Mini-curso Noções básicas do R

Sollano Rabelo Braga

Novembro, 2017

Contents

0) Links úteis

Link para baixar a versão mais atualizada do R (Mantenha o R sempre atualizado):

http://cran.fiocruz.br/bin/windows/base/

Link para baixar a versão mais atualizada do RStudio: https://www.rstudio.com/products/rstudio/download/

Curso online gratuito disponibilizado pelo Prof. Eric Gorgens: https://eliademy.com/catalog/programac-o-cientifica-no-r.html

Lista do youtube com videos de introdução ao R, criados pelo Prof. Eric Gorgens: https://www.youtube.com/playlist?list=PLLCIDTaS6A7C4Ig6gf4d66hvyvl5e3zyy

Stack Overflow: Melhor lugar para se tirar dúvidas sobre o R. Ele tem versões em português e inglês, sendo que a em inglês é bem mais completa:

http://stackoverflow.com/questions/tagged/r (ingles)

http://pt.stackoverflow.com/questions/tagged/r (portugues)

R-Bloggers: Blog sobre R: Tutoriais, notícias sobre pacotes, discussões, etc:

https://www.r-bloggers.com/

Links com pdfs disponibilizados pelo pessoal do RStudio. Eles cobrem vários topicos, desde o R básico ao avançado:

https://www.rstudio.com/resources/cheatsheets/

1) Criar um projeto e uma área de trabalho

Quando se define a área de trabalho, pode-se trabalhar de forma mais direta podendo-se chamar objetos a partir daquele local diretamente.

Para isso existem duas formas:

Por linha de comando, ou utilizando a interface do RStudio.

Com a interface do RStudio, os trabalhos são separados em projetos, que preservam todas as informações de cada trabalho feito separadamente.

O projeto pode ser criado indo-se em file -> New Project ou no canto superior direito, so símbolo do RStudio

Carregando-se o projeto, ele irá carregar automaticamente a área de trabalho deste projeto

Utilizando linhas de comando pode-se apenas definir o diretório de trabalho, com o comando setwd. Este método era utilizado antes da IDE RStudio ser criada, tendo caído em desuso ultimamente.

2) Operadores e comandos úteis

Operadores são semelhantes a funções, e nos permitem fazer as operações mais básicas no R. Podem ser divididos entre operadores matemáticos e lógicos:

Operadores matemáticos:

"+" soma;

"-" subtração;

"/" divisão;

“*" multiplicação;

“%*%" multiplicação de matrizes;

“^” exponenciação ;

“~” usado em fórmulas de modelos (y ~ x);

Operadores lógicos:

“==” igualdade (não confundir com =, o operador de atribuição);

“!=” diferente de;

“!x” diferente de x;

“x & y” x E y;

“x | y” x OU y;

“>” maior que;

“=>” maior ou igual que;

“<” menor que;

“<=” menor ou igual que;

1:5 %in% 1:20; 1:5 existe dentro de 1:20?;

Operadores lógicos serão melhor exemplificados no tópico 8.

A seguir serão mostrados alguns comandos úteis. Esta lista pode ser utilizada ao longo do curso sempre que necessário:

Interrogação na frente de um comando ou termo abre o menu de ajuda:

```
?summary
```

Dimensão dos dados:

```
dim(iris)
```

```
## [1] 150   5
```

Número de linhas dos dados:

```
nrow(iris)
```

```
## [1] 150
```

Número de colunas dos dados:

```
ncol(iris)
```

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

Comprimento dos dados:

```
length(iris)
```

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

6 primeiras linhas dos dados:

```
head(iris)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

6 ultimas linhas dos dados:

```
tail(iris)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
145	6.7	3.3	5.7	2.5	virginica
146	6.7	3.0	5.2	2.3	virginica
147	6.3	2.5	5.0	1.9	virginica
148	6.5	3.0	5.2	2.0	virginica
149	6.2	3.4	5.4	2.3	virginica
150	5.9	3.0	5.1	1.8	virginica

Nomes das variáveis:

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
```

Informações estruturais sobre os dados:

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Classe das variáveis:

```
class(iris)
```

```
## [1] "data.frame"
```

Concatenar; Une números, caracteres em um vetor:

```
c(1,3,"quatro")
```

```
## [1] "1"      "3"      "quatro"
```

Compara dois dataframes e diz se eles são idênticos ou não:

```
all.equal(iris,iris)
```

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

remove todos os NAs:

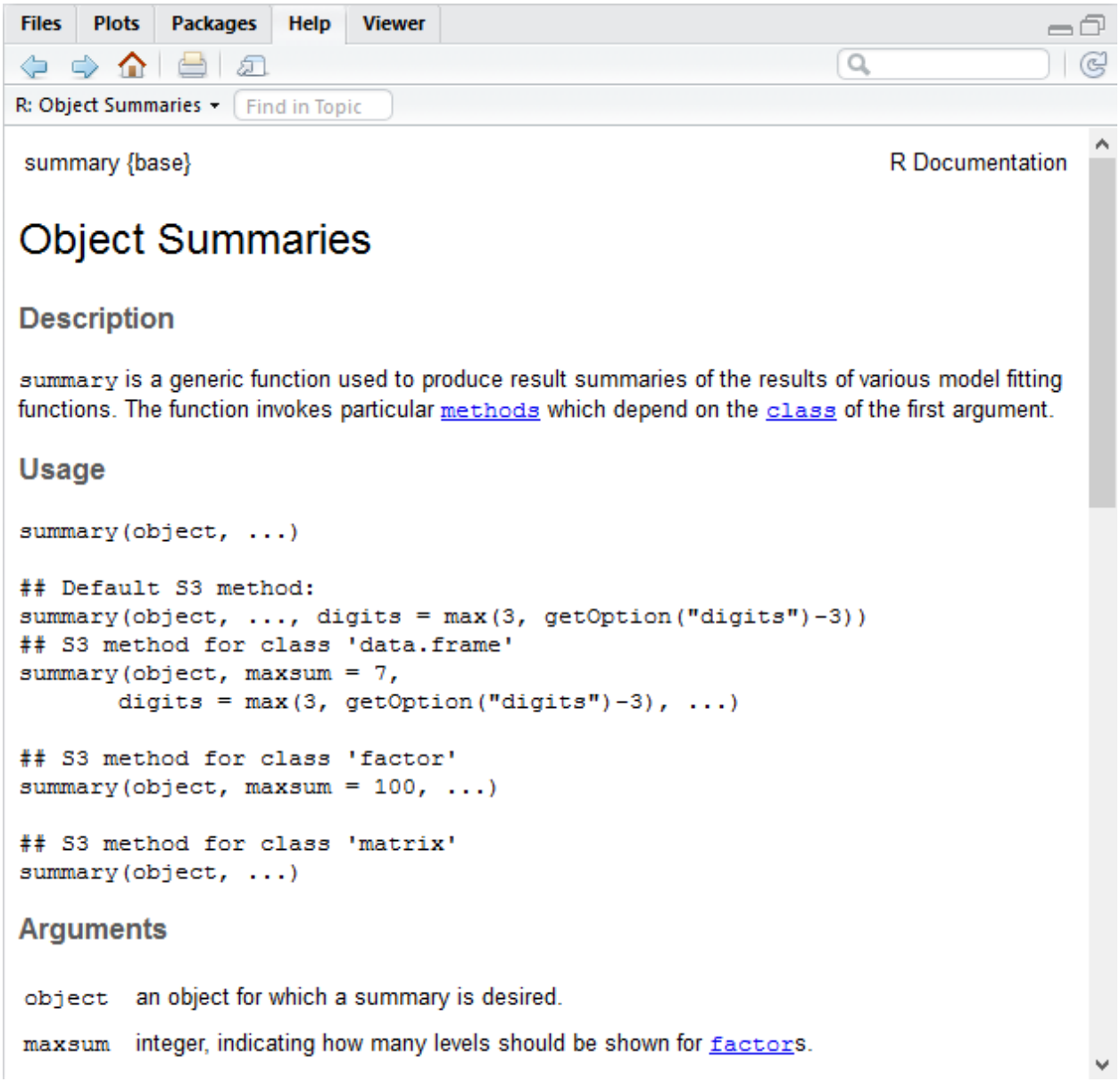


Figure 1: documentação de ajuda da função summary

```
# na.omit(iris)
```

Substitui NA por 0:

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

Substitui 0 por NA:

```
iris[iris==0] <- NA
```

Tabela de frequência absoluta:

```
table(iris$DESCCATEGORIA)
```

```
## < table of extent 0 >
```

Tabela de frequência relativa:

```
table(iris$DESCCATEGORIA)/nrow(iris) * 100
```

```
## numeric(0)
```

Apaga variável:

```
iris$Idade <- NULL
```

apaga objeto:

```
# rm(x)
```

Limpar a área de trabalho:

```
# rm(list = ls())
```

Listar todos os objetos:

```
#ls()
```

3) Criação de objetos e atribuição de valores

Objetos tem valores associados a eles quando se utiliza <- ou =. No exemplo a seguir, um objeto x e criado, que tem o valor de 10:

```
x <- 10
x = 10

x
```

```
## [1] 10
```

Para se deletar um objeto, utiliza-se a função rm():

```
rm(x)
```

Pode-se atribuir um conjunto de números, utilizando c(), ou seq():

```
y <- c(1,2,3,4,5,6,7,8)
z <- seq(from = 1, to = 8, by = 1)
z <- 1:8
```

R e sensível a maiúsculas e minúsculas (Case sensitive). então y e diferente de Y:

```
y
```

```
## [1] 1 2 3 4 5 6 7 8
```

```
#Y
```

Nomes de objetos podem conter números, mas não podem começar com números:

```
x1 <- 35
```

4) Classes de objetos (Class)

Dados carregados e salvos no ambiente R são chamados de objetos. pode-se criar objetos no R com o comando ->, <-, ou =

```
x <- 10
10 -> x
x = 10
```

Qual método utilizado para salvar objetos fica a critério do usuário entretanto, não e recomendado o uso do =, pois o mesmo símbolo e utilizado com outros significados, criando um código confuso.

neste curso será utilizado <- objetos podem pertencer a diferentes classes:

4.1) Numeric (Números decimais)

valores decimais são chamados de numéricos no R. Este e o tipo computacional de dadospadrão no R, ou seja, quando se cria um dado numérico, ele automaticamente possui classe numeric.

Pode-se verificar isto com a função class():

```
x <- 3

class(x)
```

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

pode-se verificar se x realmente e numeric com a função is.numeric()

```
is.numeric(x)
```

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

4.2) Integer (Números inteiros)

para se criar um objeto de classe integer, utiliza-se as.integer(), já que porpadrão o R cria objetos numéricos de classe numeric:

```
x <- as.integer(3)
as.integer(3)
```

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

```
x
```

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

```
class(x)
```

```
## [1] "integer"
```

Ao se utilizar a função as.integer em um valor decimal, ele se torna inteiro:

```
x <- as.integer(3.6)
x
```

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

Pode-se também converter valores lógicos (TRUE ou FALSE) em números inteiros assim como em outras linguagens, TRUE possui valor 1, e FALSE possui valor 0:

```
as.integer(TRUE)
```

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

```
as.integer(FALSE)
```

```
## [1] 0
```

4.3) Logical (lógicos)

Valores lógicos são utilizados para comparações, e são gerados toda vez que se compara objetos no R utilizando operadores como maior que (>), menor que (<), ou igual a (==).

Por exemplo, digamos que se tenha objetos chamados x e y, com valores de 1 e 2, respectivamente: Primeiro cria-se objetos para exemplo:

```
x <- 1; y <- 2
```

x e maior do que y?

```
x > y
```

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

```
class(x > y)
```

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

Se a comparação for feita com um vetor maior, o resultado são vários elementos lógicos:

```
5 > c(1:7)
```

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

Outro operador utilizado e o %in%, que indica se um conjunto de dados existe dentro de outro conjunto:

```
4:6 %in% 1:10
```

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

4.4) Character (Caracteres)

Character representa valores string no R. Palavras, frases, etc. Mesmo um dado de classe pode se tornar um valor string. Isto pode ocorrer involuntariamente, gerando erros, pois não e possível realizar calculos com dados de classe Character. Existem vários métodos de se criar e manipular objetos character, por enquanto serão utilizados os mais básicos.

Pode-se criar um objeto de classe character colocando letras ou números entre aspas, “”:

```
x <- "3.14"
y <- "programacão"
```

Classes dos objetos x e y:

```
class(x)
```

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

```
class(y)
```

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

Apesar de estar representando um número, ão se tentar realizar calculos com o objeto x, e gerado um erro:

```
# 3 + x
```

4.5) Factors (Fatores)

Variaveis nominais ou classificatórias são chamadas de fatores no R. Elas podem ser utilizadas para separar os dados em classes, ou grupos.

Sera utilizado como exemplo o dataframe Iris. pode-se obter mais informações sobre o dataframe com utilizando uma ? na frente do mesmo

```
?iris
```

Caso se rode apenas o nome do dataframe, o console fica poluído, pois será mostrado todo o dataframe no console, ou pelo menos as primeiras centenas de linhas:

```
iris
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa
5.4	3.4	1.7	0.2	setosa
5.1	3.7	1.5	0.4	setosa
4.6	3.6	1.0	0.2	setosa
5.1	3.3	1.7	0.5	setosa

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
4.8	3.4	1.9	0.2	setosa
5.0	3.0	1.6	0.2	setosa
5.0	3.4	1.6	0.4	setosa
5.2	3.5	1.5	0.2	setosa
5.2	3.4	1.4	0.2	setosa
4.7	3.2	1.6	0.2	setosa
4.8	3.1	1.6	0.2	setosa
5.4	3.4	1.5	0.4	setosa
5.2	4.1	1.5	0.1	setosa
5.5	4.2	1.4	0.2	setosa
4.9	3.1	1.5	0.2	setosa
5.0	3.2	1.2	0.2	setosa
5.5	3.5	1.3	0.2	setosa
4.9	3.6	1.4	0.1	setosa
4.4	3.0	1.3	0.2	setosa
5.1	3.4	1.5	0.2	setosa
5.0	3.5	1.3	0.3	setosa
4.5	2.3	1.3	0.3	setosa
4.4	3.2	1.3	0.2	setosa
5.0	3.5	1.6	0.6	setosa
5.1	3.8	1.9	0.4	setosa
4.8	3.0	1.4	0.3	setosa
5.1	3.8	1.6	0.2	setosa
4.6	3.2	1.4	0.2	setosa
5.3	3.7	1.5	0.2	setosa
5.0	3.3	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
5.5	2.3	4.0	1.3	versicolor
6.5	2.8	4.6	1.5	versicolor
5.7	2.8	4.5	1.3	versicolor
6.3	3.3	4.7	1.6	versicolor
4.9	2.4	3.3	1.0	versicolor
6.6	2.9	4.6	1.3	versicolor
5.2	2.7	3.9	1.4	versicolor
5.0	2.0	3.5	1.0	versicolor
5.9	3.0	4.2	1.5	versicolor
6.0	2.2	4.0	1.0	versicolor
6.1	2.9	4.7	1.4	versicolor
5.6	2.9	3.6	1.3	versicolor
6.7	3.1	4.4	1.4	versicolor
5.6	3.0	4.5	1.5	versicolor
5.8	2.7	4.1	1.0	versicolor
6.2	2.2	4.5	1.5	versicolor
5.6	2.5	3.9	1.1	versicolor
5.9	3.2	4.8	1.8	versicolor
6.1	2.8	4.0	1.3	versicolor
6.3	2.5	4.9	1.5	versicolor
6.1	2.8	4.7	1.2	versicolor
6.4	2.9	4.3	1.3	versicolor
6.6	3.0	4.4	1.4	versicolor

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
6.8	2.8	4.8	1.4	versicolor
6.7	3.0	5.0	1.7	versicolor
6.0	2.9	4.5	1.5	versicolor
5.7	2.6	3.5	1.0	versicolor
5.5	2.4	3.8	1.1	versicolor
5.5	2.4	3.7	1.0	versicolor
5.8	2.7	3.9	1.2	versicolor
6.0	2.7	5.1	1.6	versicolor
5.4	3.0	4.5	1.5	versicolor
6.0	3.4	4.5	1.6	versicolor
6.7	3.1	4.7	1.5	versicolor
6.3	2.3	4.4	1.3	versicolor
5.6	3.0	4.1	1.3	versicolor
5.5	2.5	4.0	1.3	versicolor
5.5	2.6	4.4	1.2	versicolor
6.1	3.0	4.6	1.4	versicolor
5.8	2.6	4.0	1.2	versicolor
5.0	2.3	3.3	1.0	versicolor
5.6	2.7	4.2	1.3	versicolor
5.7	3.0	4.2	1.2	versicolor
5.7	2.9	4.2	1.3	versicolor
6.2	2.9	4.3	1.3	versicolor
5.1	2.5	3.0	1.1	versicolor
5.7	2.8	4.1	1.3	versicolor
6.3	3.3	6.0	2.5	virginica
5.8	2.7	5.1	1.9	virginica
7.1	3.0	5.9	2.1	virginica
6.3	2.9	5.6	1.8	virginica
6.5	3.0	5.8	2.2	virginica
7.6	3.0	6.6	2.1	virginica
4.9	2.5	4.5	1.7	virginica
7.3	2.9	6.3	1.8	virginica
6.7	2.5	5.8	1.8	virginica
7.2	3.6	6.1	2.5	virginica
6.5	3.2	5.1	2.0	virginica
6.4	2.7	5.3	1.9	virginica
6.8	3.0	5.5	2.1	virginica
5.7	2.5	5.0	2.0	virginica
5.8	2.8	5.1	2.4	virginica
6.4	3.2	5.3	2.3	virginica
6.5	3.0	5.5	1.8	virginica
7.7	3.8	6.7	2.2	virginica
7.7	2.6	6.9	2.3	virginica
6.0	2.2	5.0	1.5	virginica
6.9	3.2	5.7	2.3	virginica
5.6	2.8	4.9	2.0	virginica
7.7	2.8	6.7	2.0	virginica
6.3	2.7	4.9	1.8	virginica
6.7	3.3	5.7	2.1	virginica
7.2	3.2	6.0	1.8	virginica
6.2	2.8	4.8	1.8	virginica
6.1	3.0	4.9	1.8	virginica

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
6.4	2.8	5.6	2.1	virginica
7.2	3.0	5.8	1.6	virginica
7.4	2.8	6.1	1.9	virginica
7.9	3.8	6.4	2.0	virginica
6.4	2.8	5.6	2.2	virginica
6.3	2.8	5.1	1.5	virginica
6.1	2.6	5.6	1.4	virginica
7.7	3.0	6.1	2.3	virginica
6.3	3.4	5.6	2.4	virginica
6.4	3.1	5.5	1.8	virginica
6.0	3.0	4.8	1.8	virginica
6.9	3.1	5.4	2.1	virginica
6.7	3.1	5.6	2.4	virginica
6.9	3.1	5.1	2.3	virginica
5.8	2.7	5.1	1.9	virginica
6.8	3.2	5.9	2.3	virginica
6.7	3.3	5.7	2.5	virginica
6.7	3.0	5.2	2.3	virginica
6.3	2.5	5.0	1.9	virginica
6.5	3.0	5.2	2.0	virginica
6.2	3.4	5.4	2.3	virginica
5.9	3.0	5.1	1.8	virginica

Uma melhor forma de se visualizar o dado e visualizar as primeiras e ultimas 6 linhas com head() e tail().

