
> a <- c(1,2,5.3,6,-2,4) # vetor numérico
> b <- c("um","dois","três") # vetor de caracteres
> c <- c (TRUE, TRUE, TRUE, FALSE, TRUE, FALSE) # vetor lógico
```

Como consultar os elementos de um vetor:

```
> a[c(2,4)] # 2º e 4º elementos do vetor
```

Matrizes

Matrizes

- Todas as colunas de uma matriz devem ter a mesma categoria de dado (numérico, caractere, etc.) e o mesmo comprimento.
- O formato geral de matrizes no R é:
  - minhamatriz <- matriz(vetor, nrow=r, ncol=c, byrow=FALSE,dimnames=list(char_vector_rownames, char_vector_colnames))
  - byrow=TRUE indica que a matriz deve ser preenchida por linhas
  - byrow=FALSE indica que a matriz deve ser preenchida por colunas (o padrão)
  - dimnames fornece rótulos opcionais para colunas e linhas

Matrizes

```
# gera matriz numérica 5 x 4
> y<-matrix(1:20, nrow=5,ncol=4)
# outro exemplo
> celulas <- c(1,26,24,68)
> rnames <- c("R1", "R2")
> cnames <- c("C1", "C2")
> minhamatriz <- matrix(células, nrow =2, ncol=2, byrow=TRUE, dimnames=list(rnames, cnames))
# Como extrair linhas, colunas os elementos de matrizes:
x[,4] # 4ª coluna da matriz
x[3,] # 3ª linha da matriz
x[2:4,1:3] # linhas 2,3,4 das colunas 1,2,3
```

Arrays

Arrays

- Arrays na linguagem de programação R são objetos de dados que podem armazenar dados em mais de duas dimensões
- A matriz 3-D também é conhecida como matriz multidimensional
- Podemos criar um array multidimensional com a função dim()

Arrays

- Exemplo:

```
# Cria dois vetores
> dados1 <- c(1,2,3,4,5,6,7,8,9,10,11,12)
> dados2 <- c(13,14,15,16,17,18,19,20,21,22,23,24)
# passa esses vetores como entrada para o array
# 4 linhas, 2 colunas e 3 dimensões
arrayresultante <- array(c(dados1, dados2), dim = c(4,2,3))
print(arrayresultante)
```

Dataframes

Dataframes

- Um dataframe é mais geral que uma matriz, pois colunas diferentes podem ter tipos de dados diferentes (numérico, caractere, booleanos etc.)

```
> d <-c(1,2,3,4)
> e <- c("vermelho", "branco", “amarelo", “azul”)
> f <- c (TRUE,TRUE,FALSE,FALSE)
meudataframe <- data.frame(d,e,f)
names(meudataframe) <- c("ID","Cor",“Bool") # nomes das variáveis
```

Dataframes

- Há variadas formas de instanciar elementos de um dataframe:

> meudataframe[2:3] # colunas 2 e 3 do dataframe

> meudataframe[c("ID", "Cor")] # colunas ID e Cor do dataframe

> meudataframe$Bool # variável boleana do dataframe

Listas

Listas

- Listas são uma coleção de objetos que podem ser de diferentes tipos
- Uma lista permite reunir uma variedade de objetos (possivelmente não relacionados)

Listas

- Exemplo:

exemplo de uma lista com 4 componentes

uma string, um vetor numérico, uma matriz e um escalar

> lista1 <- list(nome="Fred", meusnumeros=d, matriz = minhamatriz, idade=53)

> lista2 <- list(nome="Artur", meusnumeros=dados1, dataframe = meudataframe, idade=27)

exemplo de uma lista contendo duas listas

> listamultipla <- c(lista1,lista2)

Listas

- Identifique os elementos de uma lista usando a convenção [[]].

> lista1[[2]] # 2º componente da lista

Fatores

Factors

- Diga a R que uma variável é nominal tornando-a um fator
- O fator armazena os valores nominais como um vetor de inteiros no intervalo [1... k] (onde k é o número de valores únicos na variável nominal) e um vetor interno de cadeias de caracteres (os valores originais) mapeados para esses inteiros
- Exemplo:

```
# gênero variável com 20 entradas "masculinas" e  30 entradas "femininas"
> genero <- c(rep("masculino",20), rep("feminino", 30))
> genero <- factor(genero)
# R agora trata gênero como uma variável nominal
summary(gender)
```

Algumas funções úteis

lenght(objeto) # número de elementos ou componentes
str(objeto) # estrutura de um objeto
class(objeto) # classe ou tipo de objeto
names(objeto) # nomes
c(object,objeto,...) # combina objetos em um vetor
cbind(objeto, objeto, ...) # combina objetos como colunas
rbind(objeto, objeto, ...) # combina objetos como linhas
ls() # lista os objetos atuais
rm(objeto) # exclui um objeto
novo_objeto <- edit(objeto) # edita, copia e salva um novo objeto
fix(objeto) # editar no local

Importando dados no R

Importando dados no R

- Importar dados para R não é uma tarefa difícil
- Para dados do Stata, pode-se usar o pacote foreign
- Para SPSS e SAS pode-se usar o pacote Hmisc pela facilidade e funcionalidades

Importando dados de um arquivo csv

```
# a primeira linha contém nomes de variáveis, vírgula é o separador
# atribui o id da variável aos nomes das linhas
# observe o / em vez de \ em sistemas mswindows
dadoscsv <-
read.table("C:/Users/plb2/Pessoal/Doutorado/Periodos/20241/AulaEstagioDocencia/Aula-16062024/mydata.csv", header=TRUE, sep=",")
```

Importando dados de um arquivo Excel

- A melhor maneira de ler um arquivo Excel é exportá-lo para um arquivo csv e importá-lo usando o método acima
- Mas também é possível utilizar o pacote readxl

```
> library(readxl)
> dadosxlsx <- read_excel("C:/Users/plb2/Pessoal/Doutorado/Periodos/20241/AulaEstagioDocencia/Aula-16052024/mydata.xlsx")
```

Exportando dados do R

Exportando dados do R

- Existem vários métodos para exportar objetos R para outros formatos.
- Para SPSS, SAS e Stata. você precisará carregar o pacote foreign
- Para Excel, você precisará do pacote xlsReadWrite

Exportando dados do R

- Para um arquivo de texto delimitado por tabulação

```
> write.table(mydata, "c:/mydata.txt", sep="\t")
```

- Para uma planilha do Excel

```
> library(xlsReadWrite)
> write.xls(mydata, "c:/mydata.xls")
```

Visualizando dados do R

Visualizando dados do R

- Existem várias funções para listar o conteúdo de um objeto ou conjunto de dados.

lista objetos no ambiente de trabalho

> ls()

lista as variáveis em mydata

> names(mydata)

lista a estrutura de mydata

> str(mydata)

dimensões de um objeto

> dim(object)

Visualizando dados no R

- Existem várias funções para listar o conteúdo de um objeto ou conjunto de dados.

