

Introdução ao R

Aprendizagem Estatística em Altas Dimensões

Monitora: Anny K. G. Rodrigues

IME-USP

2020-09-04

O que é o R?

"R é um ambiente de software livre para computação estatística e gráficos" (<https://www.r-project.org/>).

Algumas Vantagens

1. Open Source
2. Vasta comunidade de usuários
3. Cross Platform Compatible (Unix/Linux, Mac e Windows)
4. Integração com outras linguagens (Python, C, SQL etc.)



O que é possível fazer com o R?



1. Análise de dados - Estatística, modelagem, etc.
2. Visualização de dados
3. Relatórios dinâmicos
4. Apresentações
5. Mineração de dados
6. E muito mais ...

Primeiros Passos

Instalação do R + Rstudio

A versão mais recente da linguagem R pode ser baixada em <https://www.r-project.org/>:

- Download R for Linux
- Download R for MAC
- Download R for Windows

RStudio

A plataforma Rstudio foi desenhada para rodar códigos da linguagem R. Você pode baixar em [RStudio IDE](#).

Arquivos

O R gera alguns tipos de arquivos que podem ser salvos. As principais deles são:

- **.R** - Usado para salvar códigos criados e rotinas de análises (scripts).
- **.RData** - Usado para salvar os objetos da área de trabalho (workspace).
- **.Rhistory** - Usado para salvar o histórico dos comandos executados (normalmente salvo automaticamente).
- **.Rproj** - Formato exclusivo do RStudio, serve para salvar todas as informações utilizadas anteriormente pelo RStudio de maneira simplificada.

Área de trabalho e diretório de trabalho

- `getwd()` # Mostrar o diretório de trabalho atual
- `dir()` # Listar os arquivos do diretório
- `setwd()` # Mudar o diretório de trabalho

Help!

- Pedir ajuda: **help(nome_da_função)** ou **?(nome_da_função)**

```
?mean()
```

Pacotes no R

1. Instalar

- Via CRAN

```
install.packages("tidyverse")
```

- Via GitHub

```
devtools:::install_github("tidyverse/purr")
```

2. Carregar

- library(nome_do_pacote)

```
library(tidyverse)
```

- require(nome_do_pacote)

```
require(tidyverse)
```

Atalhos

- CTRL + ENTER: roda a linha selecionada no script.
- ALT + -: (<-) sinal de atribuição.

Sintaxe Básica dos Comandos

```
2+2 # Adição
```

```
## [1] 4
```

```
3-1 # subtração
```

```
## [1] 2
```

```
4/2 # divisão
```

```
## [1] 2
```

```
5*3 # multiplicação
```

```
## [1] 15
```

Sintaxe Básica dos Comandos

```
9 %% 5    # resto da divisão de 9 por 5
```

```
## [1] 4
```

```
7 %/% 4    # parte inteira da divisão de 7 por 4
```

```
## [1] 1
```

```
4^ 2      # potenciação
```

```
## [1] 16
```

```
sqrt(16) # radiciação
```

```
## [1] 4
```

- O sinal `>` indica que o R está pronto para receber um comando.
- O sinal `+` é o estado de espera, indica que o usuário enviou um comando incompleto.

Atribuição de Objetos no R

Os objetos dentro do R podem ser salvos temporariamente na área de trabalho (workspace). Para atribuir um objeto utilizamos:

O atalho ALT + – que gera o operador < –, ou o símbolo de igual (=)

- **nome_do_objeto** < – valor
- **nome_do_objeto** = valor

```
x <- 15  
x
```

```
## [1] 15
```

```
disciplina <- "Aprendizagem Estatística"  
disciplina
```

```
## [1] "Aprendizagem Estatística"
```

```
Saldo <- TRUE  
Saldo
```

```
## [1] TRUE
```

Tipos de objetos e Estruturas de Dados

- Character: texto

```
nome <- "Ana"  
nome # sempre com aspas
```

```
## [1] "Ana"
```

```
class(nome)
```

```
## [1] "character"
```

- Numeric: números racionais ou inteiros

```
y <- 4.5  
y
```

```
## [1] 4.5
```

```
class(y)
```

```
## [1] "numeric"
```

Tipos de objetos e Estruturas de Dados

- Logical: TRUE, FALSE ou NA

```
ordem <- TRUE  
ordem
```

```
## [1] TRUE
```

```
class(TRUE)
```

```
## [1] "logical"
```

- Complex: números complexos

```
numero <- 4+5i  
numero
```

```
## [1] 4+5i
```

```
class(numero)
```

```
## [1] "complex"
```