Primeiras 6 linhas:

```
head(iris)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

Ultimas 6 linhas:

```
tail(iris)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
145	6.7	3.3	5.7	2.5	virginica
146	6.7	3.0	5.2	2.3	virginica
147	6.3	2.5	5.0	1.9	virginica
148	6.5	3.0	5.2	2.0	virginica
149	6.2	3.4	5.4	2.3	virginica
150	5.9	3.0	5.1	1.8	virginica

Iris da informações de largura e comprimento de sépalas e pétalas de flores por espécie. Pode-se verificar os

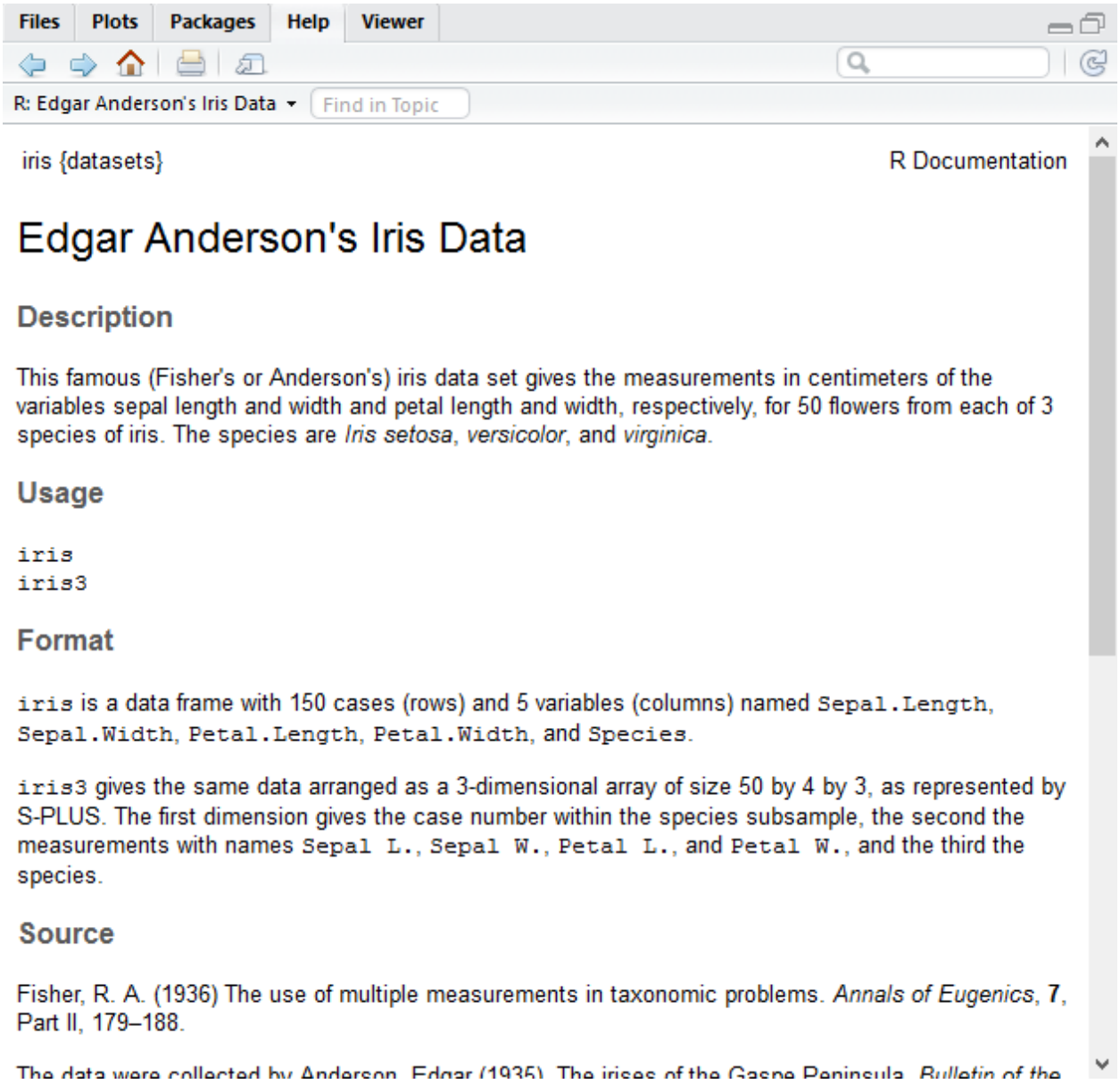


Figure 2: documentação de ajuda dao dataframe iris

nomes das variáveis de iris:

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
```

```
iris$Species
```

```
##   [1] setosa     setosa     setosa     setosa     setosa     setosa
##   [7] setosa     setosa     setosa     setosa     setosa     setosa
##  [13] setosa     setosa     setosa     setosa     setosa     setosa
##  [19] setosa     setosa     setosa     setosa     setosa     setosa
##  [25] setosa     setosa     setosa     setosa     setosa     setosa
##  [31] setosa     setosa     setosa     setosa     setosa     setosa
##  [37] setosa     setosa     setosa     setosa     setosa     setosa
##  [43] setosa     setosa     setosa     setosa     setosa     setosa
##  [49] setosa     setosa     versicolor versicolor versicolor versicolor
##  [55] versicolor versicolor versicolor versicolor versicolor versicolor
##  [61] versicolor versicolor versicolor versicolor versicolor versicolor
##  [67] versicolor versicolor versicolor versicolor versicolor versicolor
##  [73] versicolor versicolor versicolor versicolor versicolor versicolor
##  [79] versicolor versicolor versicolor versicolor versicolor versicolor
##  [85] versicolor versicolor versicolor versicolor versicolor versicolor
##  [91] versicolor versicolor versicolor versicolor versicolor versicolor
##  [97] versicolor versicolor versicolor versicolor virginica  virginica
## [103] virginica  virginica  virginica  virginica  virginica  virginica
## [109] virginica  virginica  virginica  virginica  virginica  virginica
## [115] virginica  virginica  virginica  virginica  virginica  virginica
## [121] virginica  virginica  virginica  virginica  virginica  virginica
## [127] virginica  virginica  virginica  virginica  virginica  virginica
## [133] virginica  virginica  virginica  virginica  virginica  virginica
## [139] virginica  virginica  virginica  virginica  virginica  virginica
## [145] virginica  virginica  virginica  virginica  virginica  virginica
## Levels: setosa versicolor virginica
```

Observando a classe da variável Species, veremos que se trata de um fator:

```
class(iris$Species)
```

```
## [1] "factor"
```

Um fator agrupa observações repetidas, e as separa em grupos; estes são chamados de levels, ou níveis.

Aqui obtem-se os nomes das 3 espécies neste dataframe:

```
levels(iris$Species)
```

```
## [1] "setosa"     "versicolor" "virginica"
```

E se o objetivo fosse obter os dados de largura e comprimento apenas da espécie setosa?

Como a variável Species separa os dados nos níveis "setosa", "versicolor", "virginica", basta filtrar iris com a variável Species, especificando que se deseja apenas os dados que estão dentro no nível "setosa":

```
iris[iris$Species=="setosa" , ]
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa
5.4	3.4	1.7	0.2	setosa
5.1	3.7	1.5	0.4	setosa
4.6	3.6	1.0	0.2	setosa
5.1	3.3	1.7	0.5	setosa
4.8	3.4	1.9	0.2	setosa
5.0	3.0	1.6	0.2	setosa
5.0	3.4	1.6	0.4	setosa
5.2	3.5	1.5	0.2	setosa
5.2	3.4	1.4	0.2	setosa
4.7	3.2	1.6	0.2	setosa
4.8	3.1	1.6	0.2	setosa
5.4	3.4	1.5	0.4	setosa
5.2	4.1	1.5	0.1	setosa
5.5	4.2	1.4	0.2	setosa
4.9	3.1	1.5	0.2	setosa
5.0	3.2	1.2	0.2	setosa
5.5	3.5	1.3	0.2	setosa
4.9	3.6	1.4	0.1	setosa
4.4	3.0	1.3	0.2	setosa
5.1	3.4	1.5	0.2	setosa
5.0	3.5	1.3	0.3	setosa
4.5	2.3	1.3	0.3	setosa
4.4	3.2	1.3	0.2	setosa
5.0	3.5	1.6	0.6	setosa
5.1	3.8	1.9	0.4	setosa
4.8	3.0	1.4	0.3	setosa
5.1	3.8	1.6	0.2	setosa
4.6	3.2	1.4	0.2	setosa
5.3	3.7	1.5	0.2	setosa
5.0	3.3	1.4	0.2	setosa

Isto e chamado de subsetting; existe um tópico separado neste curso para estas operações.

5) Tipos de objetos (Modes)

No R, dados podem ser organizados em diferentes tipos, ou modos; as principais serão refenciadas neste tópico.

5.1) Vectors (vetores)

No R, um vetor é uma sequência de dados de mesma classe; vetores possuem apenas uma dimensão; um vetor pode ser criado com a função c(), que concatena valores:

```
a <- c(1, 2, 5.3, 6, -2, 4)
a # vetor numérico
```

```
## [1]  1.0  2.0  5.3  6.0 -2.0  4.0
```

```
b <- c("um", "dois", "tres")
b # vetor character
```

```
## [1] "um"   "dois" "tres"
```

```
c <- c(TRUE,TRUE,TRUE,FALSE,TRUE,FALSE)
c # vetor logical
```

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

Ao se tentar unir strings e números, ou seja, objetos de classes diferentes em um único vetor, os números serão "coagidos" (coerced) para caracteres, pois um vetor so pode ter uma classe:

```
# atenção aos números entre aspas, indicando que eles na verdade são caracteres:
c(a, b)
```

```
## [1] "1"    "2"    "5.3"  "6"    "-2"   "4"    "um"   "dois" "tres"
```

5.2) Matrices (Matrizes)

No R, uma matriz é um conjunto de elementos organizados de forma retangular bidimensional; Todas as colunas de uma matriz devem ter o mesmo número de elementos e a mesma classe(numeric,character, etc).

Pode-se criar uma matriz com a função matrix():

```
d <- matrix(
  c(2, 4, 3, 1), # dados a serem inseridos
  nrow=2,              # número de linhas
  ncol=2,              # número de colunas
  byrow = TRUE)        # organizar a matriz por linha
```

matriz que vai de 1 a 4, com duas linhas, organizada por coluna:

```
e <- matrix(1:4, nrow = 2,byrow = FALSE)
```

matriz d no console:

```
d
```

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

Pode-se fazer operações matemáticas com matrizes: Multiplicação:

```
d * 4
```

```
##      [,1] [,2]
## [1,]    8   16
## [2,]   12    4
```

Multiplicação de matrizes:

```
d %*% e
```

```
##      [,1] [,2]
## [1,]   10   22
## [2,]    5   13
```

Matriz transposta:

```
t(d)
```

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

Matriz Inversa:

```
solve(d)
```

```
##      [,1] [,2]
## [1,] -0.1  0.4
## [2,]  0.3 -0.2
```

elementos de uma matriz podem ser referenciados utilizando [], utilizando df[linha, coluna], por exemplo: Primeira linha, segunda coluna:

```
d[1,2]
```

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

Todos os objetos da segunda linha:

```
d[2, ]
```

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

mais detalhes sobre manipulação de matrizes serão abordados em tópicos futuros.

5.3) Data Frames

Dataframes (df) são a forma padrão de se inserir dados no R. Um data frame é mais geral do que uma matriz, de forma que diferentes colunas podem ter diferentes classes.

Colunas são chamadas de variáveis (variables) e linhas de observações (observations);

Pode-se utilizar vetores para criar um data frame:

```
a <- c(1, 2, 3, 4)
b <- c("vermelho", "azul", "vermelho", "rosa")
c <- c(TRUE,TRUE,TRUE,FALSE)

meusdados <- data.frame(a,b,c)
meusdados
```

a	b	c
1	vermelho	TRUE
2	azul	TRUE
3	vermelho	TRUE
4	rosa	FALSE

Observa-se que agora na aba “environment, abaixo de Data, existe o df”meusdados“, com 4 observações e 3 variáveis, ou seja, 4 linhas e 3 colunas.

Outra forma de verificar isto e com a função dim():

```
dim(meusdados)
```

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

Os nomes das variáveis de um df podem ser alterados:

```
names(meusdados) <- c("Idade", "ID", "Teste")
```

Verifica-se os nomes das variáveis:

```
names(meusdados)
```

```
## [1] "Idade" "ID"    "Teste"
```

Pode-se visualizá-lo como uma tabela, clicando nele na aba environment, ou utilizando View:

```
# View(meusdados)
```

Elementos de um df podem ser chamados por nome, com $, ou por posição, com []:

variável Idade:

```
meusdados$Idade
```

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

terceira coluna do df meusdados:

```
meusdados[,3]
```

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

o R possui dataframes built-in, ou seja, que já vem instalados com o a instalação básica:

```
head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

```
head(iris)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

mais detalhes sobre a manipulação de data frames serãoabordadosem tópicos futuros.

5.4) Lists (Listas)

Listas são coleções ordenadas de objetos; Podem unir objetos de diferentes tipos, classes e domensões em um único objeto. São uteis quando se quer unir informações de objetos de domensões diferentes, e na otimização de loops.

Por exemplo: Um dataframe e um vetor podem ser unidos por uma lista:

```
meusdados <- data.frame(a = c(1, 2, 3, 4),
                        b = c("vermelho", "azul", "vermelho", "rosa"),
                        c = c(TRUE, TRUE, TRUE, FALSE))

d <- matrix( c(2, 4, 3, 1),   nrow=2, byrow = TRUE)

minhalista <- list(d, meusdados)

minhalista
```

```
## [[1]]
##      [,1] [,2]
## [1,]    2    4
## [2,]    3    1
##
## [[2]]
##   a        b     c
## 1 1 vermelho  TRUE
## 2 2     azul  TRUE
## 3 3 vermelho  TRUE
## 4 4     rosa FALSE
```

```
class(minhalista)
```

```
## [1] "list"
```

Elementos de uma lista são identificados utilizando [[]] primeiro elemento de uma lista:

```
minhalista[[1]]
```

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

segundo elemento de uma lista:

```
minhalista[[2]]
```

a	b	c
1	vermelho	TRUE
2	azul	TRUE
3	vermelho	TRUE
4	rosa	FALSE

observa-se que a classe do objeto não se altera dentro da lista:

```
class(minhalista[[1]])
```

```
## [1] "matrix"
```

Caso seja utilizado apenas um colchete, a classe do objeto será de uma lista:

```
class(minhalista[1])
```

```
## [1] "list"
```

5.5) Functions (funções)

funções são formas de se compactar rotinas ou operações complexas, que se deseja fazer repetidas vezes, em apenas um comando, facilitando a digitação, diminuindo o tamanho de scripts, etc.

O R possui funções que são nativas dele, que podem ser acessadas assim que se abre o programa. Estas são as funções conhecidas como base. Novas funções podem ser adquiridas por meio de pacotes, ou podem ser criadas pelo próprio usuário.

Para acessar uma função do R base, basta digitar o nome dela, seguido de um parênteses. Por exemplo:

```
sum(4,5)
```

```
## [1] 9
```

Todas as funções no R possuem argumentos. Argumentos são as entradas da função, que são separados por "," . Por exemplo, para se descobrir os argumentos função matrix, usa-se o comando:

```
args(matrix)
```

```
## function (data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)
## NULL
```

Para se saber mais sobre uma função, basta utilizar ? seguido do nome da função:

```
?matrix
```

Agora se tem uma descrição detalhada da função. Com isso, conclui-se que:

1. Entrada de dados, default: NA
2. número de linhas da matriz, default: 1
3. número de colunas, default: 1
4. organização das colunas, default: TRUE
5. definição de nomes dinamico, default: NULL A função matrix possui valorespadrão para todos os seus argumentos. Portando, se a função for rodada sem nenhum argumento modificado, ela irá gerar uma matrix 1x1, composta por NA:

```
matrix()
```

```
##      [,1]
## [1,]   NA
```

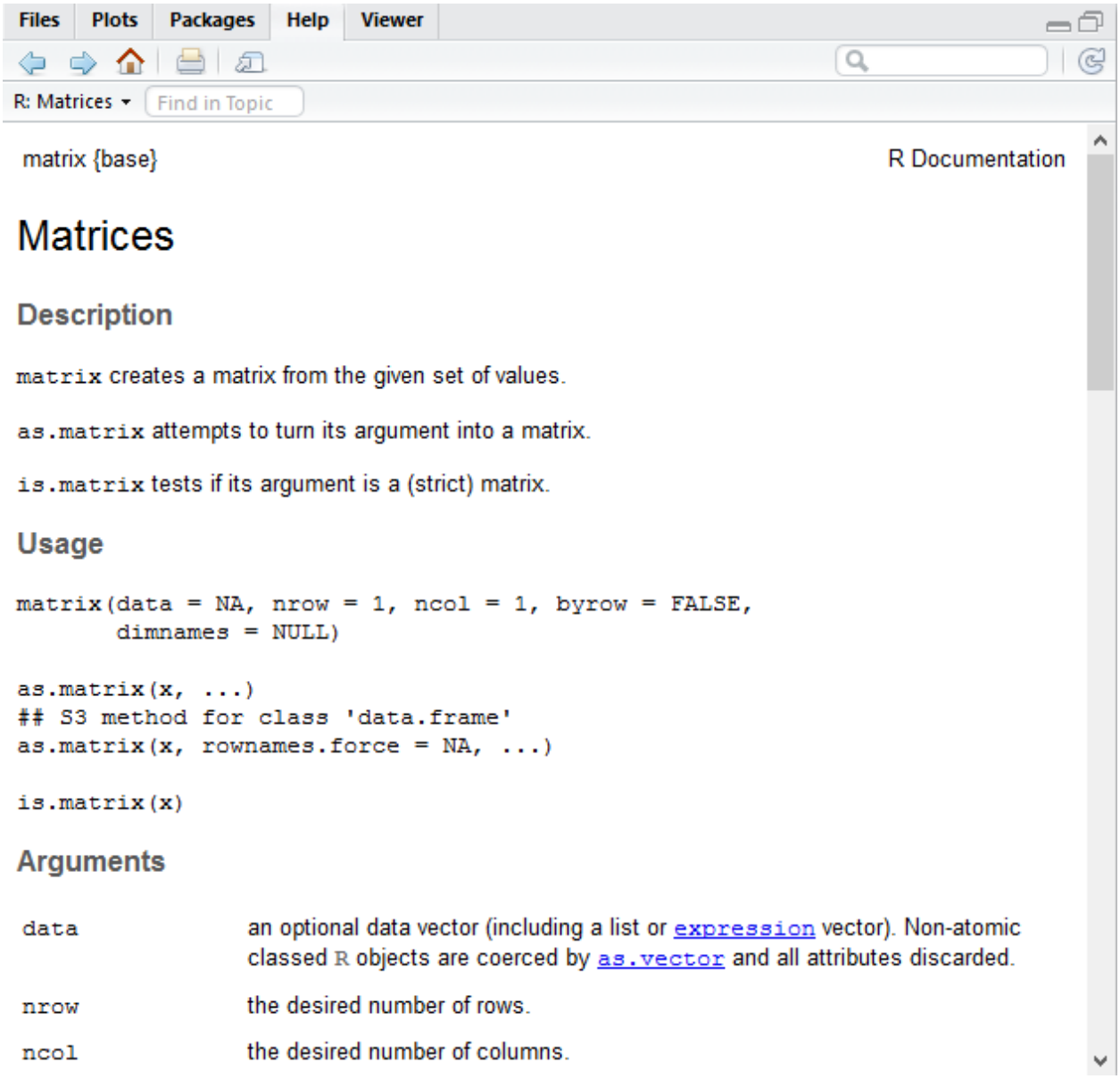


Figure 3: documentação de ajuda da função matrix

Caso se altere estes argumentos, altera-se a saída:

```
matrix(1)
```

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

```
matrix(1, nrow = 2 )
```

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

```
matrix(1, nrow = 2, ncol = 2 )
```

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

Algumas funções não funcionam sem uma entrada, como por exemplo, summary. Summary nos da um resumo de um determinado objeto.

```
args(summary)
```

```
## function (object, ...)
## NULL
```

não existe default para o argumento object, indicando que ele e um argumento obrigatório. Se a função for rodada sem uma entrada, ela irá gerar um erro:

```
#summary()
```

Agora caso inserido um dataframe como entrada, a função retorna média, quartis, mediana, mínimo e máximo de cada variável:

```
summary(mtcars)
```

```
##       mpg             cyl             disp             hp
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0
##       drat             wt             qsec             vs
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
##        am              gear            carb
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
##  Median :0.0000   Median :4.000   Median :2.000
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

6) Pacotes

Pacotes são a unidade fundamental de código reproduzível no R. Eles incluem funções do R reutilizáveis, a documentação que explica como utilizá-la, e dados de exemplo. Geralmente pacotes são feitos com um propósito específico, que facilita alguma operação que se deseja fazer no R, como por exemplo facilitar a manipulação de dados, gráficos mais complexos e bonitos, importação de arquivos que por padrão não são aceitos no R, entre outros.

Qualquer pessoa pode criar um pacote e compartilhá-lo. Mas, para que ele se torne um pacote "oficial" do R, deve ser enviado para o CRAN, onde passa por uma rigorosa analise, ate que seja aceito e disponível para download pelo mesmo.

Para se instalar pacotes utiliza-se a função install.packages.

Para instalar um pacote basta rodar a função inserindo o nome do pacote entre aspas.

É interessante sempre utilizar o argumento dependencies com valor TRUE, para que seja instalado junto com o pacote, todos os pacotes que este precisa para funcionar corretamente. A seguir exemplifica-se a instalação do pacote dplyr:

```
#install.packages("dplyr", dependencies = TRUE)
```

Esta operação e feita uma unica vez, e precisa ser refeita apenas quando se deseja atualizar o pacote.

Para se carregar um pacote, basta utilizar a função library ou require:

```
library(dplyr)
```

ou:

```
require(dplyr)
```

É necessário carregar um pacote que se deseja utilizar sempre que se iniciar uma nova seção, ou seja, sempre que se abrir o programa.

A seguir, uma lista dos pacotes mais utilizados e recomendados:

1. tidyverse - É um conjunto de pacotes para ciência de dados. inclui alguns dos pacotes mais utilizados no mundo, como dplyr, ggplot2, entre outros. Alguns destes serão detalhados abaixo. É muito recomendado explorar todos os pacotes inclusos no tidyverse.
2. dplyr - para manipulação de dados. Composto por funções simples e fáceis de usar que resolvem a maioria dos problemas envolvendo manipulação de dados;
3. tidyr - Organização dos dados. Transformar tabelas longas, unir variáveis, lidar com NA, etc;
4. readxl - para carregar arquivos direto do excel em formato .xlsx;
5. ggplot2 - para criação de gráficos bonitos e customizáveis;
6. raster - para manipulação de imagens e shape files no R;
7. ExpDes.pt - Pacote destinado a analise de delineamentos experimentais (todo em portugues);
8. agricolae - Pacote destinado a analise de delineamentos experimentais;
9. vegan - Pacote utilizado para análises descritivas de ecologia;

A única desvantagem em se utilizar pacotes, é que pode acontecer do pacote ser atualizado e uma função mudar de nome, ou deixar de existir, deixando códigos antigos inutilizados. [Códigos escritos utilizando as funções padrões do R, também conhecidas como r base são mais recomendados.

7) Importar dados no R

Pode-se importar e exportar dados no R base (sem a utilização de pacotes) pelos formatos .csv e .txt; os dois formatos são semelhantes em questão de tamanho e velocidade, fica a critério do usuário qual tipo de arquivo utilizar.

utiliza-se read.csv2 para arquivos com separador ";" e dec "," (padrão nacional/europeu); e read.csv para arquivos com separador "," e dec "." (padrão americano).

As funções de importação de dados tem como primeiro argumento o caminho em que o arquivo que se deseja importar se encontra. Por exemplo, se o arquivo estivesse na pasta R, de meus documentos, e o nome do arquivo fosse dados, e seu formato csv, o seu caminho seria:

D:/Meus Documentos/R/dados.csv

Digita-se o caminho, seguido do nome do objeto e a sua extensão. Caso o objeto esteja dentro da pasta da área de trabalho, basta digitar o nome do arquivo e sua extensão, que ele será importado. Por exemplo: Importa-se dados inserindo o caminho completo:

```
dados <- read.csv2(file = "C:/Users/solla/Documents/R/aulas_cursos/nocoes_R/dados.csv" )
```

Digitando-se apenas o nome do arquivo, localizado no diretório de trabalho:

```
dados <- read.csv2("dados.csv")
```

O dado anterior foi salvo utilizando o padrão brasileiro de separador e decimal. Caso o arquivo possua o padrão americano, deve-se usar a função read.csv, ou altera-se os argumentos de dec e sep.

Para se importar dados em .txt, utiliza-se read.table. Deve-se alterar o argumento header para TRUE, pois o padrão e FALSE, e especificar o separador, dependendo do padrão utilizado:

```
dados <- read.table("C:/Users/solla/Documents/R/aulas_cursos/nocoes_R/dados.txt",
                    header = T, dec = ",", sep = "\t")

dados <- read.table("dados.txt", header = T, dec = ",", sep = "\t")
```

Caso seja utilizada a IDE RStudio, também e possível importar dados pelo menu "import dataset", na aba environment.

porém e recomendado utilizar-se das linhas de comando, para que se ganhe familiaridade com o programa, e para que os scripts criados futuramente tenham a importação dos dados nele, tornando-os mais práticos.

Importar dados direto em formato excel

Isto pode ser feito utilizando o pacote readxl:

```
library(readxl)
teste <- read_xlsx("dados.xlsx")
teste
```

CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA
3656	101	101	51.2	21.5	Normal
3656	101	103	52.2	20.9	Normal
3656	101	201	22.6	NA	Outros Fustes
3656	101	203	56.0	21.7	Normal
3656	101	205	45.2	20.5	Normal
3656	101	302	58.4	21.6	Dominante
3656	101	304	62.9	NA	Bifurcadaacima1.30
3656	101	306	53.8	NA	Inclinada
3656	101	402	43.7	NA	Inclinada

CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA
3656	101	404	46.5	NA	Normal
3656	101	502	41.0	NA	Normal
3656	102	101	47.7	20.1	Normal
3656	102	103	43.7	NA	Bifurcadaacima1.30
3656	102	202	47.3	20.4	Normal
3656	102	204	52.3	21.1	Normal
3656	102	301	49.8	20.6	Normal
3656	102	303	46.0	NA	Normal
3656	102	305	56.1	21.5	Dominante
3656	102	401	38.4	NA	Normal
3656	102	403	51.7	NA	Normal
3656	102	405	47.0	NA	Normal
3656	102	501	55.3	NA	Normal
3656	102	503	54.2	NA	Normal
3656	103	101	58.2	NA	Bifurcadaacima1.30
3656	103	201	50.1	NA	Bifurcadaabaixo1.30
3656	103	202	52.6	21.4	Normal
3656	103	204	42.4	19.7	Normal
3656	103	302	50.9	20.9	Normal
3656	103	304	55.1	21.4	Dominante
3656	103	305	37.4	NA	Outros Fustes
3656	103	401	48.8	NA	Bifurcadaabaixo1.30
3656	103	402	50.7	NA	Normal
3656	103	404	52.3	NA	Normal
3656	103	502	47.0	NA	Normal
3656	104	101	57.5	20.5	Normal
3656	104	201	54.7	21.1	Normal
3656	104	203	50.6	NA	Bifurcadaacima1.30
3656	104	205	NA	NA	Falha
3656	104	302	57.0	22.2	Normal
3656	104	304	51.7	NA	Normal
3656	104	402	46.7	NA	Bifurcadaacima1.30
3656	104	404	57.8	21.6	Dominante
3656	104	502	50.8	NA	Normal
3654	101	101	51.0	29.2	Normal
3654	101	202	56.6	30.6	Normal
3654	101	204	53.2	28.8	Normal
3654	101	302	51.3	29.0	Normal
3654	101	304	53.2	NA	Normal
3654	101	401	54.0	NA	Normal
3654	101	403	55.8	NA	Normal
3654	101	405	59.0	30.2	Dominante
3654	101	502	40.5	NA	Normal
3654	102	101	58.9	32.5	Normal
3654	102	103	NA	NA	Falha
3654	102	202	53.6	28.9	Normal
3654	102	204	57.7	29.8	Normal
3654	102	302	55.5	29.7	Normal
3654	102	304	NA	NA	Falha
3654	102	401	24.2	NA	Bifurcadaabaixo1.30
3654	102	402	29.7	NA	Bifurcadaabaixo1.30
3654	102	403	50.9	NA	Normal

CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA
3654	102	405	53.6	NA	Normal
3654	102	502	50.5	NA	Normal
3654	103	101	NA	NA	Falha
3654	103	201	52.4	29.5	Normal
3654	103	203	NA	NA	Falha
3654	103	301	NA	NA	Falha
3654	103	303	65.5	30.8	Dominante
3654	103	305	52.8	28.6	Normal
3654	103	401	24.5	NA	Dominada
3654	103	403	56.1	27.8	Normal
3654	103	405	53.5	NA	Normal
3654	103	502	55.4	NA	Normal
3654	104	101	53.1	27.7	Normal
3654	104	103	49.3	26.8	Normal
3654	104	202	52.1	30.2	Normal
3654	104	204	49.3	27.8	Normal
3654	104	302	49.8	NA	Normal
3654	104	304	57.4	29.1	Dominante
3654	104	402	58.1	29.3	Dominante
3654	104	404	54.0	NA	Normal
3654	104	501	50.2	NA	Normal

8) Indexar ou extrair partes de um objeto (subsetting) e lógica

8.1) Subsetting

Quando se trata de dataframes, pode-se utilizar $ para selecionar uma coluna especifica por nome, ou [] para especificar uma linha e/ou coluna especifica por posição.

Utilizando [], primeiro especifica-se a linha, e depois a coluna, separados por ","; caso um destes espaços fiquem vazios, toda a seção e selecionada.

Primeiramente importa-se o dado que será utilizado neste capitulo:

```
dados <- read.csv2("dados.csv")
head(dados)
```

CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA
3656	101	101	51.2	21.5	Normal
3656	101	103	52.2	20.9	Normal
3656	101	201	22.6	NA	Outros Fustes
3656	101	203	56.0	21.7	Normal
3656	101	205	45.2	20.5	Normal
3656	101	302	58.4	21.6	Dominante

Utilizando o dataframe dados, pode-se selecionar apenas a variável CAP, utilizando $:

```
dados$CAP
```

```
##  [1] 51.2 52.2 22.6 56.0 45.2 58.4 62.9 53.8 43.7 46.5 41.0 47.7 43.7 47.3
## [15] 52.3 49.8 46.0 56.1 38.4 51.7 47.0 55.3 54.2 58.2 50.1 52.6 42.4 50.9
```

```
## [29] 55.1 37.4 48.8 50.7 52.3 47.0 57.5 54.7 50.6   NA 57.0 51.7 46.7 57.8
## [43] 50.8 51.0 56.6 53.2 51.3 53.2 54.0 55.8 59.0 40.5 58.9   NA 53.6 57.7
## [57] 55.5   NA 24.2 29.7 50.9 53.6 50.5   NA 52.4   NA   NA 65.5 52.8 24.5
## [71] 56.1 53.5 55.4 53.1 49.3 52.1 49.3 49.8 57.4 58.1 54.0 50.2
```

Com isso pode-se pedir a média do CAP; É importante utilizar o argumento na.rm, que remove qualquer na contido no dado:

```
mean(dados$CAP, na.rm = T)
```

```
## [1] 50.5
```

Caso tente-se rodar apenas o nome da variável gera-se um erro:

```
# CAP
```

Isso ocorre porque quando se roda o nome, o R procura um objeto com este nome no ambiente, e não encontra, gerando o erro. Para que se possa chamar variáveis por nome sem $, precisa-se utilizar a função attach, que fixa o dado no ambiente:

```
attach(dados)
```

Agora, chamar o nome da variável não gera erro:

```
CAP
```

```
##  [1] 51.2 52.2 22.6 56.0 45.2 58.4 62.9 53.8 43.7 46.5 41.0 47.7 43.7 47.3
## [15] 52.3 49.8 46.0 56.1 38.4 51.7 47.0 55.3 54.2 58.2 50.1 52.6 42.4 50.9
## [29] 55.1 37.4 48.8 50.7 52.3 47.0 57.5 54.7 50.6   NA 57.0 51.7 46.7 57.8
## [43] 50.8 51.0 56.6 53.2 51.3 53.2 54.0 55.8 59.0 40.5 58.9   NA 53.6 57.7
## [57] 55.5   NA 24.2 29.7 50.9 53.6 50.5   NA 52.4   NA   NA 65.5 52.8 24.5
## [71] 56.1 53.5 55.4 53.1 49.3 52.1 49.3 49.8 57.4 58.1 54.0 50.2
```

porém, quando se trabalha com mais de um dataframe, ou com scripts longos, fixar dados pode gerar erros e atrapalhar o fluxo do script. Por isso a maioria dos usuários não recomenda o uso do attach.

Para desafixar dados, basta utilizar detach:

```
detach(dados)
```

Uma outra forma de selecionar a coluna CAP e utilizando []; utilizando head(), percebe-se que a coluna CAP está na posição 4:

```
head(dados)
```

CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA
3656	101	101	51.2	21.5	Normal
3656	101	103	52.2	20.9	Normal
3656	101	201	22.6	NA	Outros Fustes
3656	101	203	56.0	21.7	Normal
3656	101	205	45.2	20.5	Normal
3656	101	302	58.4	21.6	Dominante

Portando, pode-se utilizar [] para selecionar a coluna CAP, posicionando o número 4 do lado direito da virgula, indicando posição de coluna:

```
dados[ , 4]
```

```
##  [1] 51.2 52.2 22.6 56.0 45.2 58.4 62.9 53.8 43.7 46.5 41.0 47.7 43.7 47.3
## [15] 52.3 49.8 46.0 56.1 38.4 51.7 47.0 55.3 54.2 58.2 50.1 52.6 42.4 50.9
```

```
## [29] 55.1 37.4 48.8 50.7 52.3 47.0 57.5 54.7 50.6   NA 57.0 51.7 46.7 57.8
## [43] 50.8 51.0 56.6 53.2 51.3 53.2 54.0 55.8 59.0 40.5 58.9   NA 53.6 57.7
## [57] 55.5   NA 24.2 29.7 50.9 53.6 50.5   NA 52.4   NA   NA 65.5 52.8 24.5
## [71] 56.1 53.5 55.4 53.1 49.3 52.1 49.3 49.8 57.4 58.1 54.0 50.2
```

Como o lado esquerdo está vazio, todas as linhas são mostradas.

Pode-se utilizar o argumento drop=F, para se manter a classe de dataframe, ao invés de converte-la em vetor:

```
dados[ , 4, drop = F]
```

CAP
51.2
52.2
22.6
56.0
45.2
58.4
62.9
53.8
43.7
46.5
41.0
47.7
43.7
47.3
52.3
49.8
46.0
56.1
38.4
51.7
47.0
55.3
54.2
58.2
50.1
52.6
42.4
50.9
55.1
37.4
48.8
50.7
52.3
47.0
57.5
54.7
50.6
NA
57.0
51.7
46.7
57.8
50.8

CAP
51.0
56.6
53.2
51.3
53.2
54.0
55.8
59.0
40.5
58.9
NA
53.6
57.7
55.5
NA
24.2
29.7
50.9
53.6
50.5
NA
52.4
NA
NA
65.5
52.8
24.5
56.1
53.5
55.4
53.1
49.3
52.1
49.3
49.8
57.4
58.1
54.0
50.2

Para mostrar apenas as 10 primeiras linhas, e a coluna 4, utiliza-se o seguinte comando:

```
dados[1:10 , 4]
```

```
##  [1] 51.2 52.2 22.6 56.0 45.2 58.4 62.9 53.8 43.7 46.5
```

Utilizando [] também é possível especificar uma ou mais colunas por nome, porém este deve estar entre aspas:

```
dados["CAP"]
```

CAP
51.2
52.2

CAP
22.6
56.0
45.2
58.4
62.9
53.8
43.7
46.5
41.0
47.7
43.7
47.3
52.3
49.8
46.0
56.1
38.4
51.7
47.0
55.3
54.2
58.2
50.1
52.6
42.4
50.9
55.1
37.4
48.8
50.7
52.3
47.0
57.5
54.7
50.6
NA
57.0
51.7
46.7
57.8
50.8
51.0
56.6
53.2
51.3
53.2
54.0
55.8
59.0
40.5
58.9
NA

CAP
53.6
57.7
55.5
NA
24.2
29.7
50.9
53.6
50.5
NA
52.4
NA
NA
65.5
52.8
24.5
56.1
53.5
55.4
53.1
49.3
52.1
49.3
49.8
57.4
58.1
54.0
50.2

Para selecionar mais de uma coluna, os nomes devem estar concatenados com a função c():

```
dados[c("CAP" , "ALT1")]
```

CAP	ALT1
51.2	21.5
52.2	20.9
22.6	NA
56.0	21.7
45.2	20.5
58.4	21.6
62.9	NA
53.8	NA
43.7	NA
46.5	NA
41.0	NA
47.7	20.1
43.7	NA
47.3	20.4
52.3	21.1
49.8	20.6
46.0	NA
56.1	21.5

CAP	ALT1
38.4	NA
51.7	NA
47.0	NA
55.3	NA
54.2	NA
58.2	NA
50.1	NA
52.6	21.4
42.4	19.7
50.9	20.9
55.1	21.4
37.4	NA
48.8	NA
50.7	NA
52.3	NA
47.0	NA
57.5	20.5
54.7	21.1
50.6	NA
NA	NA
57.0	22.2
51.7	NA
46.7	NA
57.8	21.6
50.8	NA
51.0	29.2
56.6	30.6
53.2	28.8
51.3	29.0
53.2	NA
54.0	NA
55.8	NA
59.0	30.2
40.5	NA
58.9	32.5
NA	NA
53.6	28.9
57.7	29.8
55.5	29.7
NA	NA
24.2	NA
29.7	NA
50.9	NA
53.6	NA
50.5	NA
NA	NA
52.4	29.5
NA	NA
NA	NA
65.5	30.8
52.8	28.6
24.5	NA

CAP	ALT1
56.1	27.8
53.5	NA
55.4	NA
53.1	27.7
49.3	26.8
52.1	30.2
49.3	27.8
49.8	NA
57.4	29.1
58.1	29.3
54.0	NA
50.2	NA

8.2) Logica e subsetting (Filtrar)

quando se faz testes lógicos entre dataframes, utilizando operadores lógicos, como >, < ==, gera-se um vetor lógico (ou booleano). Por exemplo, se pergunta-se quais dados de CAP são maiores que 50, gera-se o seguinte vetor:

```
dados$CAP > 50
```

```
##  [1]  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [12] FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE
## [23]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE  TRUE
## [34] FALSE  TRUE  TRUE  TRUE    NA  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
## [45]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE    NA  TRUE
## [56]  TRUE  TRUE    NA FALSE FALSE  TRUE  TRUE  TRUE    NA  TRUE    NA
## [67]    NA  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE
## [78] FALSE  TRUE  TRUE  TRUE  TRUE
```

Caso um vetor lógico seja inserido dentro de [], apenas as linhas em que se tem o valor TRUE são mantidas.

Com este raciocínio pode-se filtrar o dado. por exemplo:

```
dados[dados$CAP > 50 , ]
```

	CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA
1	3656	101	101	51.2	21.5	Normal
2	3656	101	103	52.2	20.9	Normal
4	3656	101	203	56.0	21.7	Normal
6	3656	101	302	58.4	21.6	Dominante
7	3656	101	304	62.9	NA	Bifurcadaacima1.30
8	3656	101	306	53.8	NA	Inclinada
15	3656	102	204	52.3	21.1	Normal
18	3656	102	305	56.1	21.5	Dominante
20	3656	102	403	51.7	NA	Normal
22	3656	102	501	55.3	NA	Normal
23	3656	102	503	54.2	NA	Normal
24	3656	103	101	58.2	NA	Bifurcadaacima1.30
25	3656	103	201	50.1	NA	Bifurcadaabaixo1.30
26	3656	103	202	52.6	21.4	Normal
28	3656	103	302	50.9	20.9	Normal
29	3656	103	304	55.1	21.4	Dominante

	CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA
32	3656	103	402	50.7	NA	Normal
33	3656	103	404	52.3	NA	Normal
35	3656	104	101	57.5	20.5	Normal
36	3656	104	201	54.7	21.1	Normal
37	3656	104	203	50.6	NA	Bifurcadaacima1.30
NA	NA	NA	NA	NA	NA	NA
39	3656	104	302	57.0	22.2	Normal
40	3656	104	304	51.7	NA	Normal
42	3656	104	404	57.8	21.6	Dominante
43	3656	104	502	50.8	NA	Normal
44	3654	101	101	51.0	29.2	Normal
45	3654	101	202	56.6	30.6	Normal
46	3654	101	204	53.2	28.8	Normal
47	3654	101	302	51.3	29.0	Normal
48	3654	101	304	53.2	NA	Normal
49	3654	101	401	54.0	NA	Normal
50	3654	101	403	55.8	NA	Normal
51	3654	101	405	59.0	30.2	Dominante
53	3654	102	101	58.9	32.5	Normal
NA.1	NA	NA	NA	NA	NA	NA
55	3654	102	202	53.6	28.9	Normal
56	3654	102	204	57.7	29.8	Normal
57	3654	102	302	55.5	29.7	Normal
NA.2	NA	NA	NA	NA	NA	NA
61	3654	102	403	50.9	NA	Normal
62	3654	102	405	53.6	NA	Normal
63	3654	102	502	50.5	NA	Normal
NA.3	NA	NA	NA	NA	NA	NA
65	3654	103	201	52.4	29.5	Normal
NA.4	NA	NA	NA	NA	NA	NA
NA.5	NA	NA	NA	NA	NA	NA
68	3654	103	303	65.5	30.8	Dominante
69	3654	103	305	52.8	28.6	Normal
71	3654	103	403	56.1	27.8	Normal
72	3654	103	405	53.5	NA	Normal
73	3654	103	502	55.4	NA	Normal
74	3654	104	101	53.1	27.7	Normal
76	3654	104	202	52.1	30.2	Normal
79	3654	104	304	57.4	29.1	Dominante
80	3654	104	402	58.1	29.3	Dominante
81	3654	104	404	54.0	NA	Normal
82	3654	104	501	50.2	NA	Normal

Com isso, fatores são muito uteis quando se deseja filtrar os dados. Por exemplo, a variável DESCCATEGORIA indica a qualidade da arvore, como dominante, normal, falha, etc. Pode-se verificar isso com a função levels:

```
levels(dados$DESCCATEGORIA)
```

```
## [1] "Bifurcadaabaixo1.30" "Bifurcadaacima1.30"  "Dominada"
## [4] "Dominante"           "Falha"               "Inclinada"
## [7] "Normal"              "Outros Fustes"
```

Seguindo o raciocínio, pode-se selecionar apenas as arvores dominantes:

```
dados[dados$DESCCATEGORIA == "Dominante", ]
```

	CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA
6	3656	101	302	58.4	21.6	Dominante
18	3656	102	305	56.1	21.5	Dominante
29	3656	103	304	55.1	21.4	Dominante
42	3656	104	404	57.8	21.6	Dominante
51	3654	101	405	59.0	30.2	Dominante
68	3654	103	303	65.5	30.8	Dominante
79	3654	104	304	57.4	29.1	Dominante
80	3654	104	402	58.1	29.3	Dominante

Aqui seleciona-se apenas as linhas em que DESCCATEGORIA é igual a dominante, e todas as colunas.

Quando se realiza subsetting, é possível utilizar varias condições, deixando o filtro mais específico. O uso dos operadores & (E) e | (ou) são muito utilizados, pois especificam mutualidade exclusiva:

No exemplo a seguir mantem-se as linhas apenas onde CODTALHAO é igual a 3656 E CODPARCELA é igual a 101:

```
dados[dados$CODTALHAO == 3656 & dados$CODPARCELA == 101, ]
```

CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA
3656	101	101	51.2	21.5	Normal
3656	101	103	52.2	20.9	Normal
3656	101	201	22.6	NA	Outros Fustes
3656	101	203	56.0	21.7	Normal
3656	101	205	45.2	20.5	Normal
3656	101	302	58.4	21.6	Dominante
3656	101	304	62.9	NA	Bifurcadaacima1.30
3656	101	306	53.8	NA	Inclinada
3656	101	402	43.7	NA	Inclinada
3656	101	404	46.5	NA	Normal
3656	101	502	41.0	NA	Normal

Isso indica que apenas o talhão 3656 será mantido, e dessas linhas, apenas as que a parcela e 101 serão mantidas.

Esta condição é exclusiva, ou seja, mesmo que 101 apareça em outros talhões, só serão exibidos os dados do talhão 3656.

Já no próximo exemplo, o operador utilizado foi |, significando, linhas do talhão 3656, OU parcela 101.