```
# classe de um objeto (numérico, matriz, dataframe, etc)
class(dadoscsv)
# imprime dadosxlsx
print(dadosxlsx)
# imprime as primeiras 3 linhas de dadosxlsx
head(dadosxlsx, n=3)
# imprime as últimas 2 linhas de dadoscsv
tail(dadoscsv, n=2)
```

Missing data no R

Missing data no R

- No R, os valores faltantes (missing data) são representados pelo símbolo NA (not available) - não disponível
- Valores impossíveis (por exemplo, divisão por zero) são representados pelo símbolo NaN (Not a Number)

Missing data no R

- Teste de valores ausentes

> is.na(x) # retorna TRUE de x está faltando

> y <- c(1,2,3,NA)

> is.na(y) # retorna um vetor (F F F T)

Missing data no R

- Excluindo valores ausentes das análises
- Funções aritméticas em missing values produzem missing values.

```
> x <- c(1,2,NA,3)
> mean(x) # retorna NA
> mean(x, na.rm=TRUE) # retorna 2
```

Missing data no R

- Função para completar os valores NA com um valor indicado

```
x[is.na(x)] <- 0
```

Datas no R

Datas no R

- As datas são representadas como o número de dias desde 01/01/1970, com valores negativos para datas anteriores

```
# use as.Date() para converter strings em datas
minhasdatas <- as.Date(c("2007/06/22", "2004/02/13"))
# número de dias entre 22/06/07 e 13/02/04
dias <- minhasdatas[1] - minhasdatas[2]
```

- Sys.Date( ) # retorna a data de hoje
- date() # retorna a data e hora atuais

Datas no R

- Os símbolos a seguir podem ser usados com a função format() para imprimir datas

Símbolo	Significado	Exemplo
%d	dia como um número (0-31)	01-31
%a	dia da semana abreviado	Mon
%A	dia da semana não abreviado	Monday
%m	mês (00-12)	00-12
%b	mês abreviado	Jan
%B	mês não abreviado	January
%y	ano com 2 dígitos	07
%Y	ano com 4 dígitos	2007

Datas no R

```
# imprime a data de hoje
> hoje <- Sys.Date()
> format(hoje, format="%B %d %Y") #"maio 16 2024"
```

Manipulando dados com R

Manipulando dados com R

- Depois de ter acesso aos seus dados, você desejará transformá-los em uma forma útil
- Isso inclui a criação de novas variáveis (incluindo recodificação e renomeação de variáveis existentes), classificação e fusão de conjuntos de dados, agregação de dados, remodelagem de dados e subconjuntos de conjuntos de dados (incluindo seleção de observações que atendam aos critérios, amostragem aleatória de observações e eliminação ou manutenção de variáveis).

Manipulando dados com R

- Cada uma dessas atividades geralmente envolve o uso de operadores (aritméticos e lógicos) e funções (numéricos, de caracteres e estatísticos) integrados ao R
- Além disso, você pode precisar usar estruturas de controle (if-then, for, while, switch) em seus programas e/ou criar suas próprias funções
- Finalmente, você pode precisar converter variáveis ou conjuntos de dados de um tipo para outro (por exemplo, numérico para caractere ou matriz para quadro de dados)

Criando novas variáveis

Criando novas variáveis

- Use o operador de atribuição <- para criar novas variáveis
- Uma grande variedade de operadores e funções estão disponíveis aqui

```
> dadoscsv$IMC<-dadoscsv$Peso/(dadoscsv$Altura/100)^2
```

Renomeando variáveis

Renomeando variáveis

- você pode inserir novamente todos os nomes de variáveis para alterar aqueles que você precisa alterar
- A limitação é que você precisa inserir todos os novos nomes!

```
> names(dadoscsv) <- c("Gender","Age","Weight", "Height", "BMI")
```

Operadores aritméticos

Operadores aritméticos

Operador	Descrição
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
^ ou **	Exponenciação
x %% y	x mod y (Exemplo: 5%%2 = 1)
x %/% y	Divisão inteira (Exemplo: 5%/%2 = 2)

Operadores lógicos

Operadores lógicos

Operador	Descrição
<	Menor que
<=	Menor ou igual a
>	Maior que
>=	Maior ou igual a
==	Exatamente igual a
!=	Diferente de
!x	Não x
x | y	x ou y
x & y	x e y
isTRUE(x)	testa se x é verdadeiro

Estruturas de controle

Estruturas de controle

- if-else

> if (cond) expr
> if (cond) expr1 else expr2

- for

> for (var in seq) expr

- While

> while (cond) expr

- ifelse

> ifelse(test,yes,no)

Estruturas de controle

- Exemplo: Transpor uma matriz
- A função padrão do R para transpor matrizes é a t()