Tipos de objetos e Estruturas de Dados

- Factor: variáveis categóricas

```
Ordem <- c("Compra", "Venda")
Ordem <- as.factor(Ordem)
Ordem
```

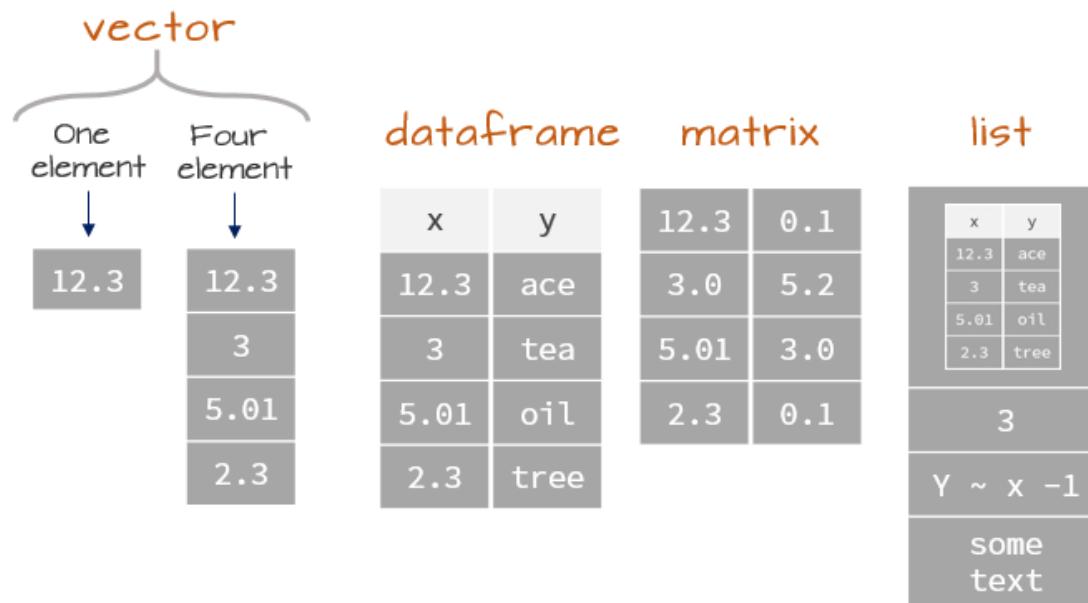
```
## [1] Compra Venda
## Levels: Compra Venda
```

A função `as.factor()` transforma a variável `Ordem` em um fator.

Levels: Refere-se aos rótulos da variável `Ordem`.



O pacote `forcats` é ideal para trabalhar com este tipo de variável.



Fonte: **Manny Gimond (2020)**

- **Vetor**: Armazena elementos de mesmo tipo e/ ou classe. Um vetor pode ser criado usando a função `c()`.
- **Matriz**: Matrizes em R podem ser pensadas como vetores indexados usando dois índices em vez de um. São estruturas bidimensionais.
- **Data Frame**: São tabelas de dados com linhas e colunas, como uma tabela do Excel. Pode conter diferentes tipos de dados (numérico, fator, ...).
- **Listas**: Uma lista é um conjunto ordenado de componentes armazenados em um vetor 1D.

Tipos de objetos e Estruturas de Dados

```
vetor <- c(1.2,2.5,3.7,4,5)
```

length(vetor) # retorna o tamanho do vetor

```
## [1] 5
```

vetor[3] # acessando o elemento da terceira posição do meu vetor

```
## [1] 3.7
```

```
matriz <- matrix(1:12, nrow = 3, ncol = 4, byrow = TRUE)
```

```
matriz
```

```
##      [,1] [,2] [,3] [,4]
```

```
## [1,]    1    2    3    4
```

```
## [2,]    5    6    7    8
```

```
## [3,]    9   10   11   12
```

dim(matriz) # retorna a dimensão da matriz

```
## [1] 3 4
```

Tipos de objetos e Estruturas de Dados

```
matrix[1,2] # acessa o elemento da linha 1 coluna 2
```

```
## [1] 2
```

```
matrix[1,] # acessa todos os elementos da linha 1
```

```
## [1] 1 2 3 4
```

```
matrix[,2] # acessa todos os elementos da coluna 2
```

```
## [1] 2 6 10
```

```
dados <- data.frame(ano = 2000:2004,  
                     prod = c(32, 54, 25, 48, 29))  
dados
```

```
##      ano prod  
## 1 2000   32  
## 2 2001   54  
## 3 2002   25  
## 4 2003   48
```