```
dados[dados$CODTALHAO == 3656 | dados$CODPARCELA == 101, ]
```

CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA
3656	101	101	51.2	21.5	Normal
3656	101	103	52.2	20.9	Normal
3656	101	201	22.6	NA	Outros Fustes
3656	101	203	56.0	21.7	Normal
3656	101	205	45.2	20.5	Normal
3656	101	302	58.4	21.6	Dominante
3656	101	304	62.9	NA	Bifurcadaacima1.30

CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA
3656	101	306	53.8	NA	Inclinada
3656	101	402	43.7	NA	Inclinada
3656	101	404	46.5	NA	Normal
3656	101	502	41.0	NA	Normal
3656	102	101	47.7	20.1	Normal
3656	102	103	43.7	NA	Bifurcadaacima1.30
3656	102	202	47.3	20.4	Normal
3656	102	204	52.3	21.1	Normal
3656	102	301	49.8	20.6	Normal
3656	102	303	46.0	NA	Normal
3656	102	305	56.1	21.5	Dominante
3656	102	401	38.4	NA	Normal
3656	102	403	51.7	NA	Normal
3656	102	405	47.0	NA	Normal
3656	102	501	55.3	NA	Normal
3656	102	503	54.2	NA	Normal
3656	103	101	58.2	NA	Bifurcadaacima1.30
3656	103	201	50.1	NA	Bifurcadaabaixo1.30
3656	103	202	52.6	21.4	Normal
3656	103	204	42.4	19.7	Normal
3656	103	302	50.9	20.9	Normal
3656	103	304	55.1	21.4	Dominante
3656	103	305	37.4	NA	Outros Fustes
3656	103	401	48.8	NA	Bifurcadaabaixo1.30
3656	103	402	50.7	NA	Normal
3656	103	404	52.3	NA	Normal
3656	103	502	47.0	NA	Normal
3656	104	101	57.5	20.5	Normal
3656	104	201	54.7	21.1	Normal
3656	104	203	50.6	NA	Bifurcadaacima1.30
3656	104	205	NA	NA	Falha
3656	104	302	57.0	22.2	Normal
3656	104	304	51.7	NA	Normal
3656	104	402	46.7	NA	Bifurcadaacima1.30
3656	104	404	57.8	21.6	Dominante
3656	104	502	50.8	NA	Normal
3654	101	101	51.0	29.2	Normal
3654	101	202	56.6	30.6	Normal
3654	101	204	53.2	28.8	Normal
3654	101	302	51.3	29.0	Normal
3654	101	304	53.2	NA	Normal
3654	101	401	54.0	NA	Normal
3654	101	403	55.8	NA	Normal
3654	101	405	59.0	30.2	Dominante
3654	101	502	40.5	NA	Normal

Ou seja, pode ser qualquer parcela, desde que seja do talhão 3656. E pode ser qualquer talhão, desde que a parcela seja 101

O talhão 3654 possui a parcela 101 da mesma forma, este mesmo resultado pode ser obtido com a função subset:

```r
subset(dados, CODTALHAO == 3656 & CODPARCELA == 101)
```

CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA
3656	101	101	51.2	21.5	Normal
3656	101	103	52.2	20.9	Normal
3656	101	201	22.6	NA	Outros Fustes
3656	101	203	56.0	21.7	Normal
3656	101	205	45.2	20.5	Normal
3656	101	302	58.4	21.6	Dominante
3656	101	304	62.9	NA	Bifurcadaacima1.30
3656	101	306	53.8	NA	Inclinada
3656	101	402	43.7	NA	Inclinada
3656	101	404	46.5	NA	Normal
3656	101	502	41.0	NA	Normal

```r
subset(dados, CODTALHAO == 3656 | CODPARCELA == 101)
```

CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA
3656	101	101	51.2	21.5	Normal
3656	101	103	52.2	20.9	Normal
3656	101	201	22.6	NA	Outros Fustes
3656	101	203	56.0	21.7	Normal
3656	101	205	45.2	20.5	Normal
3656	101	302	58.4	21.6	Dominante
3656	101	304	62.9	NA	Bifurcadaacima1.30
3656	101	306	53.8	NA	Inclinada
3656	101	402	43.7	NA	Inclinada
3656	101	404	46.5	NA	Normal
3656	101	502	41.0	NA	Normal
3656	102	101	47.7	20.1	Normal
3656	102	103	43.7	NA	Bifurcadaacima1.30
3656	102	202	47.3	20.4	Normal
3656	102	204	52.3	21.1	Normal
3656	102	301	49.8	20.6	Normal
3656	102	303	46.0	NA	Normal
3656	102	305	56.1	21.5	Dominante
3656	102	401	38.4	NA	Normal
3656	102	403	51.7	NA	Normal
3656	102	405	47.0	NA	Normal
3656	102	501	55.3	NA	Normal
3656	102	503	54.2	NA	Normal
3656	103	101	58.2	NA	Bifurcadaacima1.30
3656	103	201	50.1	NA	Bifurcadaabaixo1.30
3656	103	202	52.6	21.4	Normal
3656	103	204	42.4	19.7	Normal
3656	103	302	50.9	20.9	Normal
3656	103	304	55.1	21.4	Dominante
3656	103	305	37.4	NA	Outros Fustes
3656	103	401	48.8	NA	Bifurcadaabaixo1.30
3656	103	402	50.7	NA	Normal
3656	103	404	52.3	NA	Normal
3656	103	502	47.0	NA	Normal

CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA
3656	104	101	57.5	20.5	Normal
3656	104	201	54.7	21.1	Normal
3656	104	203	50.6	NA	Bifurcadaacima1.30
3656	104	205	NA	NA	Falha
3656	104	302	57.0	22.2	Normal
3656	104	304	51.7	NA	Normal
3656	104	402	46.7	NA	Bifurcadaacima1.30
3656	104	404	57.8	21.6	Dominante
3656	104	502	50.8	NA	Normal
3654	101	101	51.0	29.2	Normal
3654	101	202	56.6	30.6	Normal
3654	101	204	53.2	28.8	Normal
3654	101	302	51.3	29.0	Normal
3654	101	304	53.2	NA	Normal
3654	101	401	54.0	NA	Normal
3654	101	403	55.8	NA	Normal
3654	101	405	59.0	30.2	Dominante
3654	101	502	40.5	NA	Normal

9) Modificando variáveis, fatores e criando novos objetos

9.1) Criar variáveis e objetos

Para se criar novas variáveis em um dataframe, basta utilizar $ ou [], digitando o nome da variável que se deseja criar, seguido de <- ou =, e o que irá compor a nova variável. Por exemplo, a conversão de circunferência para diâmetro:

```
dados$DAP <- dados$CAP / pi
dados["DAP"] <- dados$CAP / pi
head(dados)
```

CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA	DAP
3656	101	101	51.2	21.5	Normal	16.297466
3656	101	103	52.2	20.9	Normal	16.615776
3656	101	201	22.6	NA	Outros Fustes	7.193803
3656	101	203	56.0	21.7	Normal	17.825354
3656	101	205	45.2	20.5	Normal	14.387607
3656	101	302	58.4	21.6	Dominante	18.589297

Percebe-se que os dados possuem muitas casas decimais; pode-se utilizar a função round para resolver isto.

No primeiro argumento informa-se o objeto ou variável que deseja-se alterar; e no segundo o número de casas desejadas:

```
dados$DAP <- round(dados$DAP, 4)
head(dados)
```

CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA	DAP
3656	101	101	51.2	21.5	Normal	16.2975
3656	101	103	52.2	20.9	Normal	16.6158

CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA	DAP
3656	101	201	22.6	NA	Outros Fustes	7.1938
3656	101	203	56.0	21.7	Normal	17.8254
3656	101	205	45.2	20.5	Normal	14.3876
3656	101	302	58.4	21.6	Dominante	18.5893

Para se deletar uma variável, utiliza-se NULL:

```
dados$DAP_QUAD <- dados$DAP ^ 2

head(dados)
```

CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA	DAP	DAP_QUAD
3656	101	101	51.2	21.5	Normal	16.2975	265.60851
3656	101	103	52.2	20.9	Normal	16.6158	276.08481
3656	101	201	22.6	NA	Outros Fustes	7.1938	51.75076
3656	101	203	56.0	21.7	Normal	17.8254	317.74489
3656	101	205	45.2	20.5	Normal	14.3876	207.00303
3656	101	302	58.4	21.6	Dominante	18.5893	345.56207

```
dados$DAP_QUAD <- NULL

head(dados)
```

CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA	DAP
3656	101	101	51.2	21.5	Normal	16.2975
3656	101	103	52.2	20.9	Normal	16.6158
3656	101	201	22.6	NA	Outros Fustes	7.1938
3656	101	203	56.0	21.7	Normal	17.8254
3656	101	205	45.2	20.5	Normal	14.3876
3656	101	302	58.4	21.6	Dominante	18.5893

NA, ou Not Avaiable, são valores inválidos, ou vazios no R. Este tipo de dado pode atrapalhar quando se utiliza fórmulas como média, ou na hora de plotar gráficos. Devido a isso, e interessante se ter uma cópia dos dados originais, pode se remove os NAs. Isso pode ser feito com a função na.omit:

```
dados_g <- na.omit(dados)
head(dados_g)
```

	CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA	DAP
1	3656	101	101	51.2	21.5	Normal	16.2975
2	3656	101	103	52.2	20.9	Normal	16.6158
4	3656	101	203	56.0	21.7	Normal	17.8254
5	3656	101	205	45.2	20.5	Normal	14.3876
6	3656	101	302	58.4	21.6	Dominante	18.5893
12	3656	102	101	47.7	20.1	Normal	15.1834

9.2) Converter variáveis

Para se converter variáveis de uma classe para outra, basta utilizar a família de funções "as". Por exemplo, para se transformar uma variável character em uma variável factor, utiliza-se a função as.factor:

```
class(dados$DESCCATEGORIA)
```

```
## [1] "factor"
```

```
dados$DESCCATEGORIA <- as.factor(dados$DESCCATEGORIA)
class(dados$DESCCATEGORIA)
```

```
## [1] "factor"
```

Ou se para transformar uma variável numeric para character, utiliza-se as.character:

```
class(dados$CODTALHAO)
```

```
## [1] "integer"
```

```
dados$CODTALHAO <- as.factor(dados$CODTALHAO)
class(dados$CODTALHAO)
```

```
## [1] "factor"
```

10) Graficos de dispersao, histogramas e boxplot com ggplot2

10.0) Carregar dados e pacotes

```
library(tidyverse)
dados_g <- read.csv2("dados.csv")
dados_g <- na.omit(dados_g)
dados_g$DAP <- dados_g$CAP/pi
head(dados_g)
```

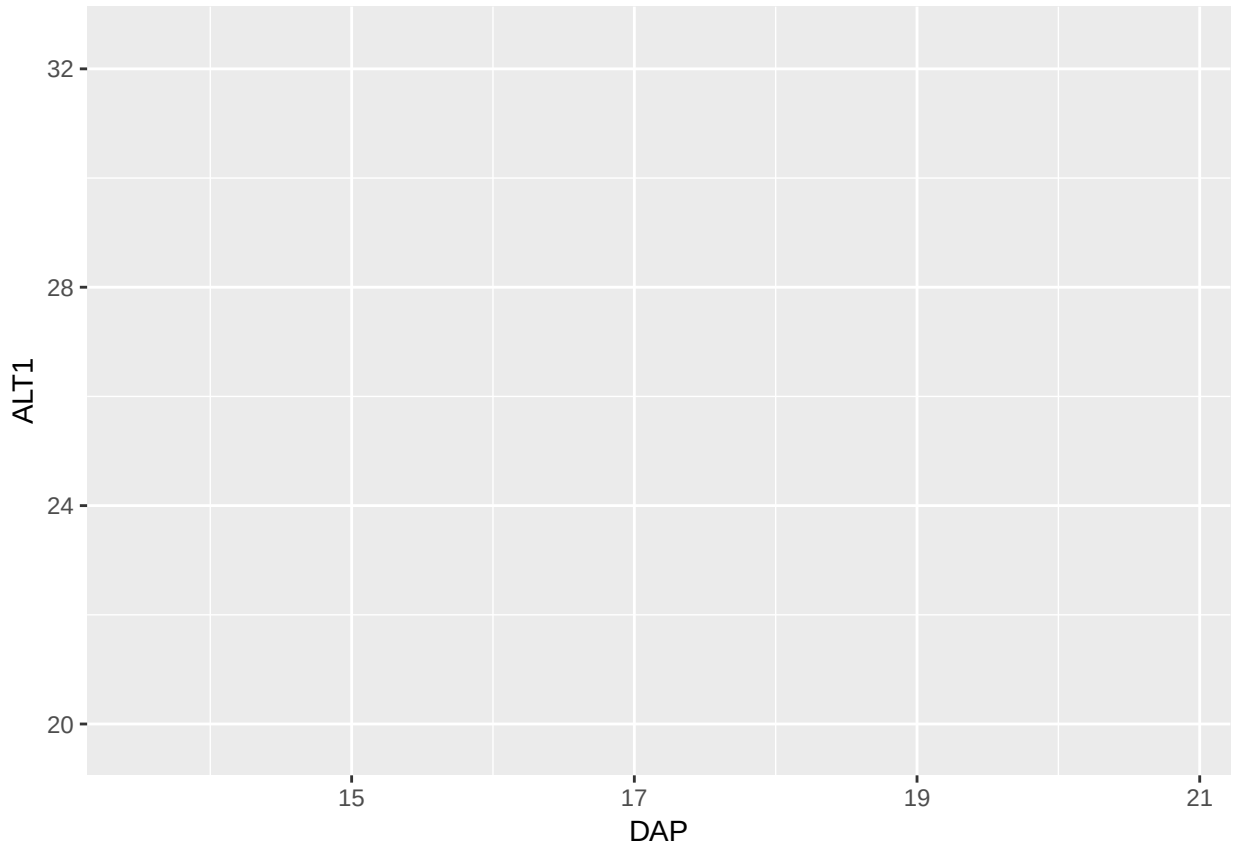
	CODTALHAO	CODPARCELA	CODARVORE	CAP	ALT1	DESCCATEGORIA	DAP
1	3656	101	101	51.2	21.5	Normal	16.29747
2	3656	101	103	52.2	20.9	Normal	16.61578
4	3656	101	203	56.0	21.7	Normal	17.82535
5	3656	101	205	45.2	20.5	Normal	14.38761
6	3656	101	302	58.4	21.6	Dominante	18.58930
12	3656	102	101	47.7	20.1	Normal	15.18338

10.1) Dispersão

Histograma para a altura em função da idade

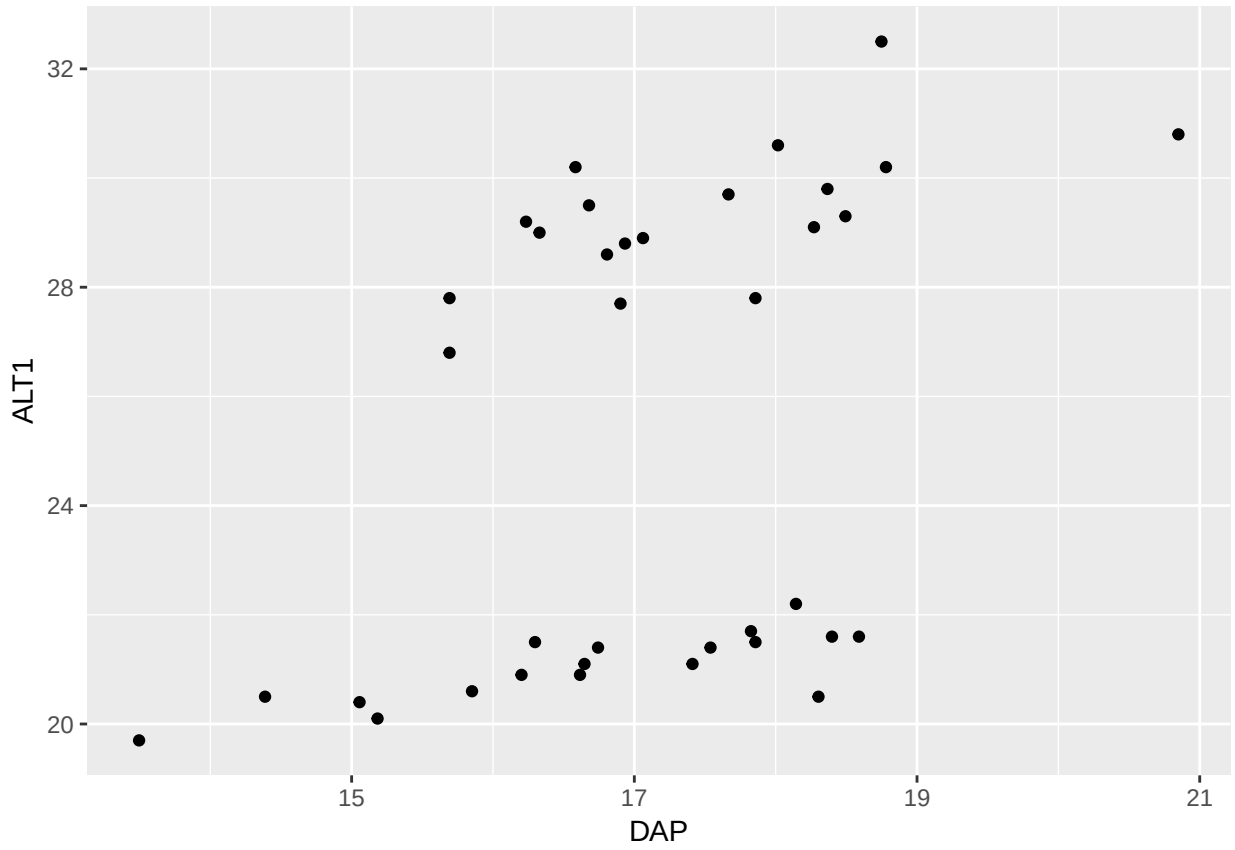
Primeiro cria-se a base do grafico