```
> transpor <- function(x){
>  if (!is.matrix(x)){
>    warning("argumento não é uma matriz: retornando NA")
>    return(NA_real_)
>  }
>  y <- matrix(1, nrow=ncol(x), ncol=nrow(x))
>  for (i in 1:nrow(x)){
>   for (j in 1:ncol(x)){
>     y[j,i] <- x[i,j]
>    }
>  }
>  return(y)
>}
> z <- matrix(1:10, nrow=5, ncol=2)
> tz <- transpor(z)
```

Funções nativas do R

Funções nativas do R

- Quase tudo em R é feito através de funções
- Aqui estamos nos referindo apenas às funções numéricas e de caracteres que são comumente usadas na criação ou recodificação de variáveis
- Observe que embora os exemplos no slide seguinte apliquem funções a variáveis individuais, muitos também podem ser aplicados a vetores e matrizes

Funções numéricas

Funções numéricas

Função	Descrição
abs(x)	valor absoluto de x
sqrt(x)	raiz quadrada de x
ceiling(x)	teto - ceiling(3.475) é 4
floor(x)	piso – floor(3.475) é 3
trunc(x)	truncar – trunc(5.99) é 5
round(x, digits=n)	arredondar – round(3.475, digits=2) é 3.48
cos(x), sin(x), tan(x)	cosseno, seno e tangente de x
log(x)	logaritmo natural de x
log10(x)	logaritmo na base 10 de x
exp(x)	exponencial de x

Funções estatísticas

Funções estatísticas

- A tabela a seguir descreve funções relacionadas às distribuições de probabilidade
- Para os geradores de números aleatórios abaixo, você pode usar set.seed(1234) ou algum outro número inteiro para criar números pseudo-aleatórios reproduzíveis

Descrição	Descrição
dnorm(x)	função de densidade normal (por padrão m=0 sd=1)
pnorm(q)	probabilidade cumulativa da normal para q
qnorm(p)	quantil da normal
rnorm(n, m=0, sd=1)	n desvios normais aleatórios com média m e desvio padrão sd
dbinom(x, size, prob)	retorna a densidade da distribuição binomial
pbinom(q, size, prob)	retorna a função distribuição da binomial
qbinom(p, size, prob)	retorna a função quantil da distribuição binomial
rbinom(n, size, prob)	gera desvios aleatórios de acordo com a distribuição binomial
dpois(x, lambda)	retorna a densidade da distribuição de poisson
ppois(q, lambda)	retorna a função distribuição de poisson
qpois(p, lambda)	retorna a função quantil da distribuição de poisson
rpois(n, lambda)	gera desvios aleatórios de acordo com a distribuição de poisson
dunif(x, min=0, max=1)	retorna a densidade da distribuição uniforme
punif(q, min=0, max=1)	retorna a função distribuição uniforme
qunif(p, min=0, max=1)	retorna a função quantil da distribuição uniforme
runif(n, min=0, max=1)	retorna a função quantil da distribuição uniforme

Função	Descrição
mean(x)	A média de um objeto x
sd(x)	desvio padrão de um objeto x
var(x)	variância de um objeto x
median(x)	mediana de um objeto x
quantile(x, probs)	quantis onde x é o vetor numérico cujos quantis são desejados e probs é um vetor numérico com probabilidades em [0,1].
sum(x)	soma dos objetos de x
diff(x, lag = 1)	diferenças com lag indicando qual espaçamento usar
min(x)	Valor mínimo do objeto x
max(x)	Valor máximo do objeto x

Ordenação de dados no R

Ordenação de dados no R

- Para classificar um dataframe em R, use a função order()
- Por padrão, a classificação é ASCENDING
- Utilize o sinal de menos para indicar a ordem DESCENDING

ordenando os dados pelas idades

> dadoscsvsorted = dadoscsv[order(dadoscsv$Age),]

Merging no R

Merging no R

- Para mesclar dois dataframes (conjuntos de dados) horizontalmente, use a função merge
- É possível unir dois dataframes por uma ou mais variáveis-chave comuns (ou seja, um INNER JOIN)

```
> novosdados <- read.table("C:/Users/plb2/Pessoal/Doutorado/Periodos/20241/AulaEstagioDocencia/Aula-16052024/newdata.csv", header=TRUE, sep=",")
> dadoscompletos <- merge(dadoscsv,novosdados,by="ID")
```

Adicionando linhas a uma tabela

Adicionando linhas a uma tabela

- Para unir dois dataframes (conjuntos de dados) verticalmente, use a função rbind
- Os dois dataframes devem ter as mesmas variáveis, mas não precisam estar na mesma ordem

```
> write.table(dadoscompletos, file = "C:/Users/plb2/Pessoal/Doutorado/Periodos/20241/AulaEstagioDocencia/Aula-16052024/dadoscompletos.csv", sep = ",")

> novodado <- read.table("C:/Users/plb2/Pessoal/Doutorado/Periodos/20241/AulaEstagioDocencia/Aula-16052024/novodado.csv", header=TRUE, sep=",")