Tipos de objetos e Estruturas de Dados

```
dim(dados) # dimensões do data frame
```

```
## [1] 5 2
```

```
str(dados) # tipos de variáveis no data frame
```

```
## 'data.frame':    5 obs. of  2 variables:  
##   $ ano : int  2000 2001 2002 2003 2004  
##   $ prod: num  32 54 25 48 29
```

```
dados[,1]
```

```
## [1] 2000 2001 2002 2003 2004
```

```
dados[,2]
```

```
## [1] 32 54 25 48 29
```

Tipos de objetos e Estruturas de Dados

```
minha_lista <- list() # cria uma lista vazia  
ls() # lista todos os objetos
```

```
## [1] "dados"          "disciplina"    "matriz"        "minha_lista" "nome"  
## [6] "numero"         "ordem"        "Ordem"        "Saldo"       "vetor"  
## [11] "x"              "y"
```

```
minha_lista[[1]] <- dados  
minha_lista[[2]] <- matriz  
minha_lista[[3]] <- vetor
```

Tipos de objetos e Estruturas de Dados

```
minha_lista
```

```
## [[1]]
##     ano prod
## 1 2000    32
## 2 2001    54
## 3 2002    25
## 4 2003    48
## 5 2004    29
##
## [[2]]
##     [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
##
## [[3]]
## [1] 1.2 2.5 3.7 4.0 5.0
```

```
minha_lista[[3]] # acessando o vetor na lista
```

```
## [1] 1.2 2.5 3.7 4.0 5.0
```

Um pouco mais sobre data frames

- head(): Mostra o cabeçalho do data frame.
- tail(): Mostra as últimas linhas do data frame.
- names(): Apresenta os nomes das variáveis.
- View(): Mostra o dataframe.

Operadores Relacionais

- Igual a: ==
- Diferente de: !=
- Maior que: >
- Maior ou igual: >=
- Menor que: <
- Menor ou igual: <=

Operadores Lógicos

- E: & Será verdadeiro se os dois forem TRUE

```
x <- 10  
x>5 & x>7
```

```
## [1] TRUE
```

- OU: | Será verdadeiro se um dos dois forem TRUE

```
y <- 8  
y>7 | y<3
```

```
## [1] TRUE
```

- Negação: !

```
z <- 3  
(!z>3)
```

```
## [1] TRUE
```

Estrutura Condisional: IF e ELSE

```
situacao <- "Aprovado"
if(situacao=="Aprovado"){
  print("Parabéns!")
}
```

```
## [1] "Parabéns!"
```

```
situacao1 <- "Reprovado"
if(situacao1=="Aprovado"){
  print("Parabéns")
}else{
  print("Tente outra vez!")
}
```

```
## [1] "Tente outra vez!"
```

```
ordem <- c(0,1,0,1,0,1) # 0-venda 1- pra compra
ordem <- ifelse(ordem==1, "Compra", "Venda")
print(ordem)
```

```
## [1] "Venda"   "Compra"  "Venda"   "Compra"  "Venda"   "Compra"
```

Estrutura de Repetição: FOR

```
df <- iris  
  
head(df) # possui 6 linhas e 5 colunas
```

```
##   Sepal.Length Sepal.Width  Petal.Length Petal.Width Species  
## 1         5.1       3.5        1.4       0.2  setosa  
## 2         4.9       3.0        1.4       0.2  setosa  
## 3         4.7       3.2        1.3       0.2  setosa  
## 4         4.6       3.1        1.5       0.2  setosa  
## 5         5.0       3.6        1.4       0.2  setosa  
## 6         5.4       3.9        1.7       0.4  setosa
```

```
especies <- as.character(df$Species)  
for(i in 1:6){  
  print(especies[i])  
}
```

```
## [1] "setosa"  
## [1] "setosa"
```

Manipulação de Dados

Tidyverse

É uma coleção de pacotes R voltados para a ciência de dados. Todos os pacotes compartilham uma mesma filosofia de desenvolvimento, sintaxe e estruturas de dados.



Principais pacotes do Tidyverse

ggplot2

- **ggplot2**: Destinado para criação de gráficos.
- **dplyr**: Manipulação de dados.
- **tidyverse**: Reorganiza os dados
- **readr**: Leitura dos dados
- **purrr**: Ferramentas para programação funcional, trabalha com funções e vetores.
- **tibble**: Recriação moderna do data frame.
- **magrittr**: Facilita a escrita e leitura do código
- **stringr**: Manipulação strings.
- **forcats**: Manipulação com fatores.