```
a <- ggplot(dados_g, aes(x = DAP, y = ALT1))
a
```



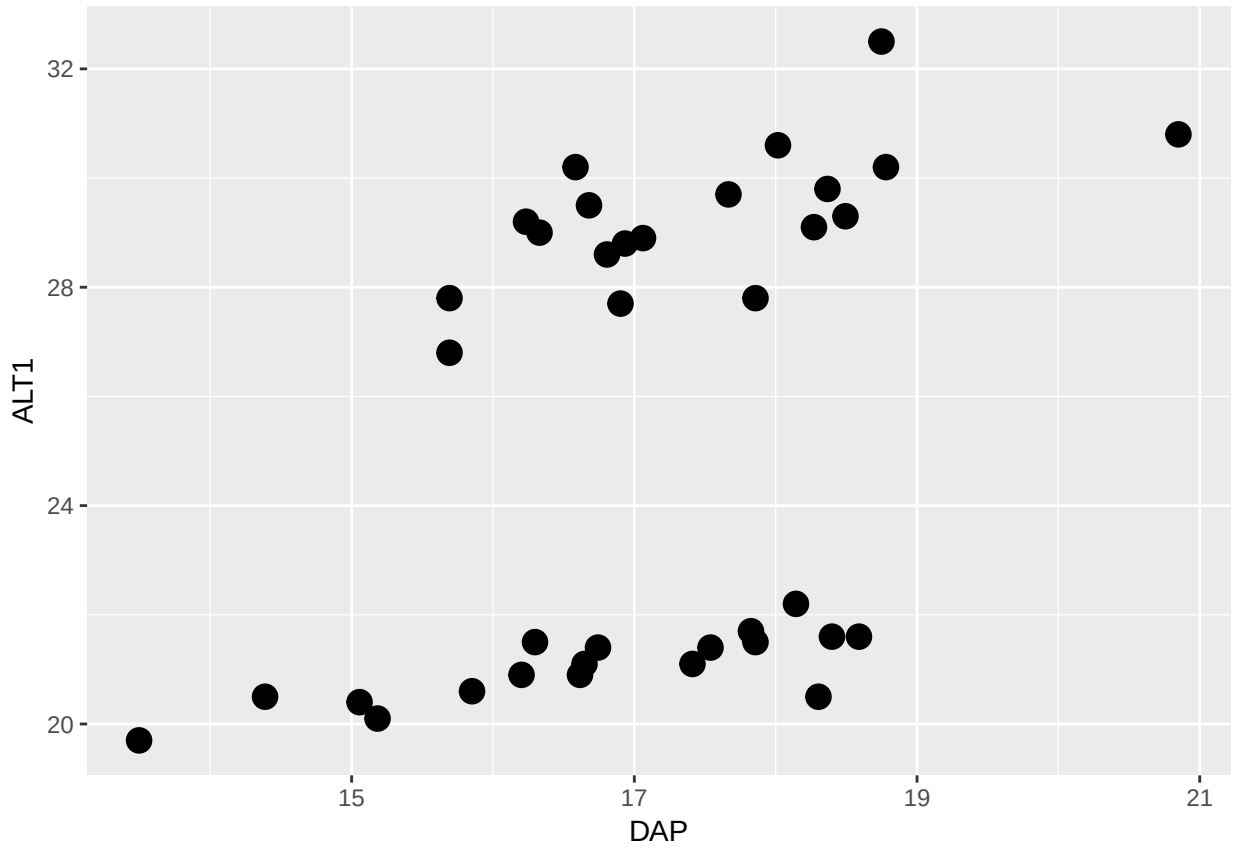
depois adiciona-se outras camadas

```
a + geom_point()
```



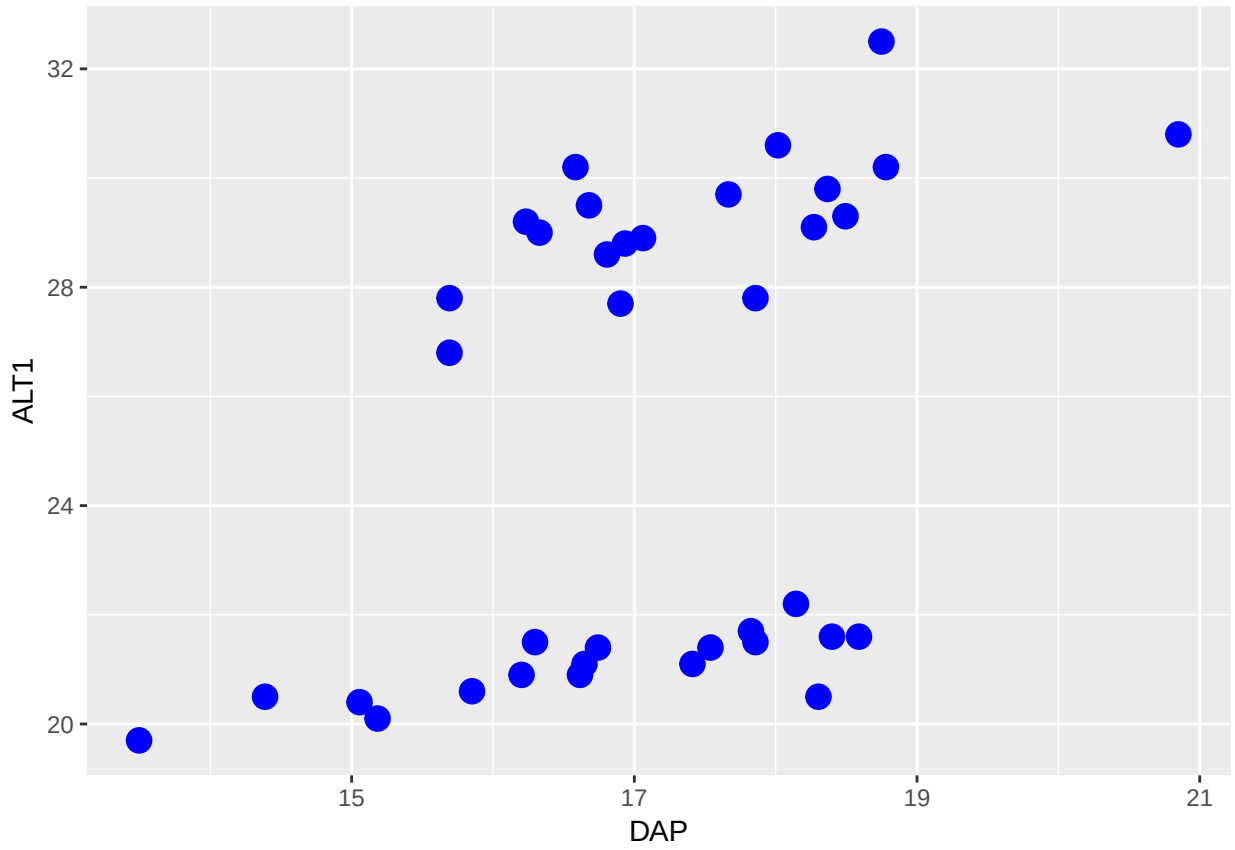
tamanho personalizado

```
a + geom_point( size = 4)
```



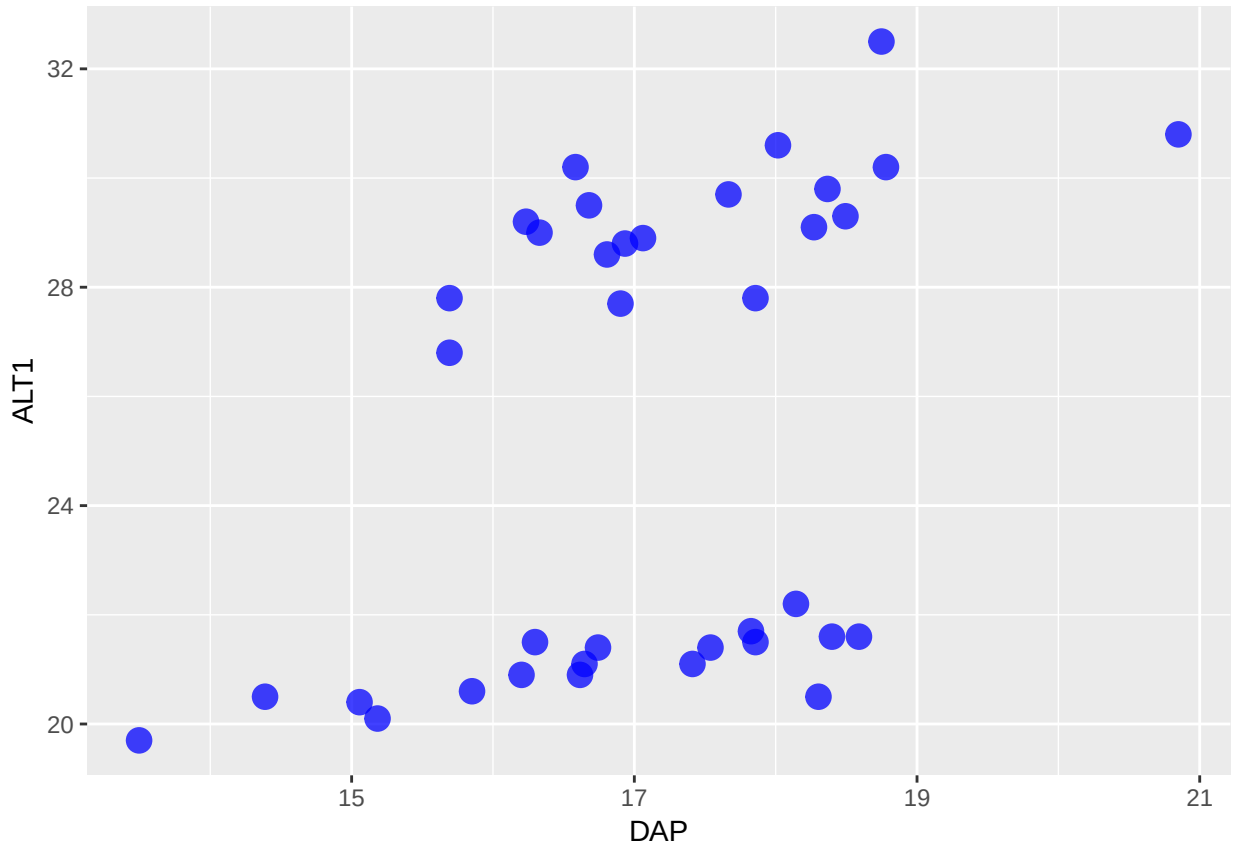
Cor personalizada

```
a + geom_point(size = 4, color = "blue")
```



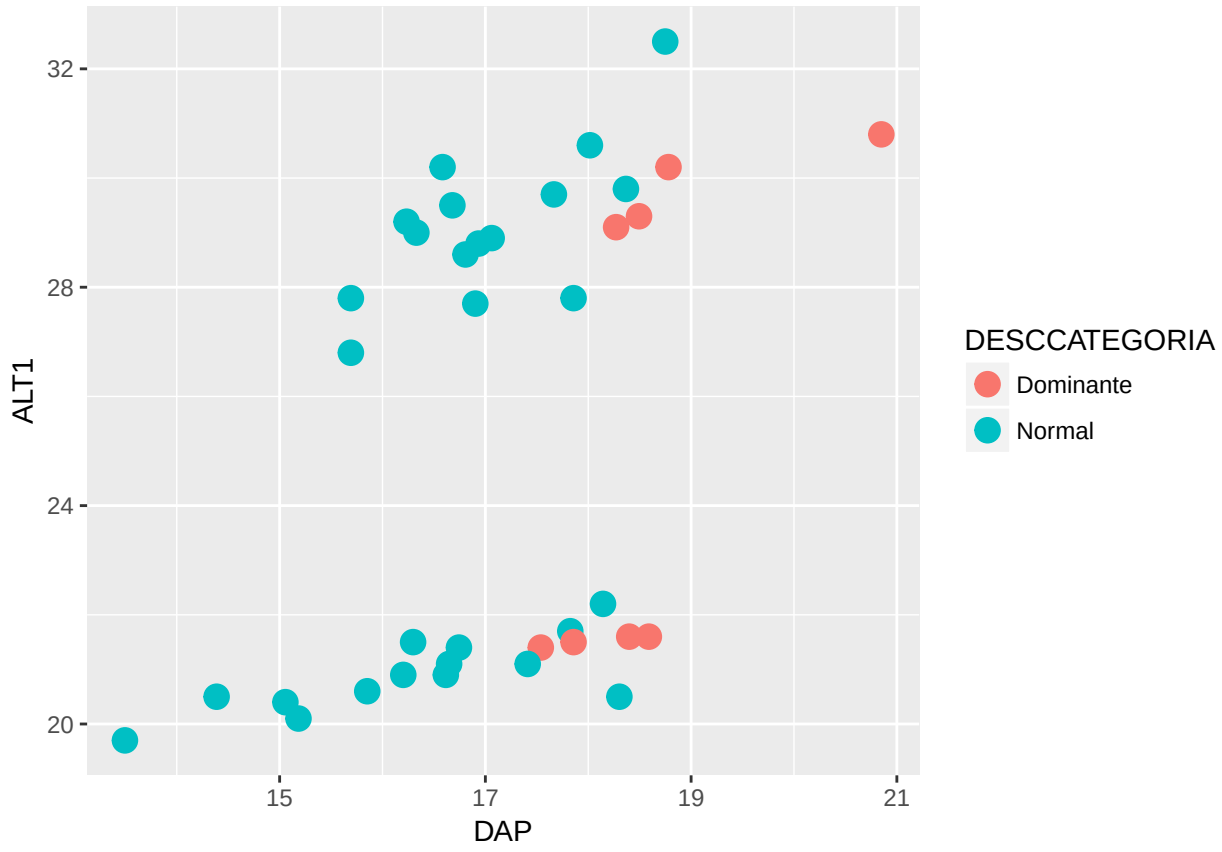
Transparencia nos pontos

```
a + geom_point(size = 4, color = "blue", alpha = .75)
```



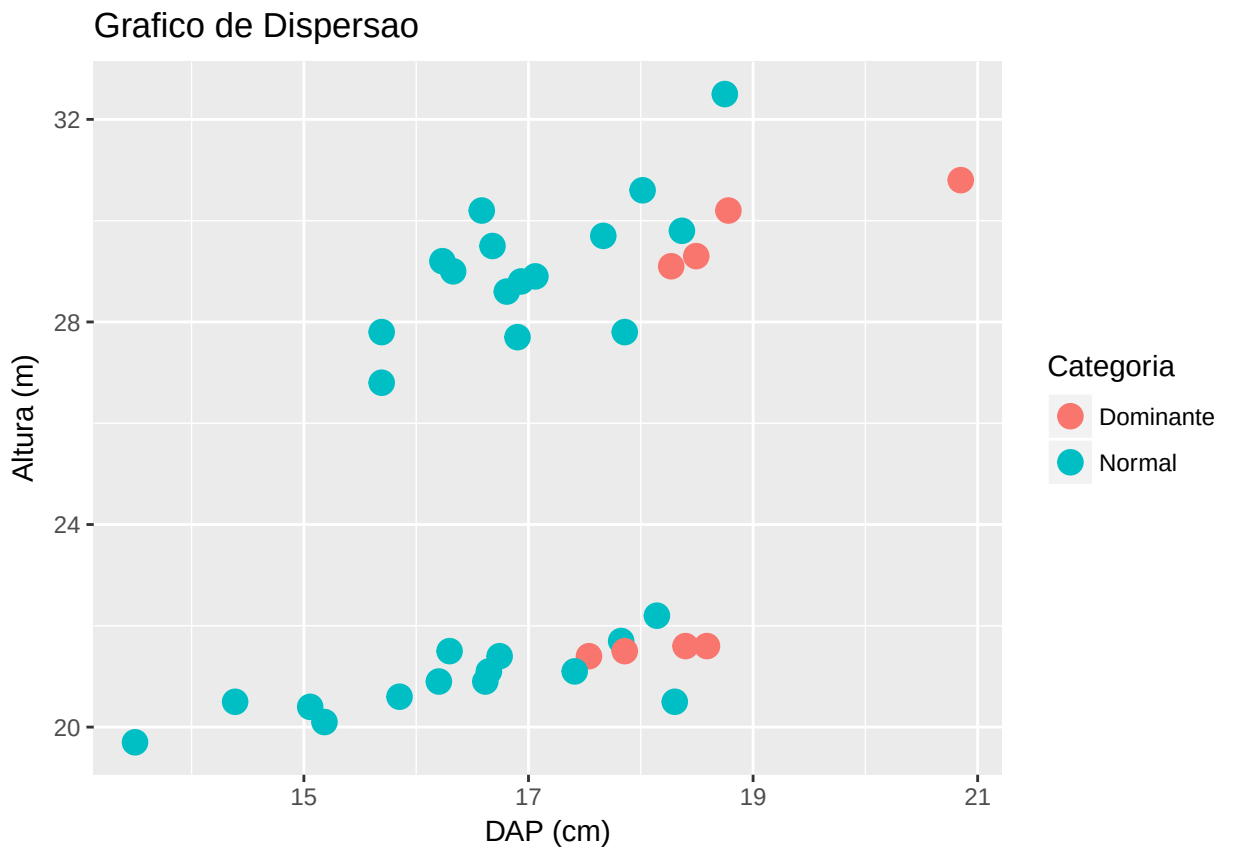
pode-se preencher os dados com uma variavel classificatoria

```
a + geom_point(size = 4, aes(color=DESCCATEGORIA))
```



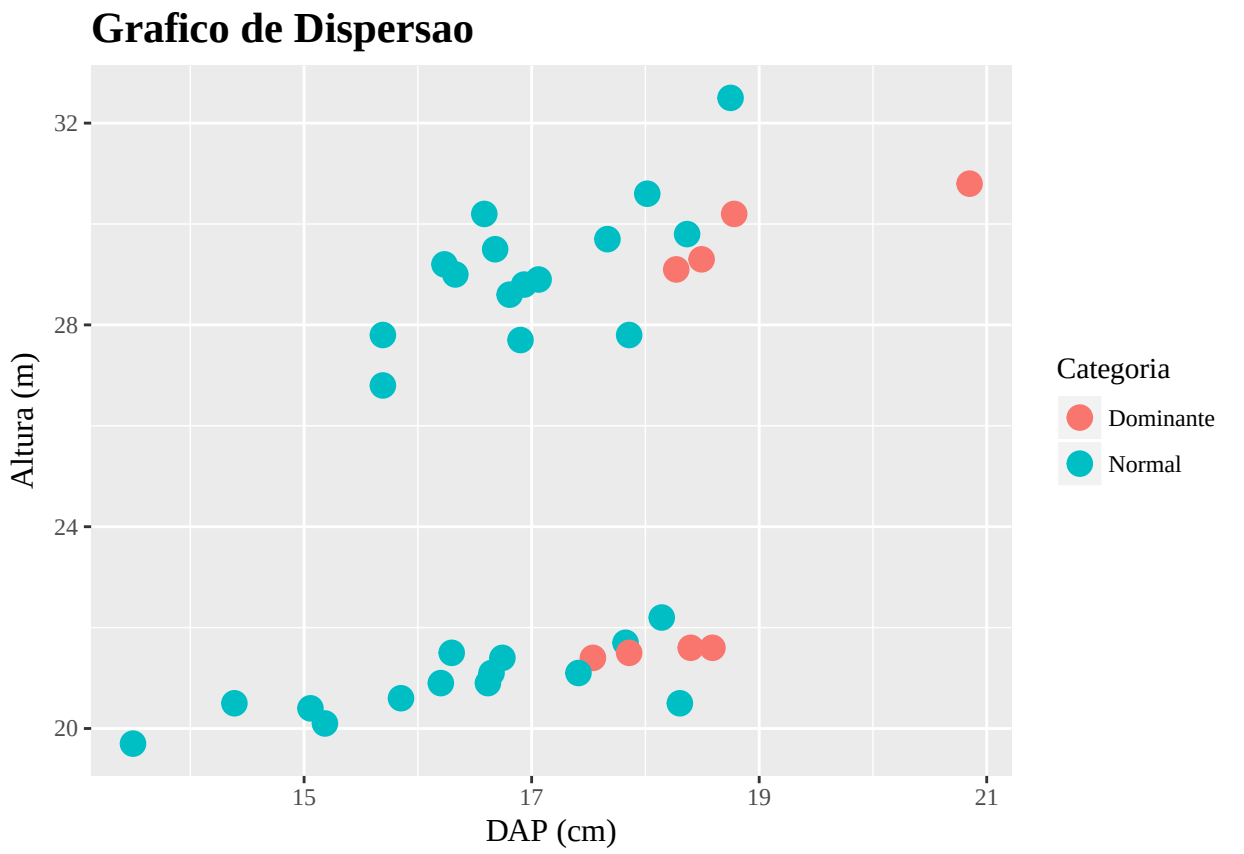
titulo personalizado