> novodadodadosmaiscompletos<-rbind(dadoscompletos,novodado)
```

Conversão de tipos de dados no R

Conversão de tipos de dados no R

- As conversões de tipo em R costumam funcionar conforme o esperado
- Por exemplo, adicionar uma sequência de caracteres a um vetor numérico converte todos os elementos do vetor em caracteres
- Use is.datatype para testar o tipo de dados. Retorna TRUE ou FALSE
- Use as.datatype para convertê-lo explicitamente

> is.numeric(), is.character(), is.vector(), is.matrix(), is.data.frame()
> as.numeric(), as.character(), as.vector(), as.matrix(), as.data.frame()

Gráficos no R

Gráficos de Pizza ou Pie Charts

Gráficos de Pizza ou Pie Charts

- A linguagem de programação R possui múltiplas bibliotecas para criar tabelas e gráficos
- Um gráfico de pizza é uma representação de valores como fatias de um círculo com cores diferentes.
- As fatias são rotuladas e os números correspondentes para cada fatia também é representado no gráfico
- Em R, o gráfico de pizza é criado usando a função pie() que aceita números positivos como entrada vetorial
- Os parâmetros adicionais são usados para controlar rótulos, cor, título etc.

Gráficos de Pizza ou Pie Charts

- A sintaxe básica para criar um gráfico de pizza usando R é:

  pie(x, labels, radius, main, col, clockwise)

- A seguir está a descrição dos parâmetros utilizados
  - x é um vetor contendo os valores numéricos usados no gráfico de pizza
  - labels são usados para dar descrição às fatias.
  - radius indica o raio do círculo do gráfico de pizza (valor entre −1 e +1).
  - main indica o título do gráfico.
  - col indica a paleta de cores.
  - clockwise é um valor lógico que indica se as fatias são desenhadas em sentido horário ou anti-horário

Gráficos de Pizza ou Pie Charts

- Exemplo:

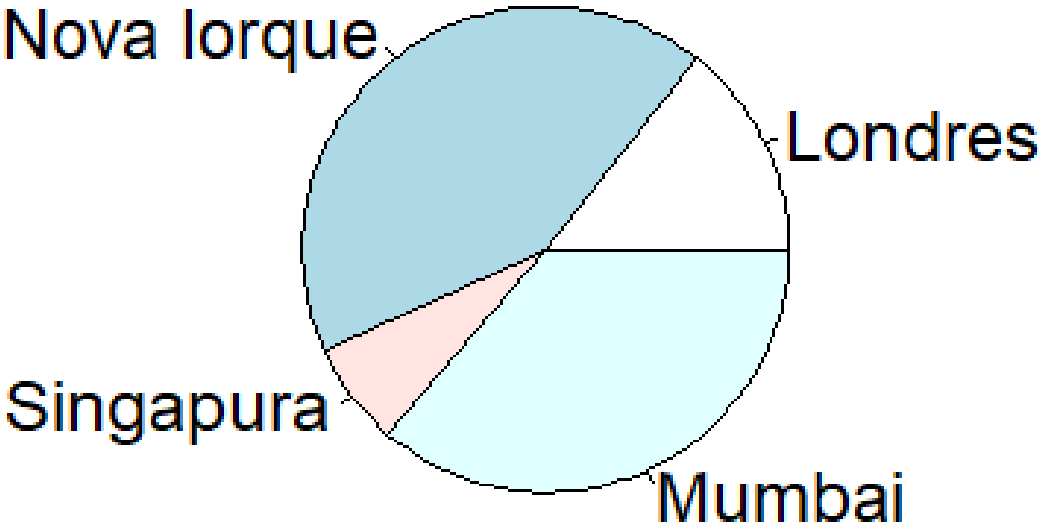
Cria os dados para o gráfico

> x <- c(21, 62, 10, 53)

> labels <- c("Londres", "Nova Iorque", "Singapura","Mumbai")

Plotando o gráfico

> pie(x,labels)



Gráficos de Barras ou Bar Charts

Gráficos de Barras ou Bar Charts

- Um gráfico de barras representa dados em barras retangulares com comprimento da barra proporcional ao valor da variável.
- R usa a função barplot() para criar gráficos de barras
- R pode desenhar barras verticais e horizontais no gráfico de barras
- No gráfico de barras, cada uma das barras pode receber cores diferentes

Gráficos de Barras ou Bar Charts

- A sintaxe básica para criar um gráfico de barras em R é:

barplot(H, xlab, ylab, main, names.arg, col)

- A seguir está a descrição dos parâmetros utilizados na função:
  - H é um vetor ou matriz contendo valores numéricos usados em gráfico de barras.
  - xlab é o rótulo do eixo x.
  - ylab é o rótulo do eixo y.
  - main é o título do gráfico de barras.
  - names.arg é um vetor de nomes que aparecem em cada barra
  - col é usado para dar cores às barras do gráfico

Gráficos de Barras ou Bar Chart

- Exemplo

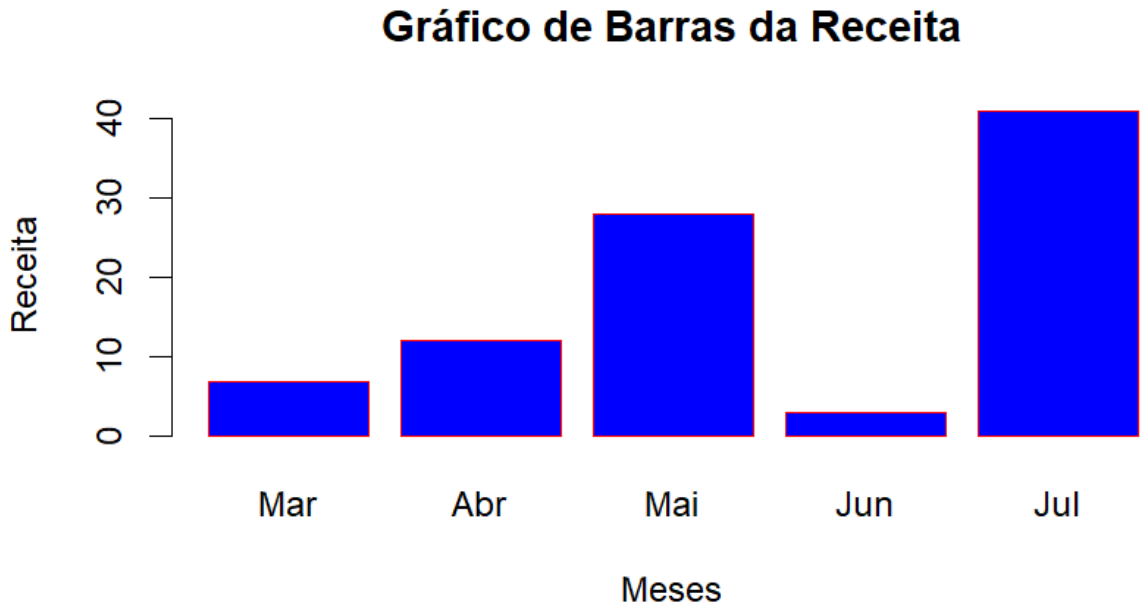
Criando os dados para o gráfico

> H <- c(7,12,28,3,41)

> M <- c("Mar","Abr","Mai","Jun","Jul")

Plotando o gráfico de barras

> barplot(H, names.arg = M , xlab = "Meses", ylab = "Receita", col = "blue", main = "Gráfico de Barras da Receita", border = "red")



Boxplot

Boxplot

- Boxplots são uma medida de quão bem distribuído está os dados em um conjunto de dados
- Ele divide o conjunto de dados em três quartis
- Este gráfico apresenta o mínimo, máximo, mediana, primeiro quartil e terceiro quartil do conjunto dos dados
- Também é útil para comparar a distribuição de dados entre conjuntos de dados, desenhando boxplots para cada um deles
- Boxplots são criados em R usando a função boxplot()

Boxplot

A sintaxe básica para criar um boxplot em R é:

boxplot(x, data, notch, varwidth, names, main)

A seguir está a descrição dos parâmetros utilizados:

– x é um vetor ou uma fórmula

– data é o dataframe

– notch é um valor lógico. Defina como TRUE para desenhar um notch.

– varwidth é um valor lógico. Defina como verdadeiro para desenhar a largura do caixa proporcional ao tamanho da amostra.

– names são os rótulos dos grupos que serão impressos sob cada boxplot

– main é usado para dar um título ao gráfico.

Boxplot

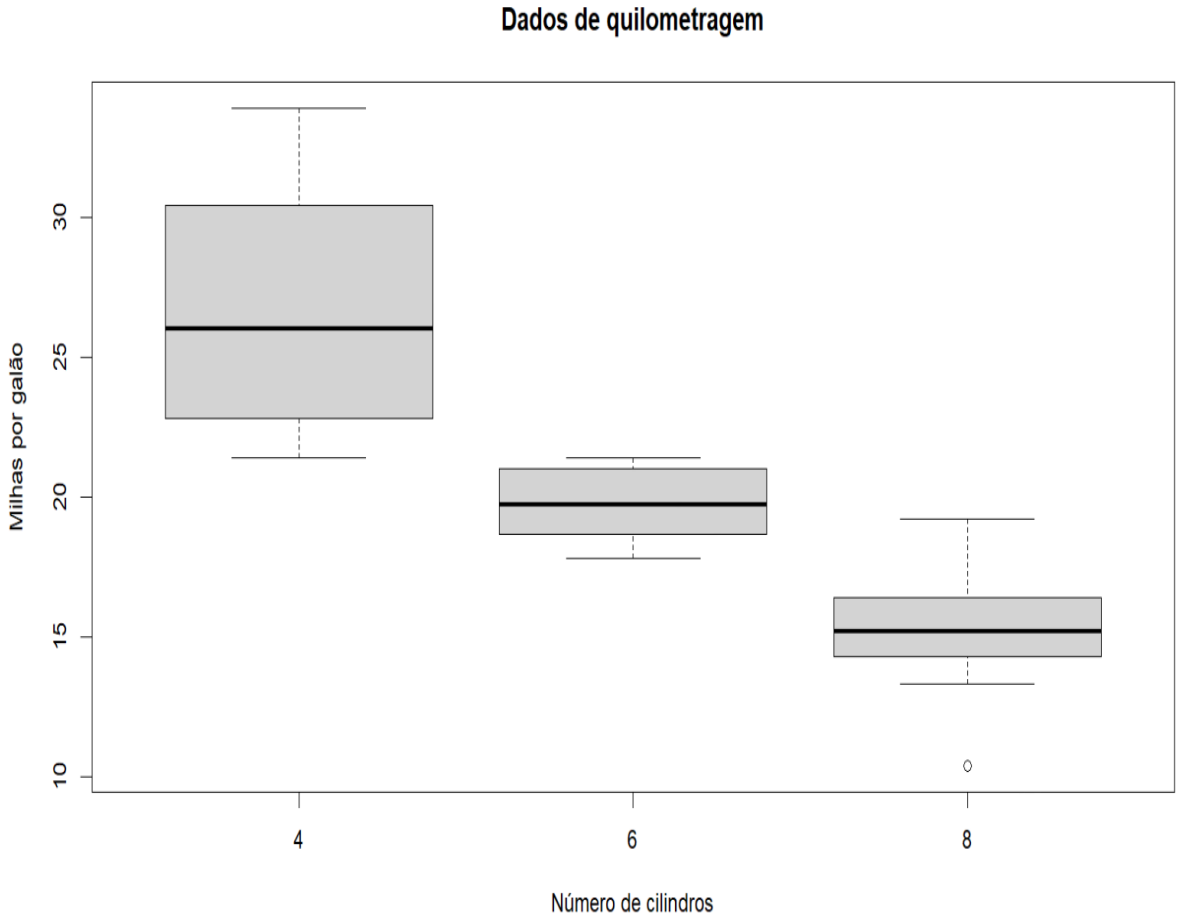
- Exemplo: Usamos o conjunto de dados “mtcars” disponível no R ambiente para criar um boxplot. Vamos utilizar as colunas "mpg" e "cyl" de mtcars.