O Pipe %>%

O objetivo do pipe é ajudá-lo a escrever código de uma maneira que seja mais fácil de ler e entender.

```
library(tidyverse)  
  
notas <- c(9.555,10,9.889)  
  
media <- round(mean(notas),2)  
  
media2 <- notas %>% mean() %>% round(2) # com pipe
```

Importação de arquivos

- `read.table()`

Pacote `readr`: funções para ler arquivos texto

- `read_csv`
- `read_csv2`

Pacote `readxl`: função para ler arquivo Excel

- `read_excel`

dplyr: Principais verbos

A ideia do pacote **dplyr** é tornar a manipulação de dados explícita.

- **filter()**: seleciona linhas
- **arrange()**: ordena de acordo com uma ou mais colunas
- **select()**: seleciona colunas
- **mutate()**: cria/modifica colunas
- **summarise()**: sumariza/agrega colunas
- **group_by()**: agrupa colunas

filter

```
data("starwars")
dados <- starwars

dados %>% filter(species == "Human" & eye_color == "blue")
```

A tibble: 12 x 13

	name	height	mass	hair_color	skin_color	eye_color	birth_year	gender
	<chr>	<int>	<dbl>	<chr>	<chr>	<chr>	<dbl>	<chr>
## 1	Luke...	172	77	blond	fair	blue	19	male
## 2	Owen...	178	120	brown, gr...	light	blue	52	male
## 3	Beru...	165	75	brown	light	blue	47	female
## 4	Anak...	188	84	blond	fair	blue	41.9	male
## 5	Wilh...	180	NA	auburn, g...	fair	blue	64	male
## 6	Jek ...	180	110	brown	fair	blue	NA	male
## 7	Lobot	175	79	none	light	blue	37	male
## 8	Mon ...	150	NA	auburn	fair	blue	48	female
## 9	Qui-...	193	89	brown	fair	blue	92	male
## 10	Fini...	170	NA	blond	fair	blue	91	male
## 11	Cifie...	183	NA	brown	fair	blue	82	male
## 12	Joc...	167	NA	white	fair	blue	NA	female
## #	... with 5 more variables: homeworld <chr>, species <chr>, films <list>, vehicles <list>, starships <list>							

arrange

Ordenar as linhas da base de dados conforme uma ou mais variáveis.

```
# Ordenando os personagens pela altura em cm  
dados %>% arrange(height) # default ordem crescente
```

```
## # A tibble: 87 x 13  
##   name    height  mass hair_color skin_color eye_color birth_year gender  
##   <chr>     <int> <dbl> <chr>       <chr>       <chr>        <dbl> <chr>  
## 1 Yoda      66     17  white       green       brown         896 male  
## 2 Ratt...    79     15  none        grey, blue  unknown       NA male  
## 3 Wick...    88     20  brown       brown       brown          8 male  
## 4 Dud ...   94     45  none        blue, grey  yellow       NA male  
## 5 R2-D2     96     32  <NA>       white, bl... red          33 <NA>  
## 6 R4-P...    96     NA  none        silver, r... red, blue    NA female  
## 7 R5-D4     97     32  <NA>       white, red  red           NA <NA>  
## 8 Sebu...    112    40  none        grey, red  orange       NA male  
## 9 Gasg...    122    NA  none        white, bl... black       NA male  
## 10 Watto    137    NA  black       blue, grey yellow      NA male  
## # ... with 77 more rows, and 5 more variables: homeworld <chr>, species <ch  
## #   films <list>, vehicles <list>, starships <list>
```

filter e arrange

```
dados %>% filter(eye_color=="yellow") %>% arrange(mass)
```

```
## # A tibble: 11 x 13
##   name    height  mass hair_color skin_color eye_color birth_year gender
##   <chr>    <int> <dbl> <chr>       <chr>       <chr>        <dbl> <chr>
## 1 Dud ...     94    45 none      blue, grey yellow        NA male
## 2 Zam ...    168    55 blonde    fair, gre... yellow        NA female
## 3 C-3PO     167    75 <NA>      gold       yellow      112 <NA>
## 4 Palp...    170    75 grey      pale       yellow      82 male
## 5 Dart...    175    80 none      red        yellow      54 male
## 6 Pogg...    183    80 none      green      yellow      NA male
## 7 Ki-A...    198    82 white    pale       yellow      92 male
## 8 Dext...    198   102 none      brown      yellow      NA male
## 9 Dart...    202   136 none      white      yellow     41.9 male
## 10 Watto    137     NA black    blue, grey yellow      NA male
## 11 Yara...    264     NA none      white      yellow      NA male
## # ... with 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

```
#Dud Bolt
```

select

Seleciona as colunas na base de dados.