```
a + geom_point(size = 4, aes(color=DESCCATEGORIA)) +
  labs(x="DAP (cm)", y="Altura (m)", colour = "Categoria", title = "Grafico de Dispersao")
```



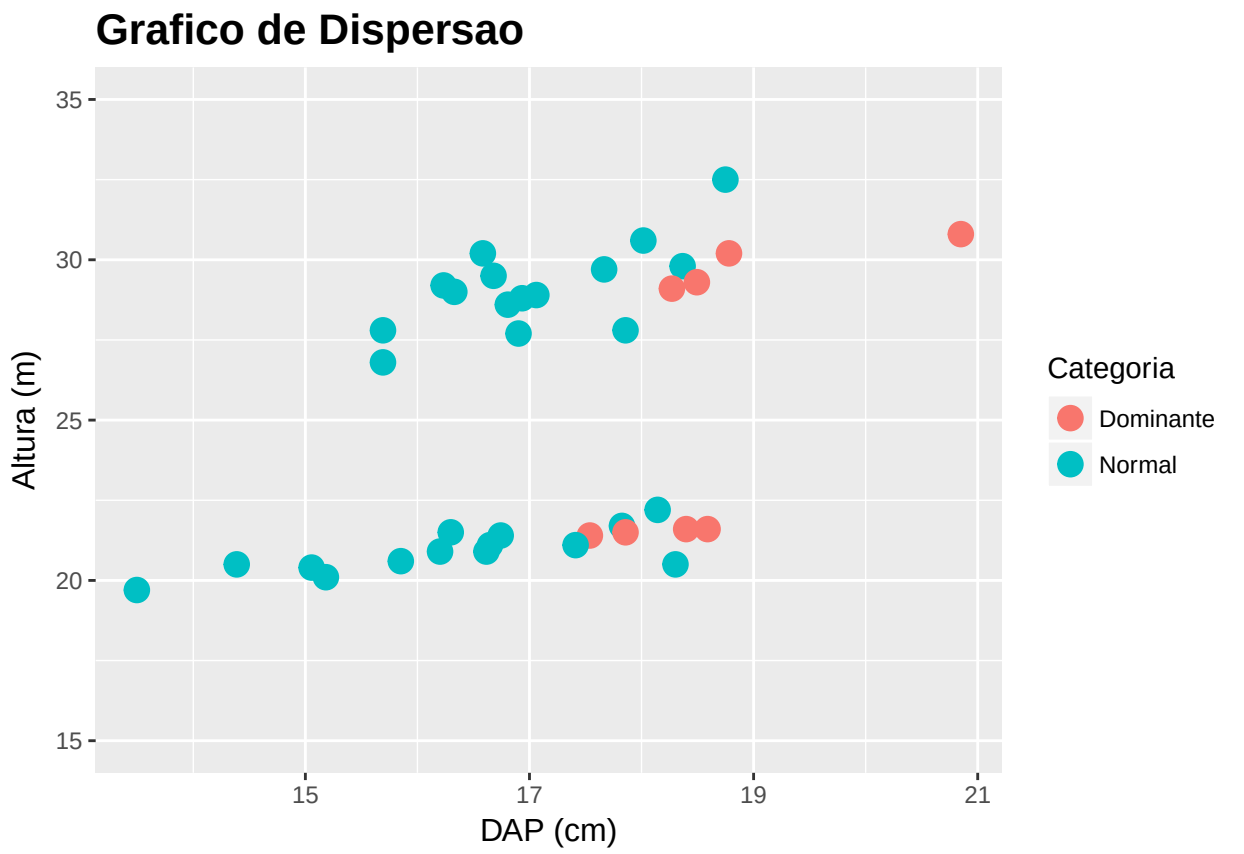
Tamanho e fonte personalizado

```
a + geom_point(size = 4, aes(color=DESCCATEGORIA)) +
  labs(x="DAP (cm)", y="Altura (m)", colour = "Categoria", title = "Grafico de Dispersao") +
theme_gray(base_family = "serif") +
theme(
  plot.title=element_text(size = 16, face="bold", vjust = 0.9),
  axis.title=element_text(size = 12))
```



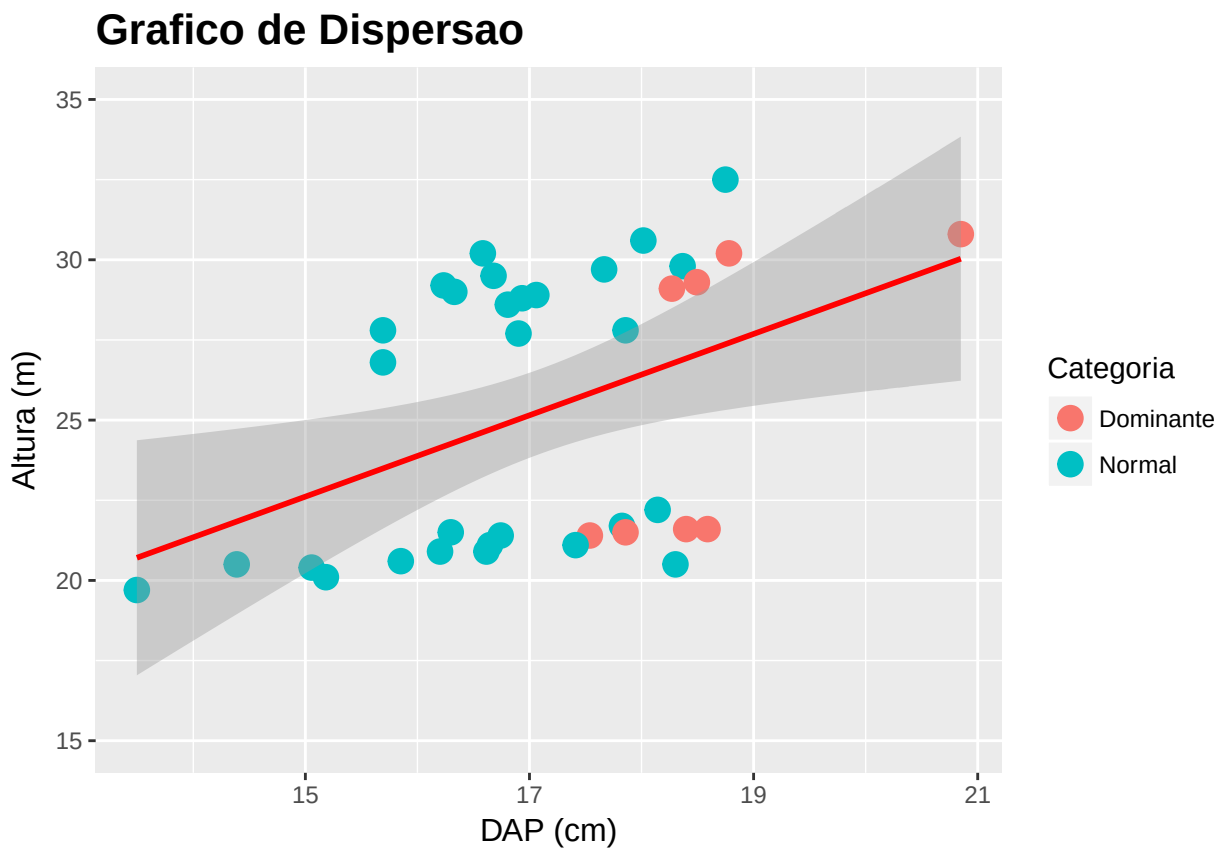
Limites do eixo y personalizado

```
a + geom_point(size = 4, aes(color=DESCCATEGORIA)) +
  labs(x="DAP (cm)", y="Altura (m)", colour = "Categoria", title = "Grafico de Dispersao") +
  theme(
    plot.title=element_text(size = 16, face="bold", vjust = 0.9),
    axis.title=element_text(size = 12)) +
  coord_cartesian(ylim = c(15,35))
```



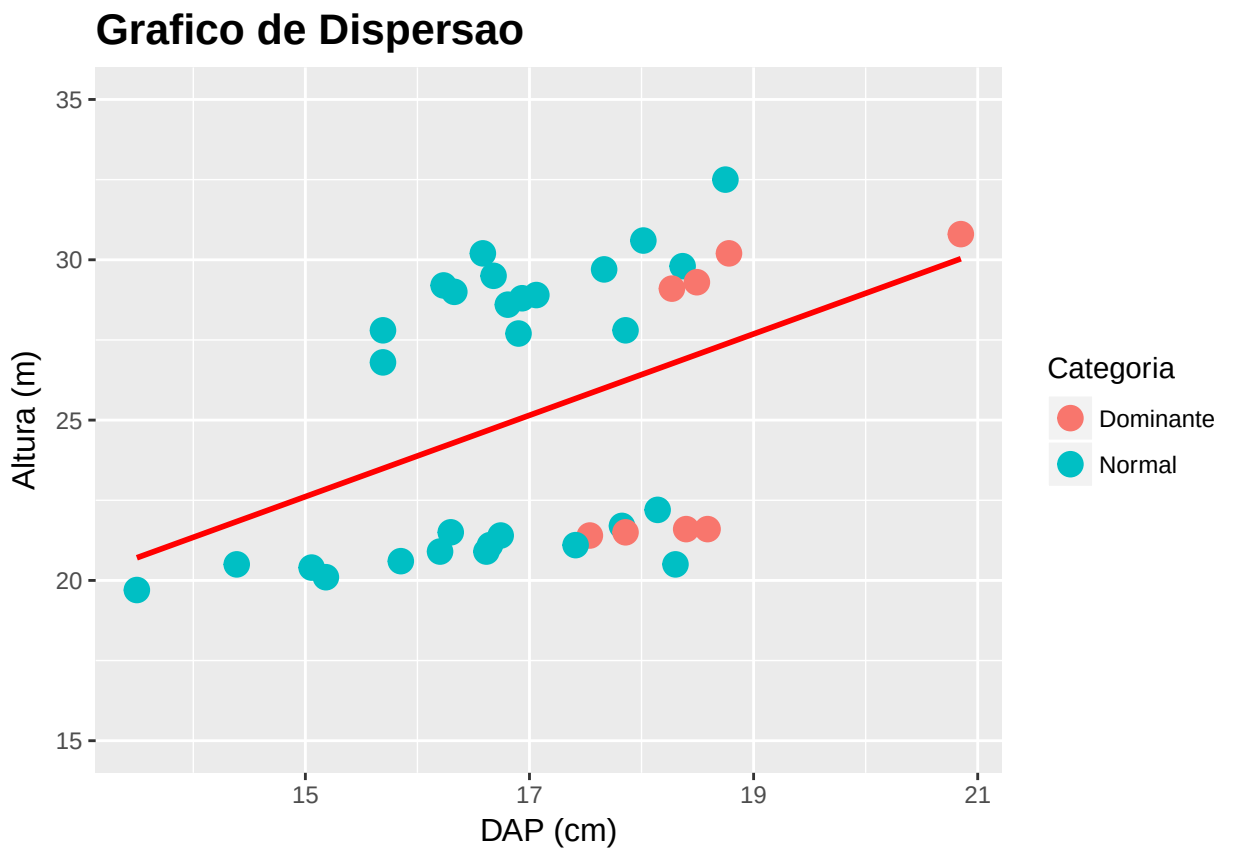
linha de tendencia

```
a  + geom_point(size = 4, aes(color=DESCCATEGORIA)) +
  labs(x="DAP (cm)", y="Altura (m)", colour = "Categoria", title = "Grafico de Dispersao") +
  theme(
    plot.title=element_text(size = 16, face="bold", vjust = 0.9),
    axis.title=element_text(size = 12)) +
  coord_cartesian(ylim = c(15,35)) +
  geom_smooth(method = "lm", color = "red")
```



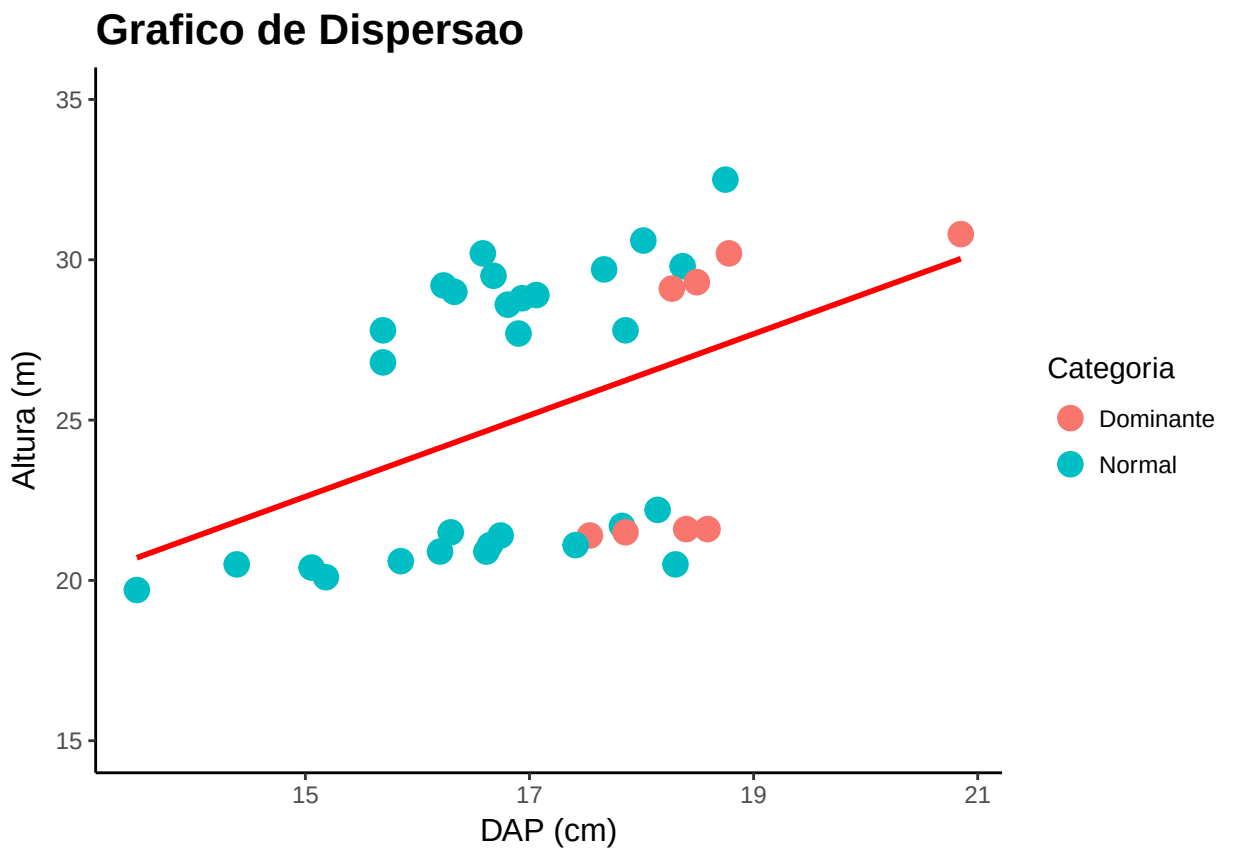
linha de tendencia sem intervalo de confiança

```
a  + geom_point(size = 4, aes(color=DESCCATEGORIA)) +
  labs(x="DAP (cm)", y="Altura (m)", colour = "Categoria", title = "Grafico de Dispersao") +
  theme(
    plot.title=element_text(size = 16, face="bold", vjust = 0.9),
    axis.title=element_text(size = 12)) +
  coord_cartesian(ylim = c(15,35)) +
  geom_smooth(method = "lm", color = "red", se=FALSE)
```



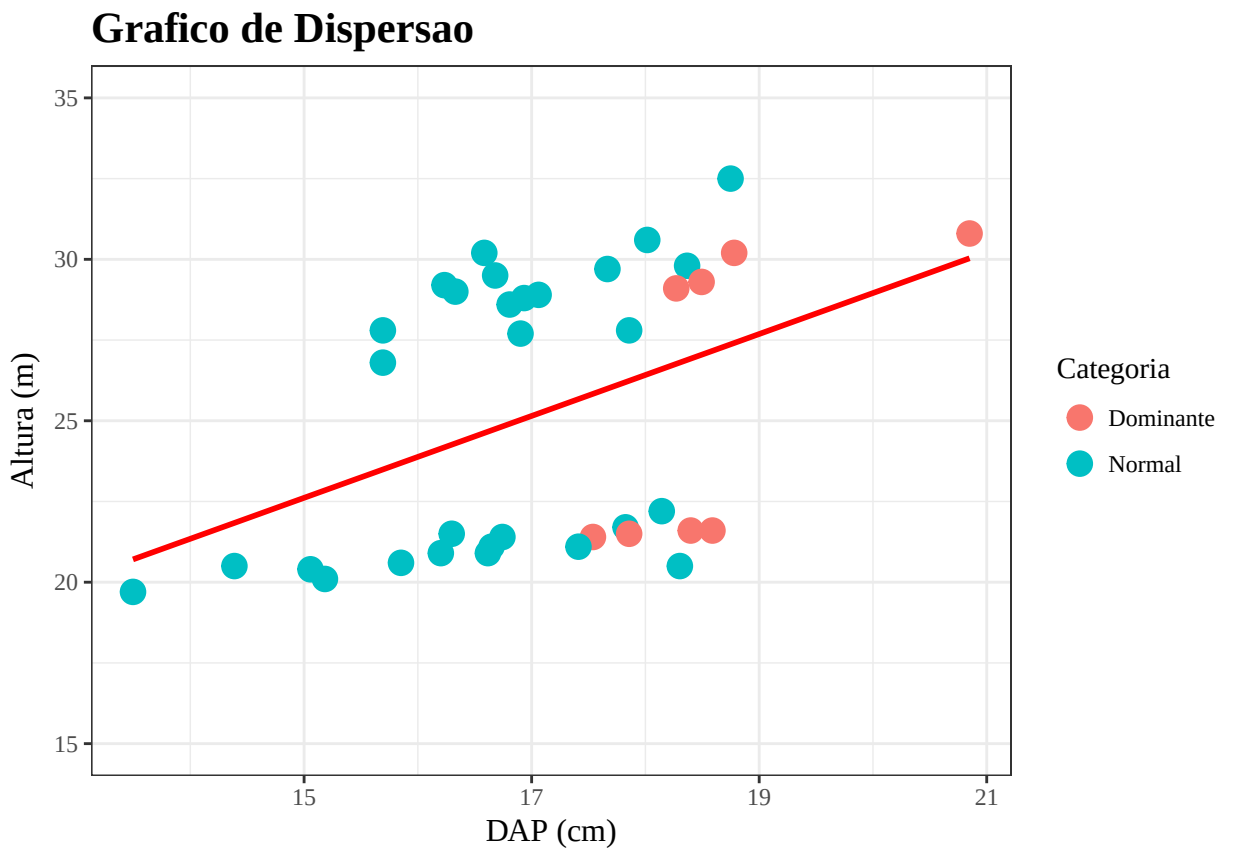
tema diferente

```
a  + geom_point(size = 4, aes(color=DESCCATEGORIA)) +
  labs(x="DAP (cm)", y="Altura (m)", colour = "Categoria", title = "Grafico de Dispersao") +
  theme_classic() +
  theme(
    plot.title=element_text(size = 16, face="bold", vjust = 0.9),
    axis.title=element_text(size = 12)) +
  coord_cartesian(ylim = c(15,35)) +
  geom_smooth(method = "lm", color = "red", se=FALSE)
```



Fonte times new roman

```
a  + geom_point(size = 4, aes(color=DESCCATEGORIA)) +
    labs(x="DAP (cm)", y="Altura (m)", colour = "Categoria", title = "Grafico de Dispersao") +
    theme_bw(base_family = "serif") +
    theme(
      plot.title=element_text(size = 16, face="bold", vjust = 0.9),
      axis.title=element_text(size = 12)) +
    coord_cartesian(ylim = c(15,35)) +
    geom_smooth(method = "lm", color = "red", se=FALSE)
```



Pode-se utilizar subsetting com uma variavel categorica