```
> input <- mtcars[,c('mpg','cyl’)]
> print(head(input))
```

```
                   mpg cyl
Mazda RX4         21.0   6
Mazda RX4 Wag     21.0   6
Datsun 710        22.8   4
Hornet 4 Drive    21.4   6
Hornet Sportabout 18.7   8
Valiant           18.1   6
```

Boxplot

```
> input <- mtcars[,c('mpg','cyl’)]
> print(head(input))
# Plotando o boxplot
> boxplot(mpg ~ cyl, data = mtcars,
xlab = "Número de cilindros", ylab =
"Milhas por galão", main = "Dados de
quilometragem")
```



Histograma

Histograma

- Um histograma representa as frequências dos valores de uma variável agrupados em intervalos
- O histograma é semelhante ao gráfico de barras, mas a diferença é que agrupa os valores em intervalos contínuos
- Cada barra no histograma representa a altura do número de valores presentes naquele intervalo
- O R pode criar histogramas usando a função hist()
- Esta função recebe um vetor como entrada e usa mais alguns parâmetros para traçar histogramas

Histograma

A sintaxe básica para criar um histograma usando R é:

hist(v,main,xlab,xlim,ylim,breaks,col,border)

A seguir está a descrição dos parâmetros utilizados:

- v é o vetor contendo valores numéricos usados no histograma
- main indica o título do gráfico
- xlab é usado para fornecer uma descrição do eixo x
- xlim é usado para especificar o intervalo de valores no eixo x
- ylim é usado para especificar o intervalo de valores no eixo y
- breaks é usado para mencionar a largura de cada barra
- col é usado para definir a cor das barras
- border é usado para definir a cor da borda de cada barra

Histograma

Exemplo:

```
> x <- rnorm(10000, m=0, sd=1)
> hist(x)
```

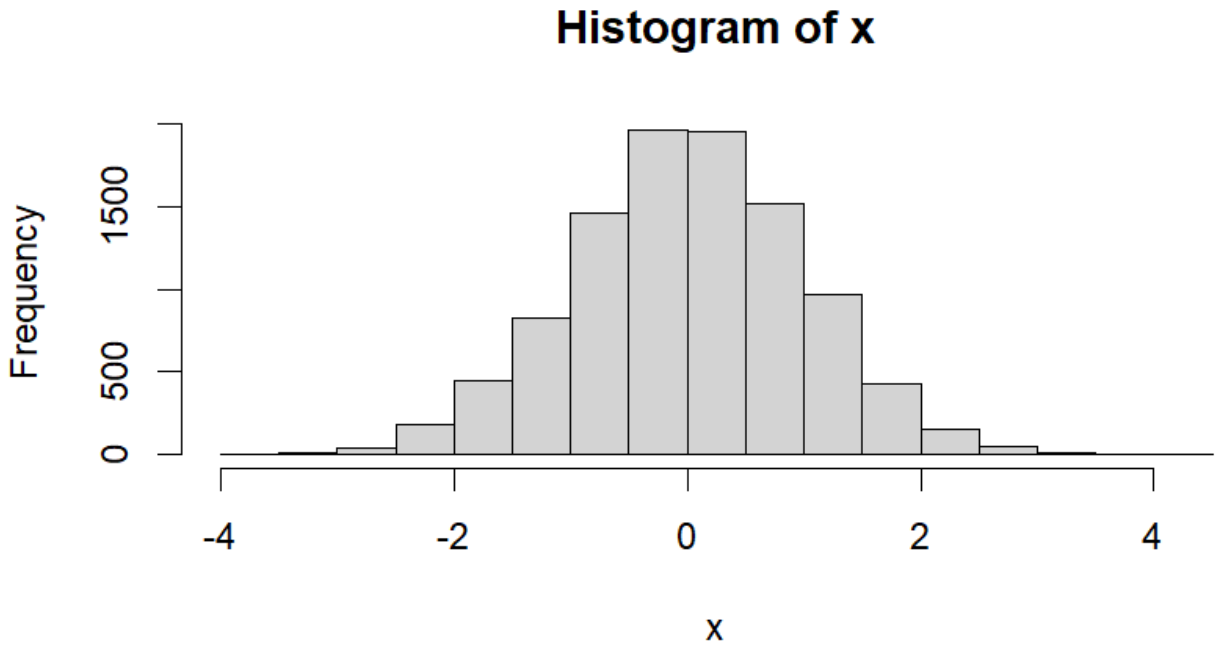


Gráfico de Linha

Gráfico de linha

- Um gráfico de linhas é um gráfico que conecta uma série de pontos desenhando segmentos de linha entre eles
- Esses pontos são ordenados em um de seus valores de coordenadas (geralmente a coordenada x)
- Gráficos de linhas são geralmente usados para identificar tendências nos dados
- A função plot() em R é usada para criar o gráfico de linha

Gráfico de linha

- A sintaxe básica para criar um gráfico de linhas em R é:

plot(v,type,col,xlab,ylab)

- A seguir está a descrição dos parâmetros usados
  - v é um vetor contendo os valores numéricos.
  - type assume o valor "p" para desenhar apenas os pontos, "i" para desenhar apenas as linhas e "o" para desenhar ambos pontos e linhas
  - xlab é o rótulo do eixo x.
  - ylab é o rótulo do eixo y.
  - main é o título do gráfico.
  - col é usado para dar cores aos pontos e às linhas.

Gráfico de linha

```
# Criando os dados para o gráfico
> v <- c(7,12,28,3,41)
# Plotando o gráfico de linha
> plot(v,type = "o", col = "red", xlab = "Mês", ylab = "Quantidade de chuva", main = "Gráfico da quantidade de chuva")
```

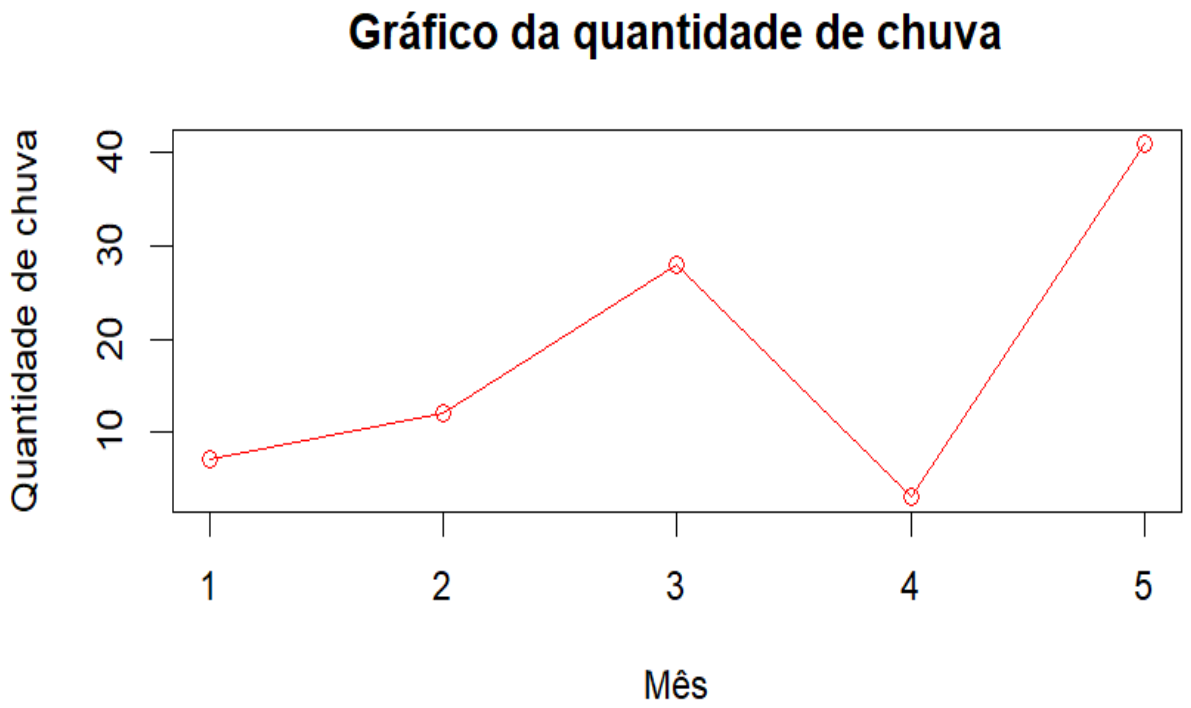


Gráfico de linha com múltiplas linhas

Gráfico de linha com múltiplas linhas