```
dados %>% select(name,species,homeworld)
```

```
## # A tibble: 87 x 3
##   name           species homeworld
##   <chr>          <chr>    <chr>
## 1 Luke Skywalker Human    Tatooine
## 2 C-3PO            Droid    Tatooine
## 3 R2-D2            Droid    Naboo
## 4 Darth Vader     Human    Tatooine
## 5 Leia Organa     Human    Alderaan
## 6 Owen Lars       Human    Tatooine
## 7 Beru Whitesun lars Human    Tatooine
## 8 R5-D4            Droid    Tatooine
## 9 Biggs Darklighter Human    Tatooine
## 10 Obi-Wan Kenobi Human   Stewjon
## # ... with 77 more rows
```

Funções úteis do select

- **starts_with("h")**: seleciona colunas que começam com "h"

```
dados %>% select(starts_with("h")) %>% names()
```

```
## [1] "height"      "hair_color"   "homeworld"
```

- **ends_with("or")**: seleciona colunas que terminam com "or"

```
dados %>% select(ends_with("or")) %>% names()
```

```
## [1] "hair_color"  "skin_color"  "eye_color"
```

- **contains("w")**: seleciona colunas que contêm "w"

```
dados %>% select(contains("w")) %>% names()
```

```
## [1] "homeworld"
```

mutate

Cria ou modifica uma coluna na base de dados.

```
# criando uma nova variável altura em metros  
dados %>% select(name,height) %>%  
  mutate(height_m=height/100)
```

```
## # A tibble: 87 x 3  
##   name           height heighth_m  
##   <chr>          <int>    <dbl>  
## 1 Luke Skywalker     172     1.72  
## 2 C-3PO              167     1.67  
## 3 R2-D2                96     0.96  
## 4 Darth Vader        202     2.02  
## 5 Leia Organa         150     1.5  
## 6 Owen Lars            178     1.78  
## 7 Beru Whitesun lars   165     1.65  
## 8 R5-D4                97     0.97  
## 9 Biggs Darklighter    183     1.83  
## 10 Obi-Wan Kenobi      182     1.82  
## # ... with 77 more rows
```

summarise

Resume os valores das colunas em um só valor, de acordo com funções como média, mediana, min, máx etc.

```
dados %>% summarise(mean(mass, na.rm = TRUE))  
dados %>% summarise(max(mass, na.rm = TRUE))  
dados %>% summarise(min(mass, na.rm = TRUE))
```

```
## # A tibble: 1 x 1  
##   `mean(mass, na.rm = TRUE)`  
##                 <dbl>  
## 1                  97.3  
## # A tibble: 1 x 1  
##   `max(mass, na.rm = TRUE)`  
##                 <dbl>  
## 1                1358  
## # A tibble: 1 x 1  
##   `min(mass, na.rm = TRUE)`  
##                 <dbl>  
## 1                  15
```

group_by

Agrupa as colunas de uma base de dados.

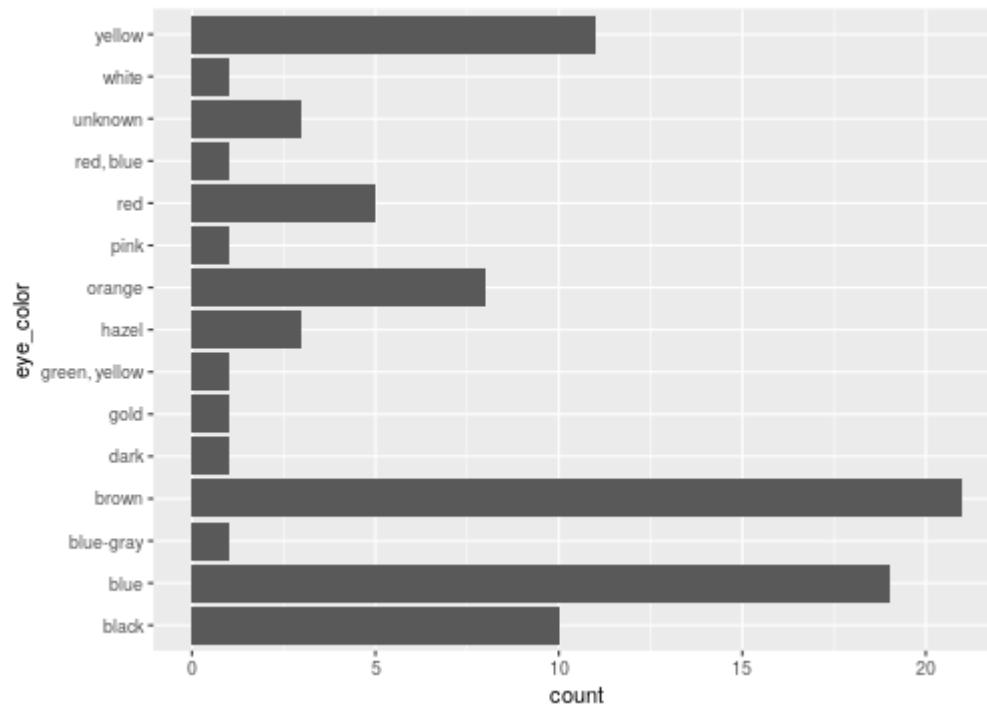
```
dados %>% group_by(species) %>%
  summarise(Total=n()) %>%
  arrange(desc(Total)) %>% drop_na()
```