```
a + geom_point(size = 4, aes(color=DESCCATEGORIA), show.legend = F) +
  labs(x="DAP (cm)", y="Altura (m)", colour = "Categoria", title = "Grafico de Dispersao") +
  theme_bw(base_family = "serif") +
  theme(
    plot.title=element_text(size = 16, face="bold", vjust = 0.9),
    axis.title=element_text(size = 12)) +
  coord_cartesian(ylim = c(15,35)) +
  geom_smooth(method = "lm", color = "red", se=FALSE) +
  facet_grid(.~CODTALHAO)
```

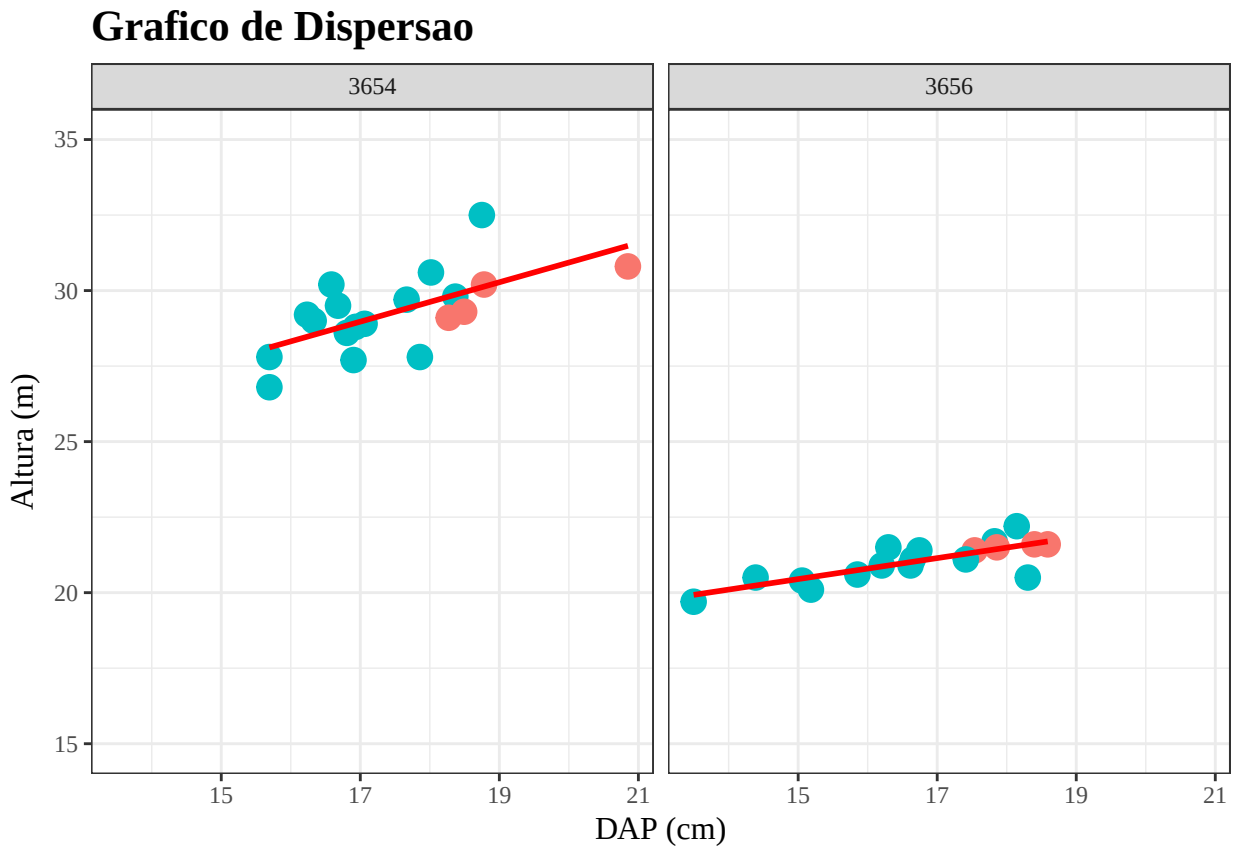


Grafico em escala de cinza para variável discreta

```
a + geom_point(size = 4, aes(color=DESCCATEGORIA)) +
  labs(x="DAP (cm)", y="Altura (m)", colour = "Categoria", title = "Grafico de Dispersao") +
  theme_bw(base_family = "serif") +
  theme(
    plot.title=element_text(size = 16, face="bold", vjust = 0.9),
    axis.title=element_text(size = 12)) +
  coord_cartesian(ylim = c(15,35)) +
  geom_smooth(method = "lm", color = "red", se=FALSE) +
  facet_grid(.~CODTALHAO) +
  scale_colour_grey(start = 0.6, end = 0.2)
```

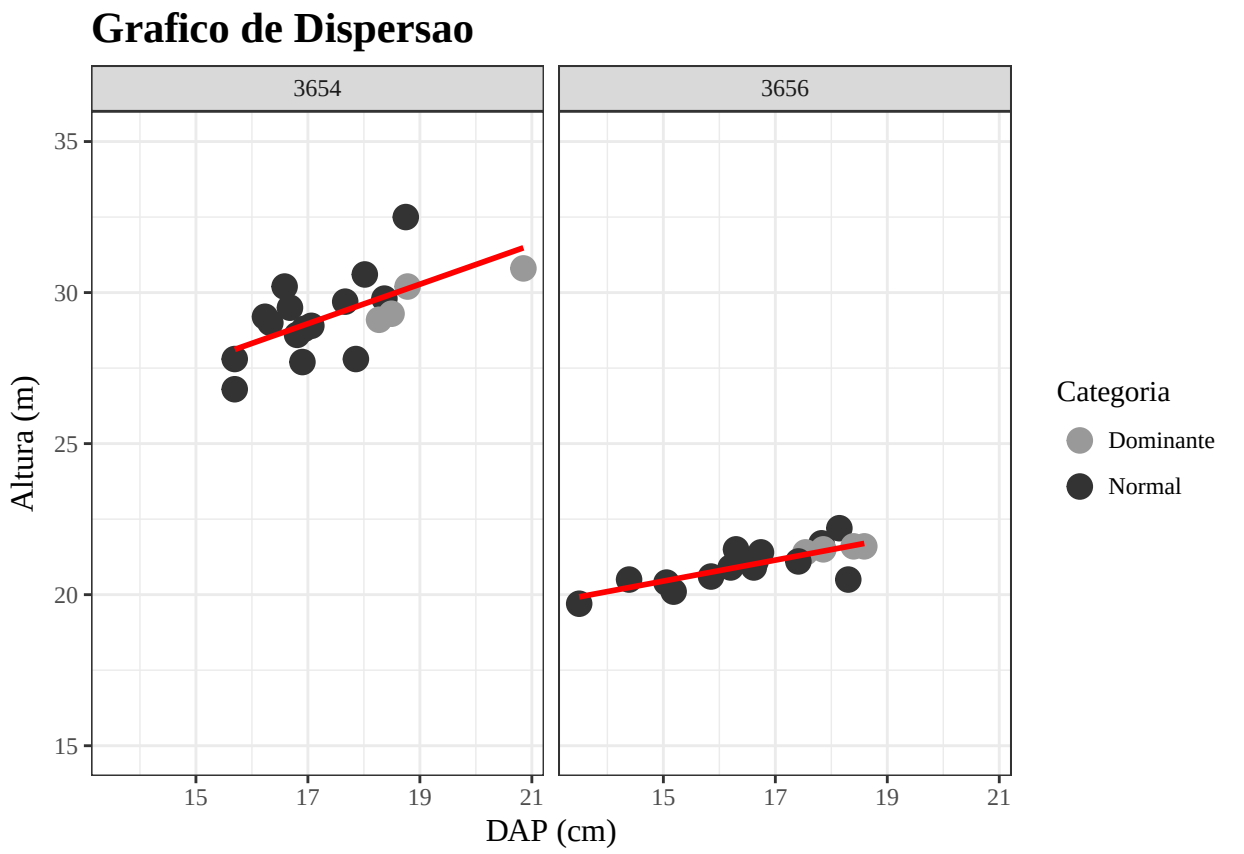
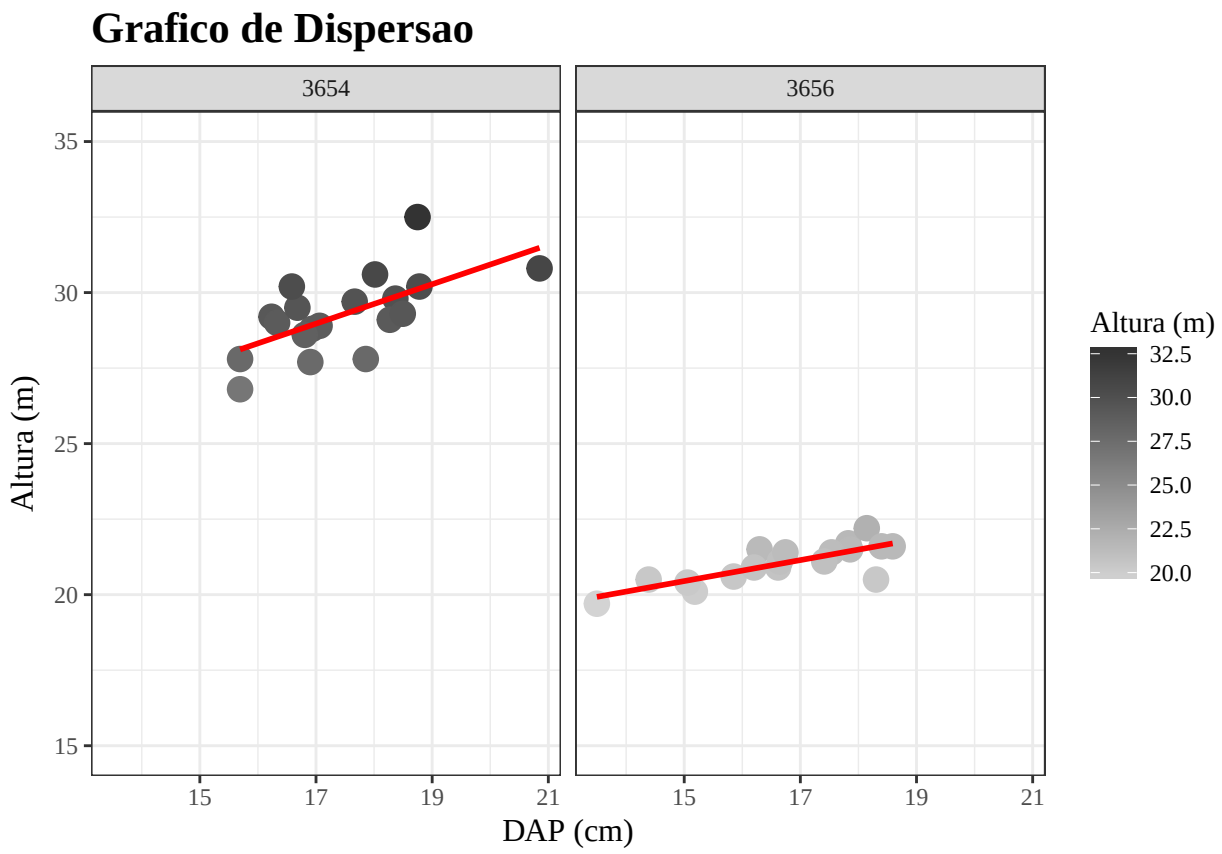


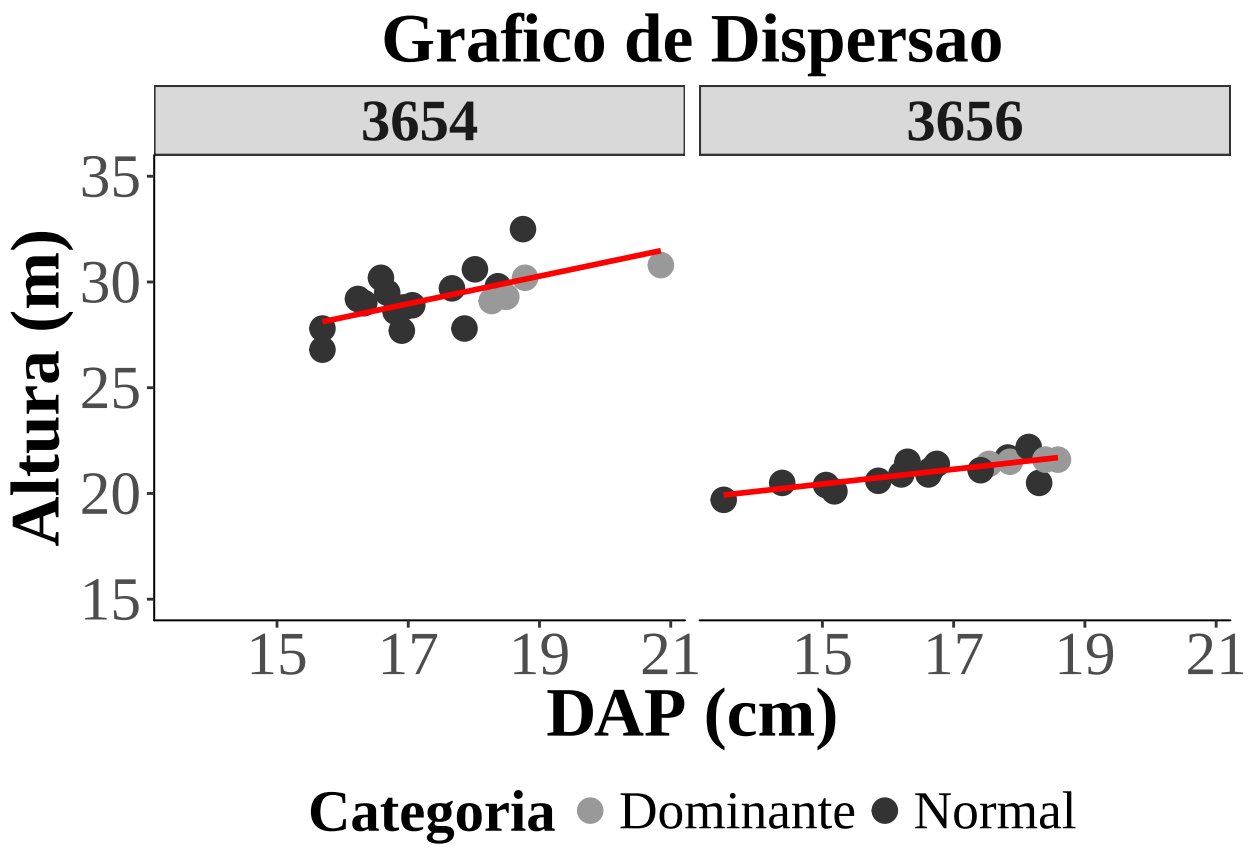
Grafico em escala de cinza para variável contínua

```
a + geom_point(size = 4, aes(color=ALT1)) +
  labs(x="DAP (cm)", y="Altura (m)", colour = "Altura (m)", title = "Grafico de Dispersao") +
  theme_bw(base_family = "serif") +
  theme(
    plot.title=element_text(size = 16, face="bold", vjust = 0.9),
    axis.title=element_text(size = 12)) +
  coord_cartesian(ylim = c(15,35)) +
  geom_smooth(method = "lm", color = "red", se=FALSE) +
  facet_grid(.~CODTALHAO) +
  scale_colour_gradient(low = "light gray",high = "gray20")
```



Mais customizações em theme

```
a + geom_point(size = 4, aes(color=DESCCATEGORIA)) +
  labs(x="DAP (cm)", y="Altura (m)", colour = "Categoria", title = "Grafico de Dispersao") +
  theme_bw(base_family = "serif") +
  theme(
    legend.position = "bottom",
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.border = element_blank(),
    plot.title = element_text(size = 26, face = "bold", hjust = 0.5),
    axis.title = element_text(size = 26, face = "bold"),
    axis.text = element_text(size = 23),
    axis.line.x = element_line(color = "black"),
    axis.line.y = element_line(color = "black"),
    strip.text.x = element_text(size = 22,face = "bold"),
    legend.text = element_text(size = 20),
    legend.title = element_text(size = 22, face="bold") ) +
  coord_cartesian(ylim = c(15,35)) +
  geom_smooth(method = "lm", color = "red", se=FALSE) +
  facet_grid(.~CODTALHAO) +
  scale_colour_grey(start = 0.6, end = 0.2)
```



10.2) Histogramas

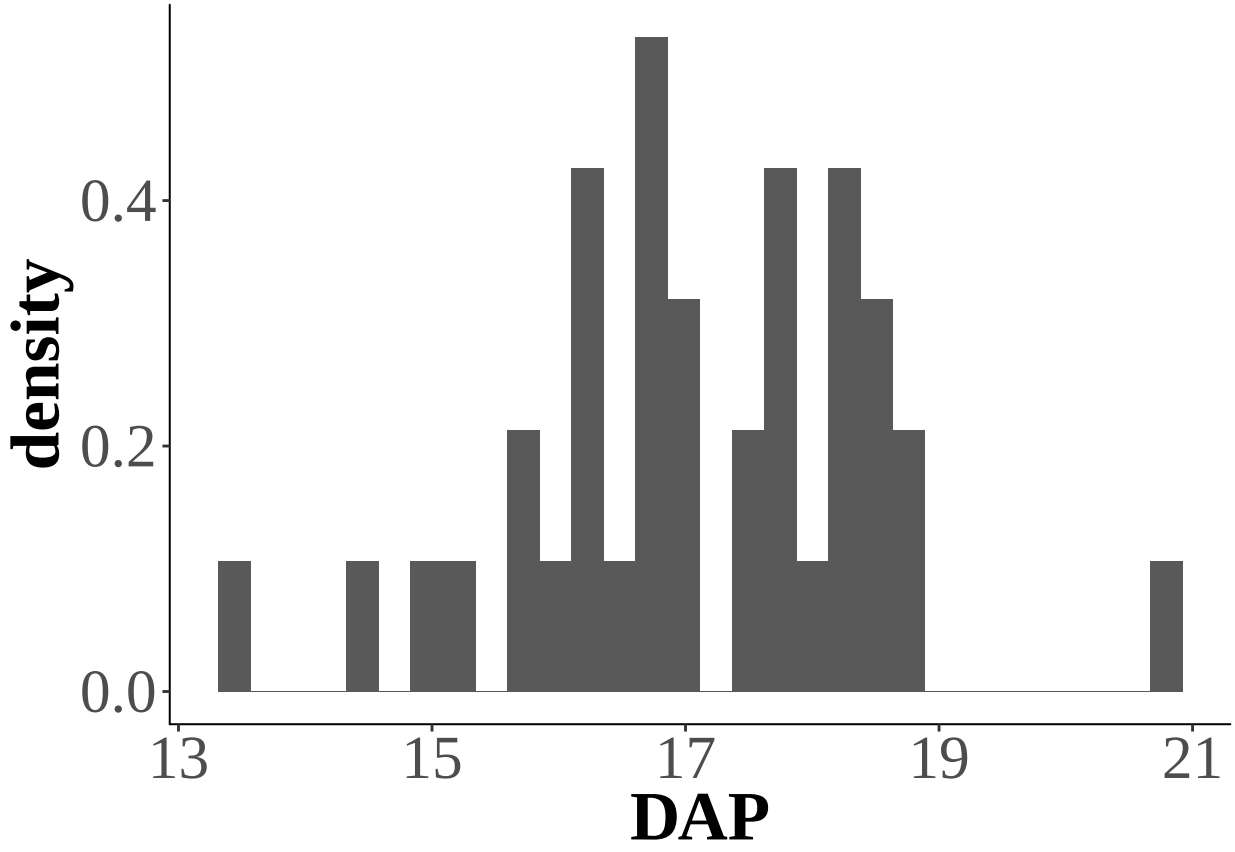
Histograma para o DAP

```
b <- ggplot(dados_g, aes(x = DAP, y=..density..)) +
  theme_bw(base_family = "serif") +
  theme(
    legend.position = "bottom",
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.border = element_blank(),
    plot.title = element_text(size = 26, face = "bold", hjust = 0.5),
    axis.title = element_text(size = 26, face = "bold"),
    axis.text = element_text(size = 23),
    axis.line.x = element_line(color = "black"),
    axis.line.y = element_line(color = "black"),
    strip.text.x = element_text(size = 22,face = "bold"),
    legend.text = element_text(size = 20),
    legend.title = element_text(size = 22, face="bold") ) +
  scale_colour_grey(start = 0.6, end = 0.2) +
  scale_fill_grey(start = 0.6, end = 0.2)
```

para se criar um histograma basta alterar o argumento geom_