```
# Gráfico de linha com múltiplas linhas
> t <- c(14,7,6,19,3)
> lines(t, type = "o", col = "blue")
```

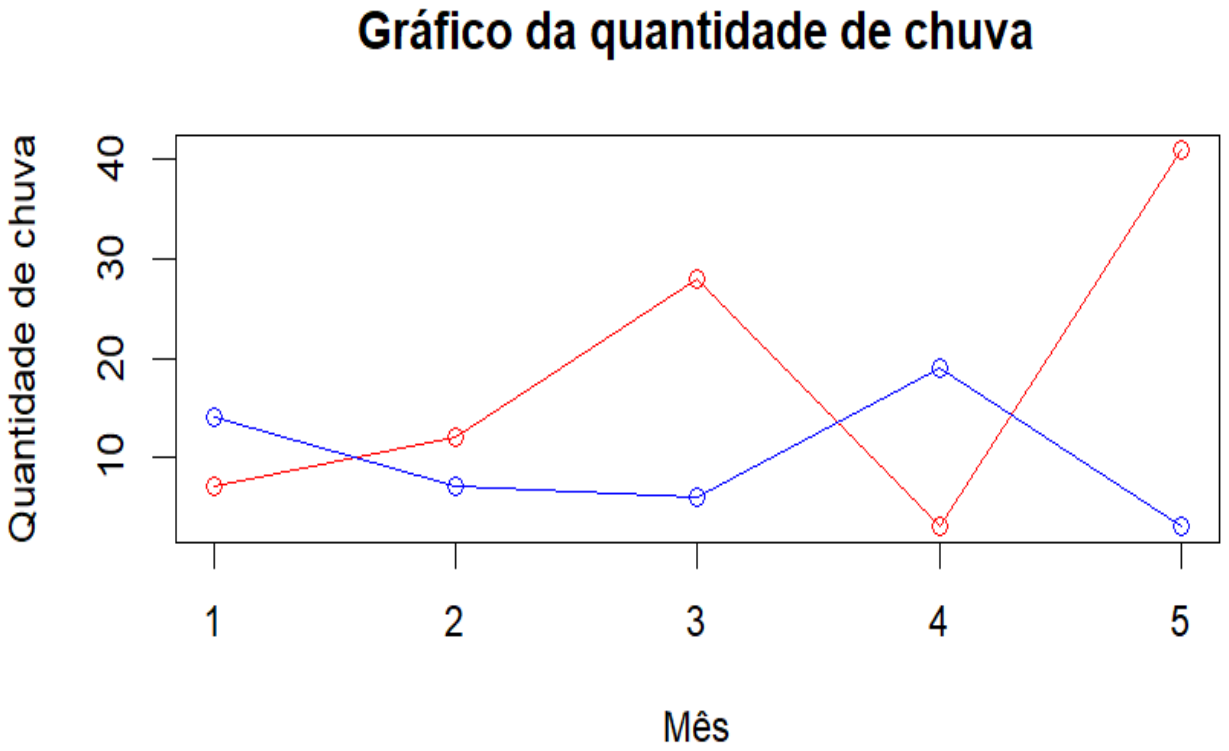


Diagrama de dispersão ou Scatter Plot

Diagrama de dispersão - Scatter Plot

- Os diagrama de dispersão mostram pontos plotados no plano cartesiano
- Cada ponto representa os valores de duas variáveis
- Uma variável é escolhida no eixo horizontal e outra no eixo vertical
- O diagrama de dispersão simples é criado usando também a função plot()

Diagramas de dispersão – Scatter Plot

- Usaremos novamente o conjunto de dados “mtcars” disponível no ambiente do R para criar um gráfico de dispersão
- Vamos usar as colunas "wt" e "mpg" em mtcars

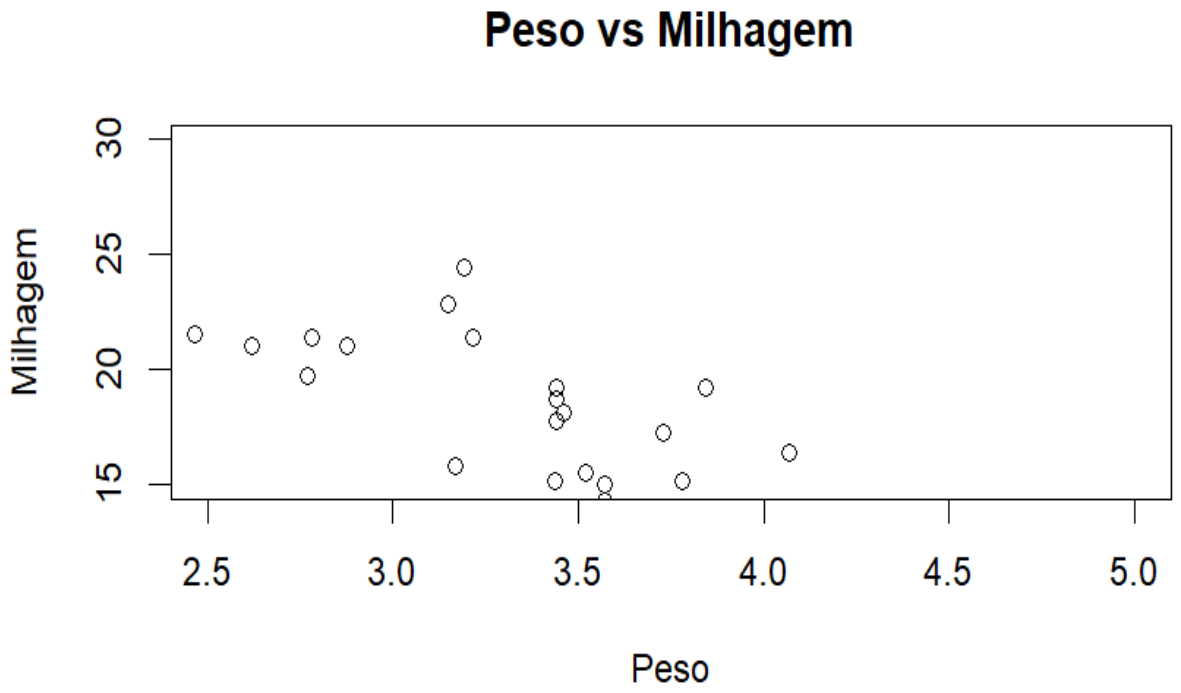
```
> input <- mtcars[,c('wt','mpg’)]
> print(head(input))
```

```
                   wt  mpg
Mazda RX4         2.620 21.0
Mazda RX4 Wag     2.875 21.0
Datsun 710        2.320 22.8
Hornet 4 Drive    3.215 21.4
Hornet Sportabout 3.440 18.7
Valiant           3.460 18.1
```

Diagrama de dispersão – Scatter Plot

Esboçando o diagrama de dispersão para carros com peso entre 2,5 a 5 e quilometragem entre 15 e 30