```
## # A tibble: 37 x 2
##       species   Total
##       <chr>     <int>
## 1 Human         35
## 2 Droid          5
## 3 Gungan         3
## 4 Kaminoan       2
## 5 Mirialan       2
## 6 Twi'lek         2
## 7 Wookiee        2
## 8 Zabrak          2
## 9 Aleena          1
## 10 Besalisk        1
## # ... with 27 more rows
```

ggplot2

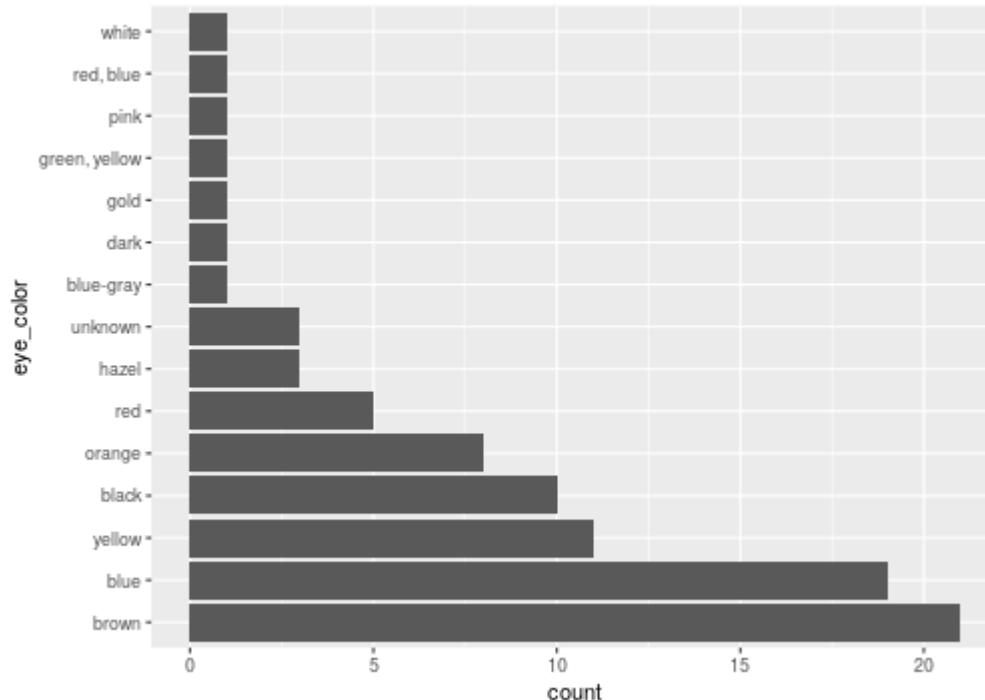
- divide os gráficos em componentes semânticos como escalas e camadas.

```
ggplot(dados, aes(x = eye_color)) +  
  geom_bar() +  
  coord_flip() # inverte os eixos do gráfico
```



ggplot2