```
> plot(x = input$wt,y = input$mpg, xlab = "Peso",     ylab = "Milhagem",     xlim = c(2.5,5), ylim = c(15,30),     main = "Peso vs Milhagem")
```



Matriz de diagramas de dispersão

Matriz de diagramas de dispersão

- Quando temos mais de duas variáveis e queremos encontrar a correlação entre uma variável versus o restantes usamos a matriz do gráfico de dispersão
- Usaremos a função pairs() para criar matrizes de gráficos de dispersão.

– Sintaxe:

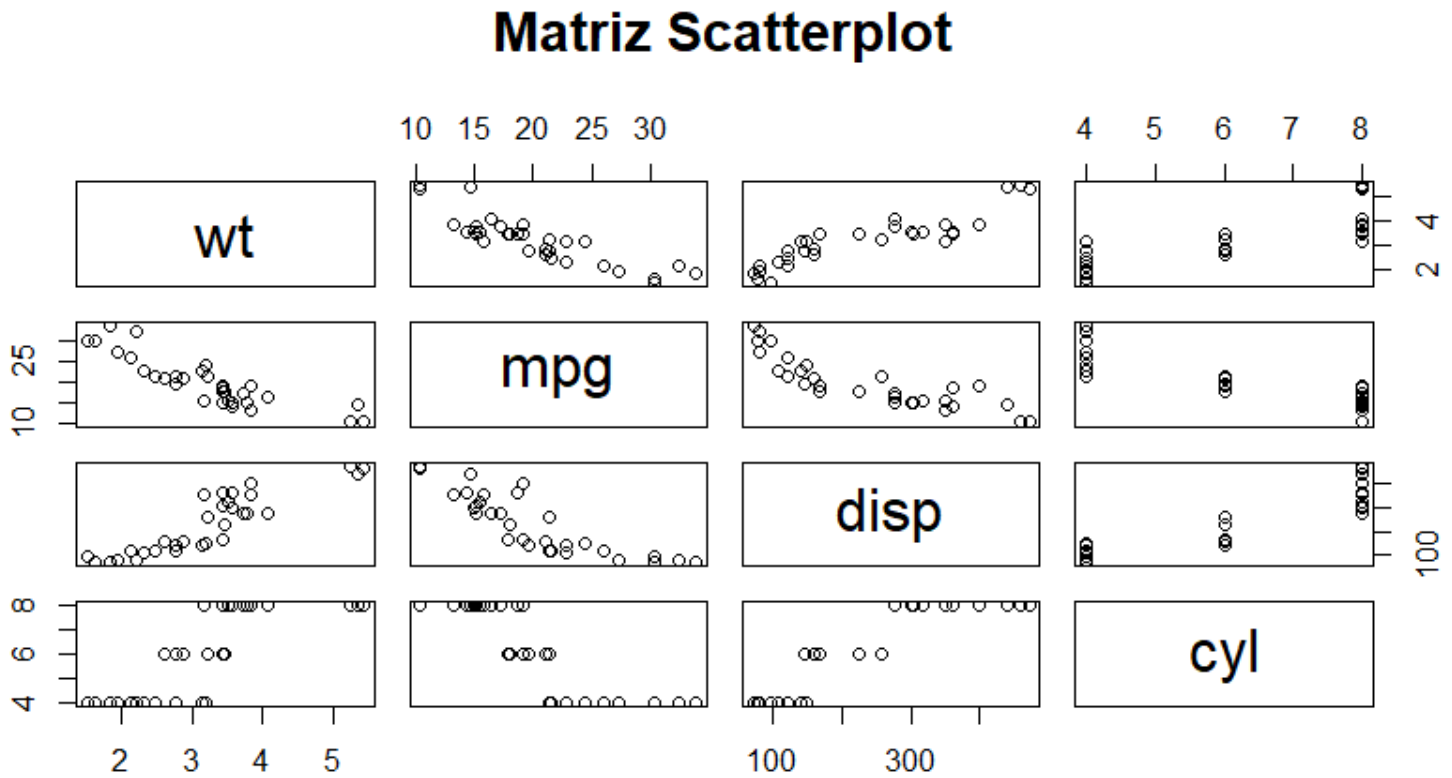
pairs(formula, data)

A seguir está a descrição dos parâmetros utilizados:

- formula: representa a série de variáveis usadas em pares
- data: representa o conjunto de dados do qual as variáveis serão utilizadas

Matriz de diagramas de dispersão

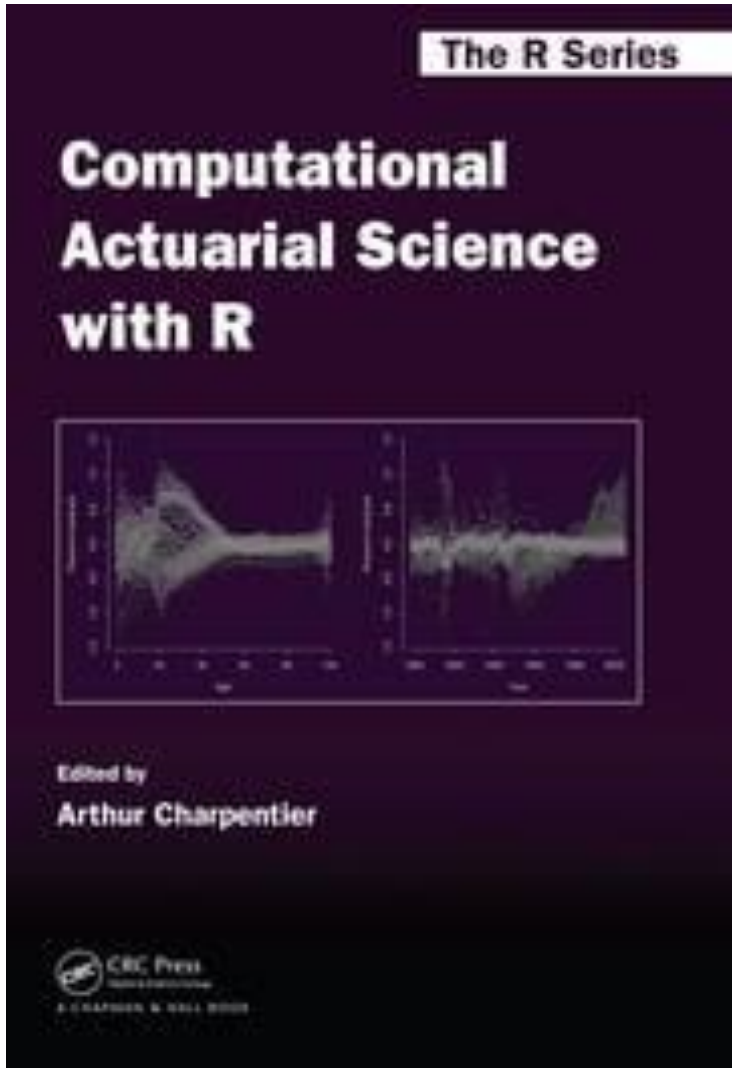
```
> pairs(~wt+mpg+disp+cyl,data = mtcars,main = "Matriz Scatterplot")
```



Bibliografia utilizada para essa aula

Algumas leituras interessantes...

Livro de R e aplicações das Ciências Atuariais



Obrigado!