```
dados %>% mutate(eye_color=fct_infreq(eye_color)) %>%# ordenando os
ggplot(aes(x = eye_color)) +
  geom_bar()+
  coord_flip()
```

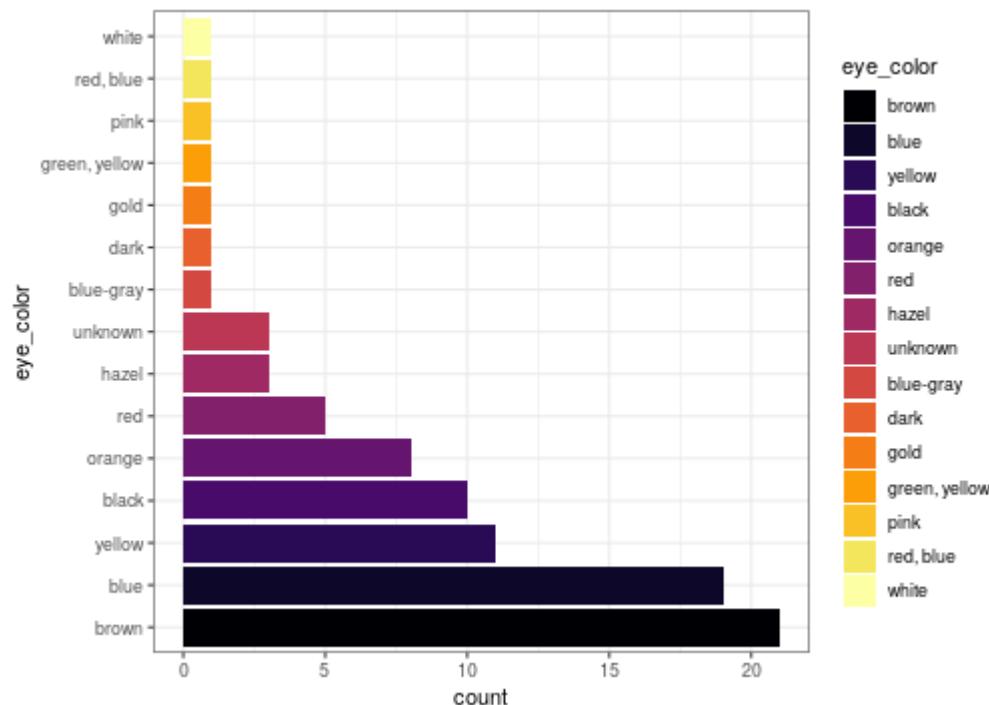


Os levels da variável eye_color estão ordenados.

```
library(viridis)
```

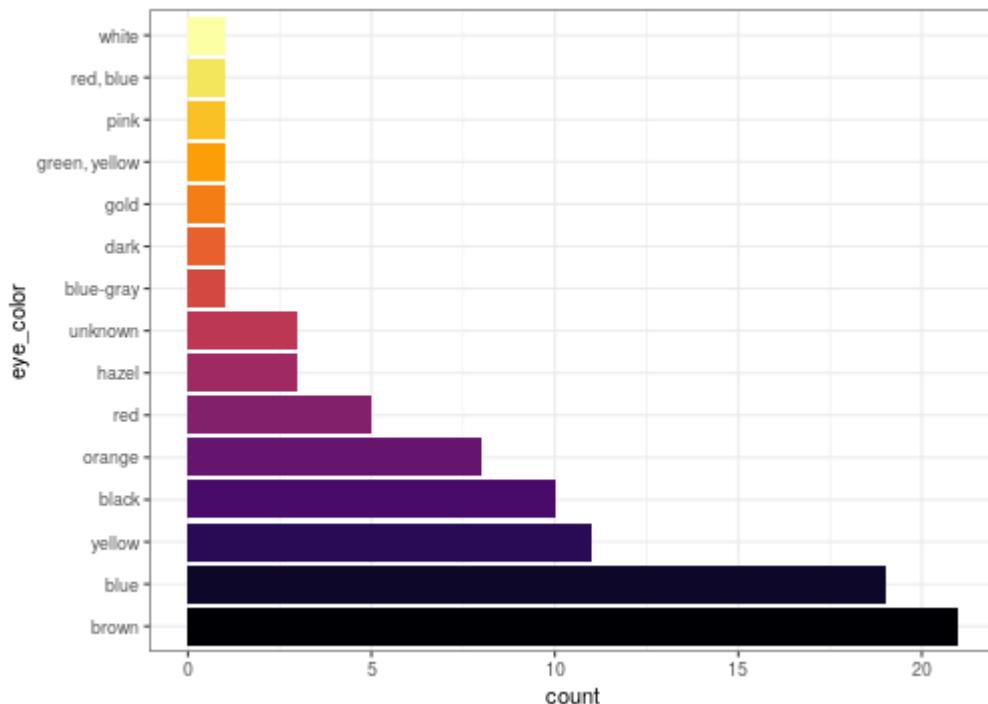
```
## Loading required package: viridisLite
```

```
dados %>% mutate(eye_color=fct_infreq(eye_color)) %>%# ordenando os
ggplot(aes(x = eye_color, fill=eye_color)) +
  geom_bar()+
  coord_flip() +
  theme_bw()+
  scale_fill_viridis_d(option = "inferno")
```

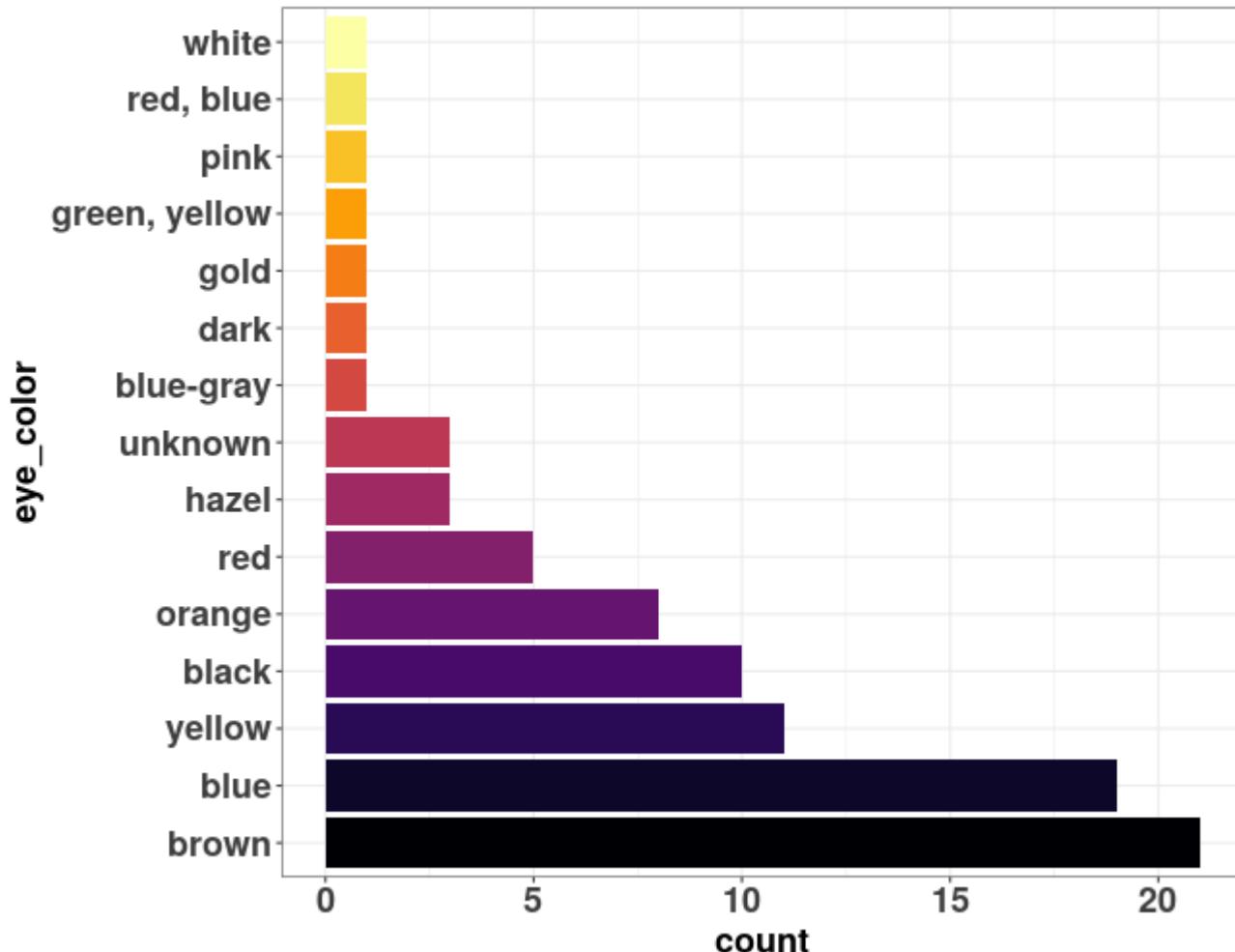


ggplot2

```
dados %>% mutate(eye_color=fct_infreq(eye_color)) %>%# ordenando os
ggplot(aes(x = eye_color, fill=eye_color)) + geom_bar()+
  coord_flip() + theme_bw() + scale_fill_viridis_d(option = "inferno"
  theme(legend.position = "none")
```



ggplot2



Referências

- R-ladies SP- https://github.com/r-ladies/meetup-presentations_sao-paulo
- UFPR- <http://leg.ufpr.br/~walmes/cursoR/data-vis/slides/02-tibble.pdf>
- http://ecologia.ib.usp.br/bie5782/doku.php?id=bie5782:03_apostila:01-intro
- Manny Gimond-
https://mgimond.github.io/ES218/Week02a.html#data_structures
- R for Data Science- <https://r4ds.had.co.nz/>
- Allison Horst - <https://github.com/allisonhorst/stats-illustrations>
- Xarigan - <https://bookdown.org/yihui/rmarkdown/xaringan.html>

Obrigada!