```
b + geom_histogram()
```

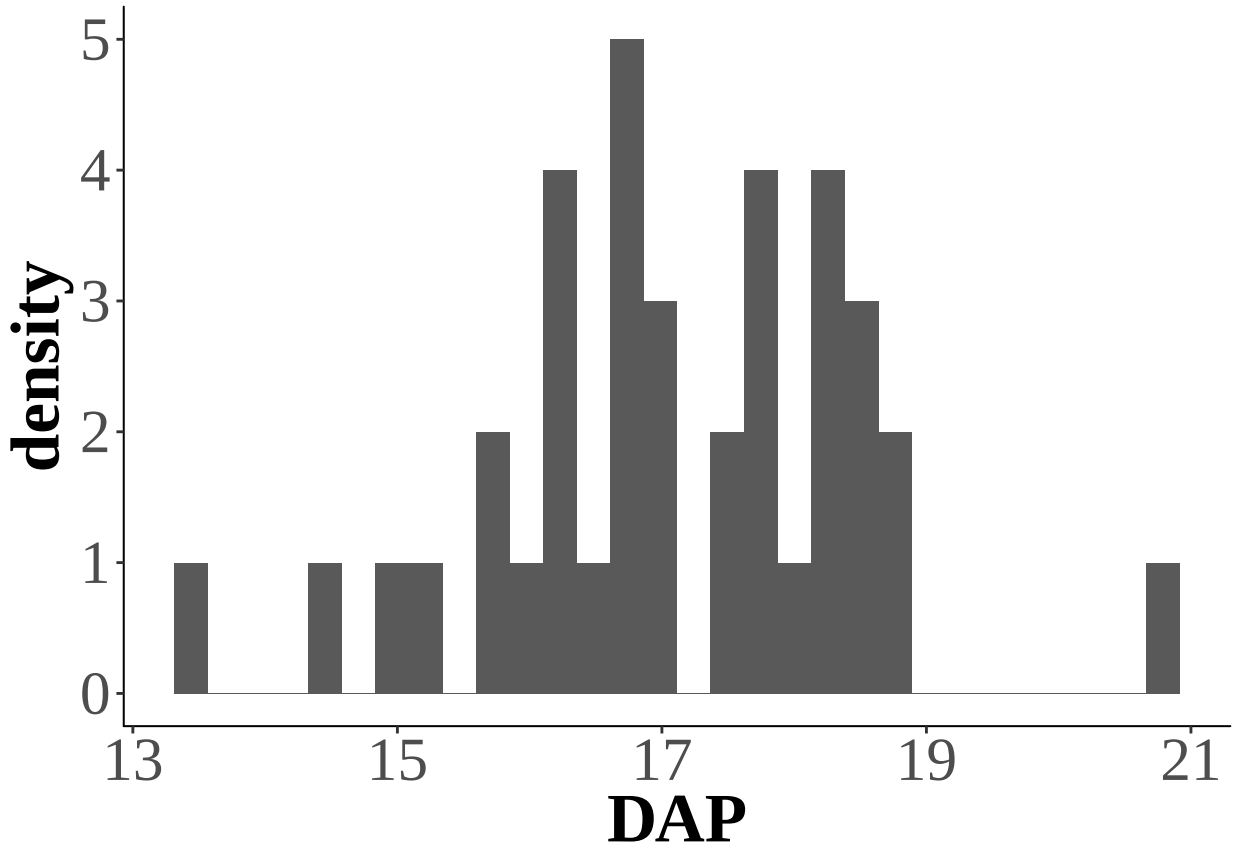
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



frequencia ao inves de densidade

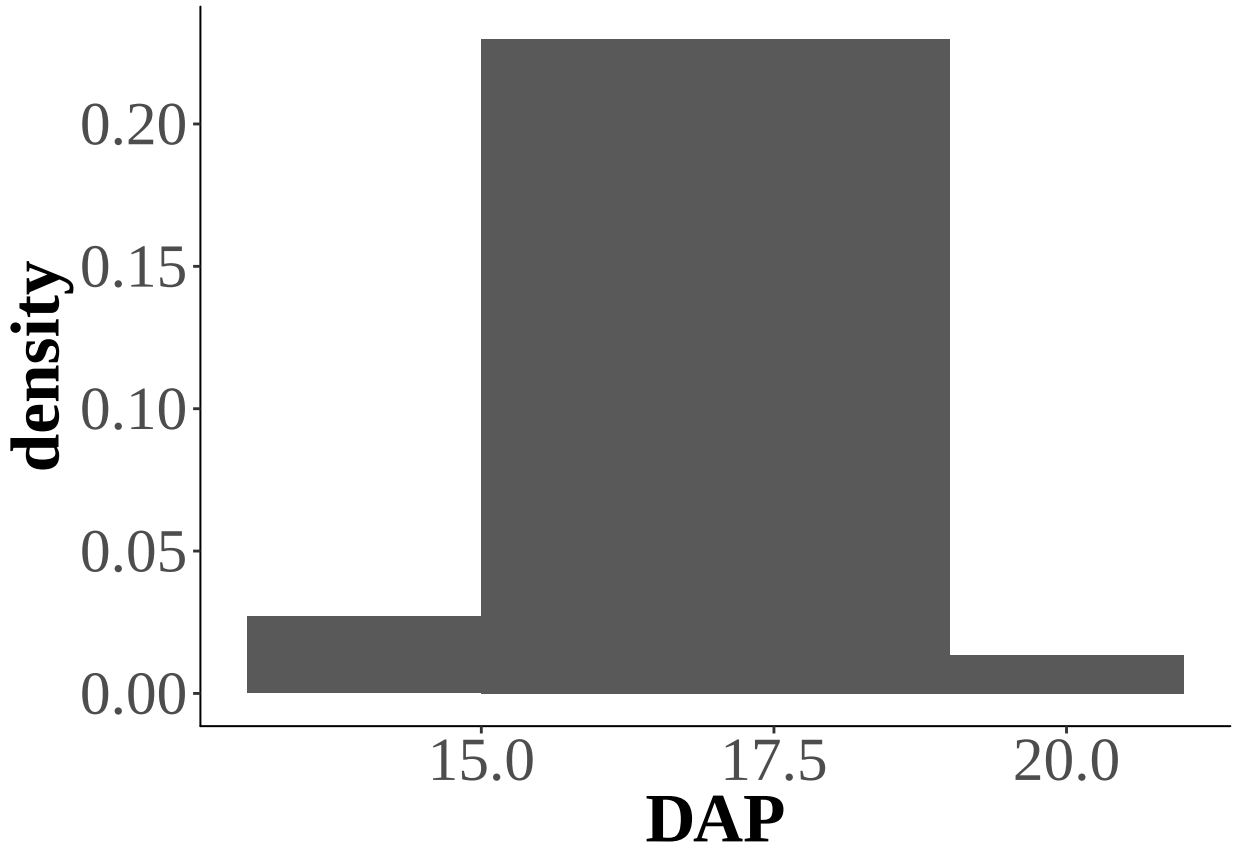
```
b + geom_histogram(aes(y=..count..))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



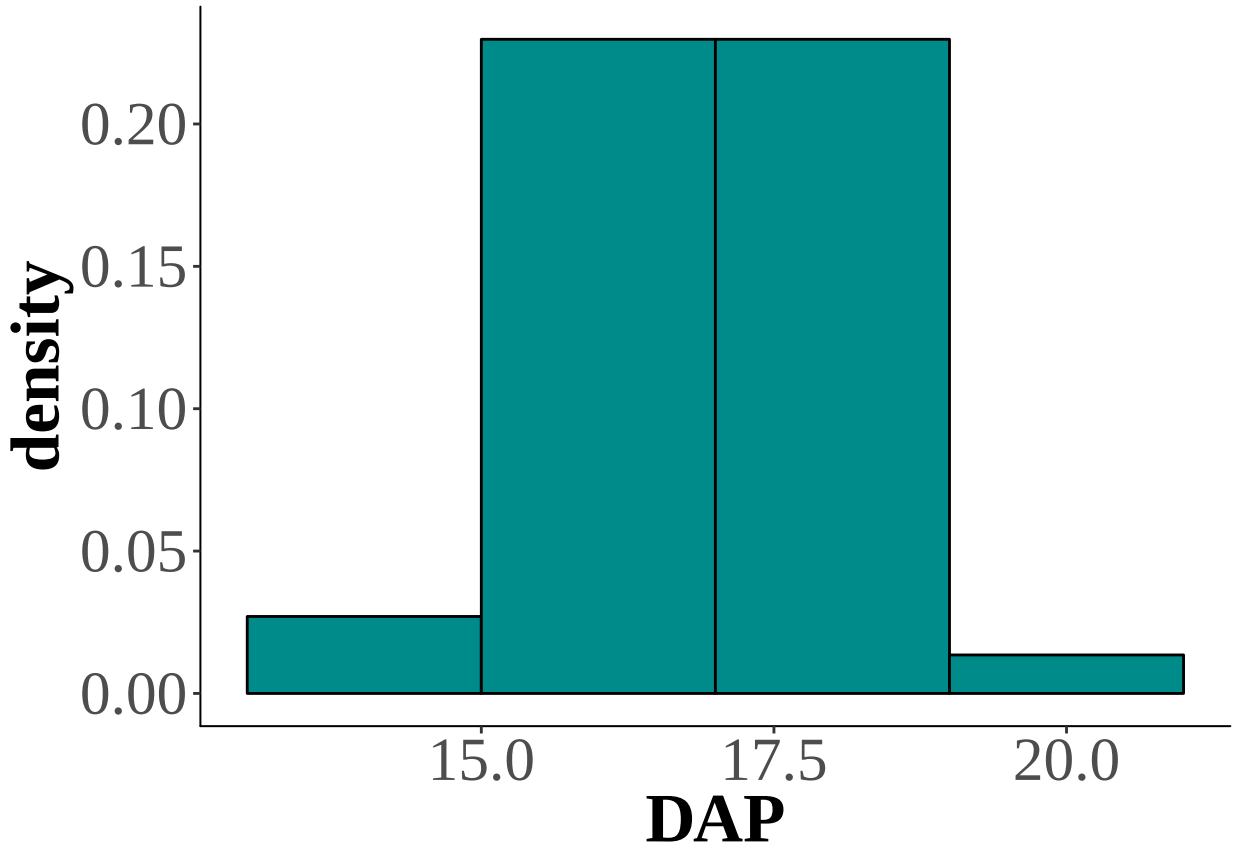
Largura das colunas

```
b + geom_histogram(binwidth = 2)
```



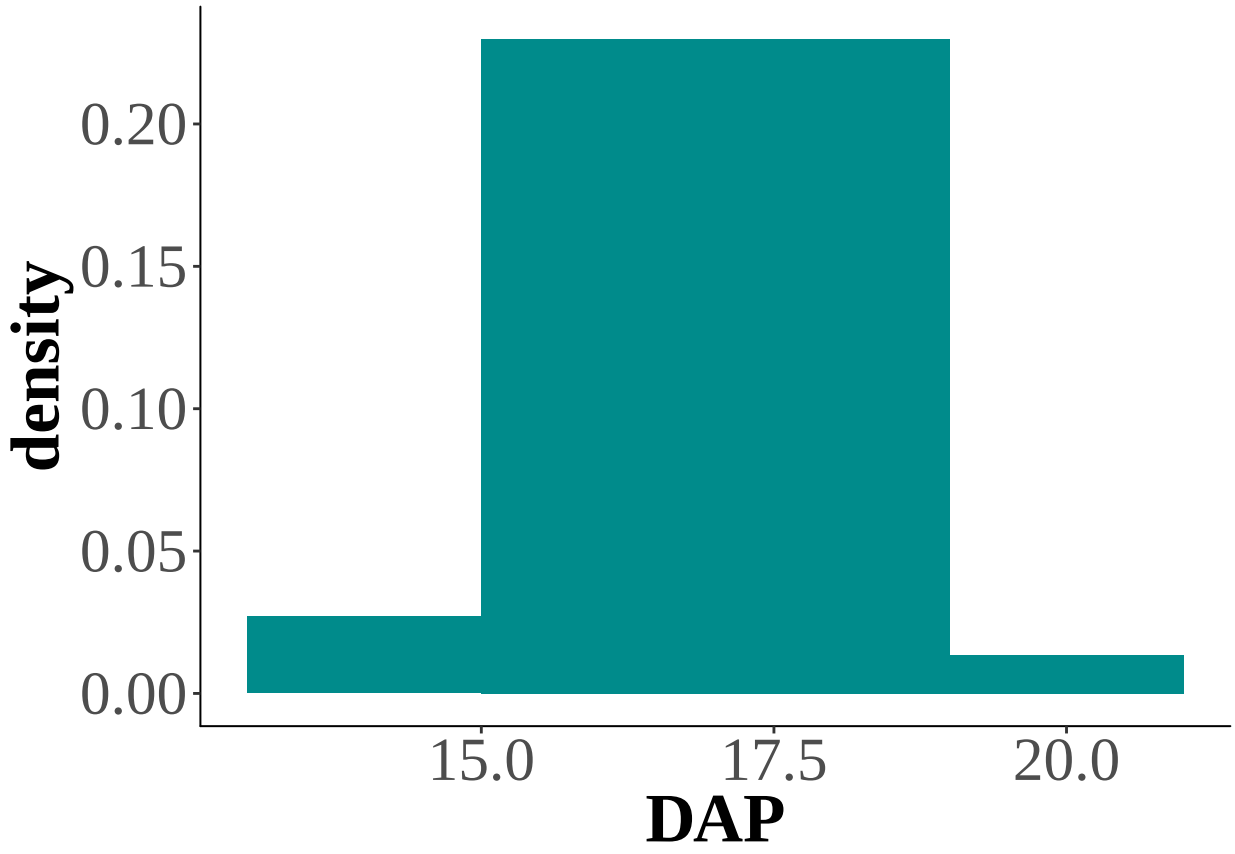
cor personalizada

```
b + geom_histogram(binwidth = 2, fill = "cyan4", color = "black")
```



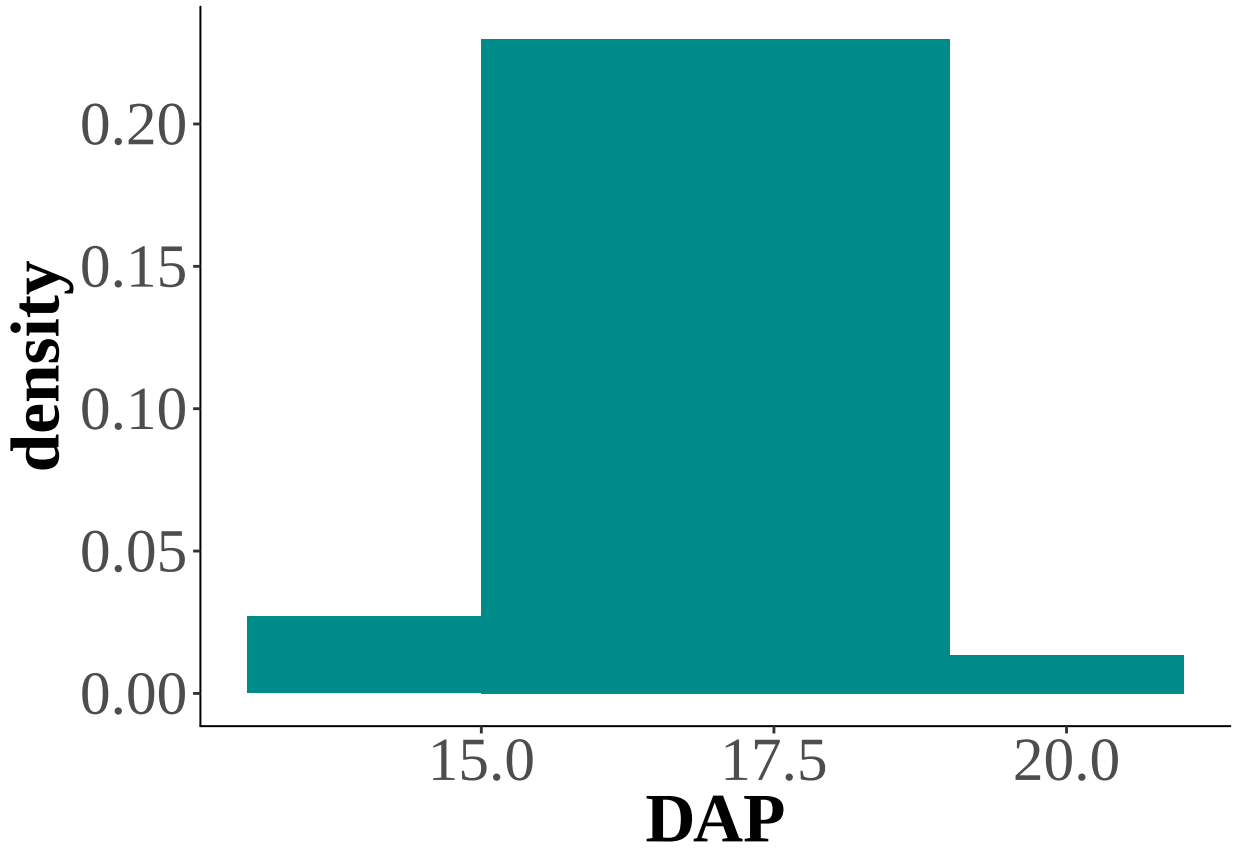
Preenchimento personalizado

```
b + geom_histogram(binwidth = 2, fill = "cyan4")
```



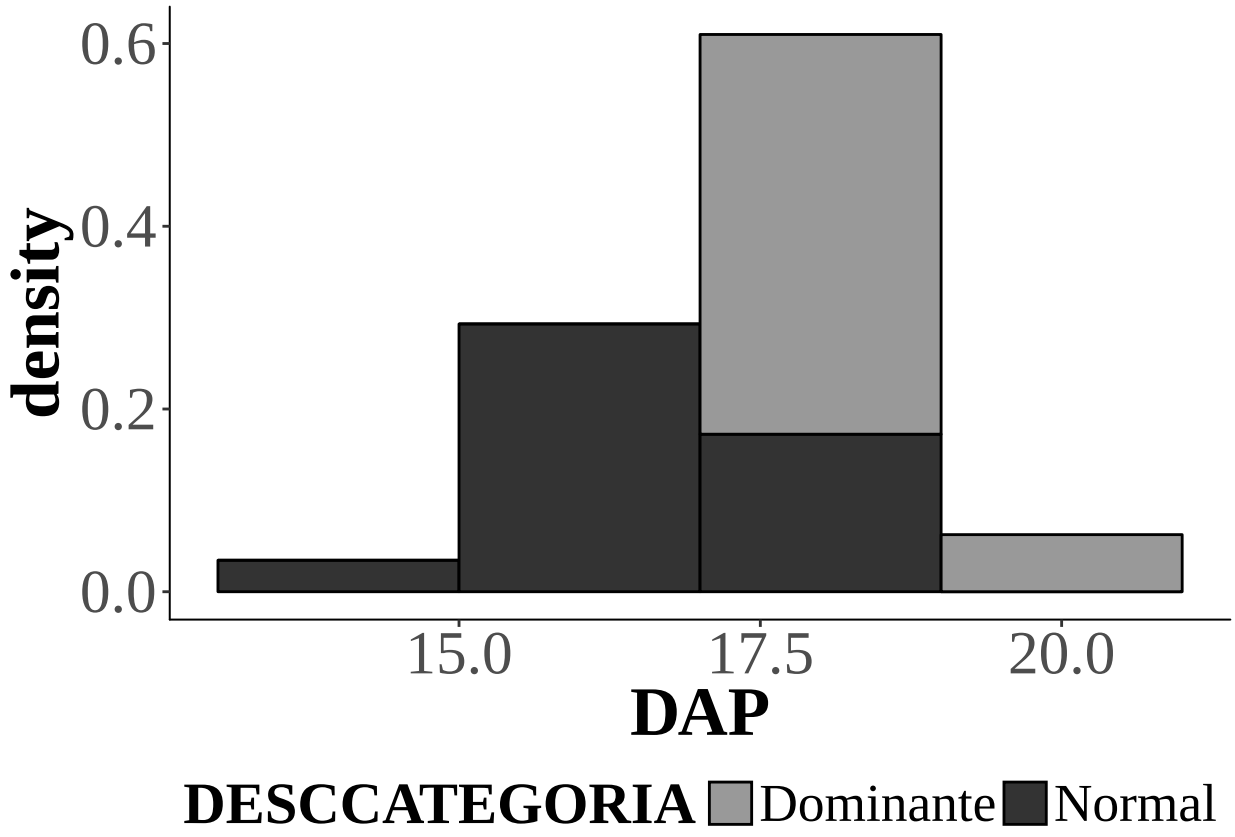
Largura das colunas

```
b + geom_histogram(binwidth = 2, fill = "cyan4")
```



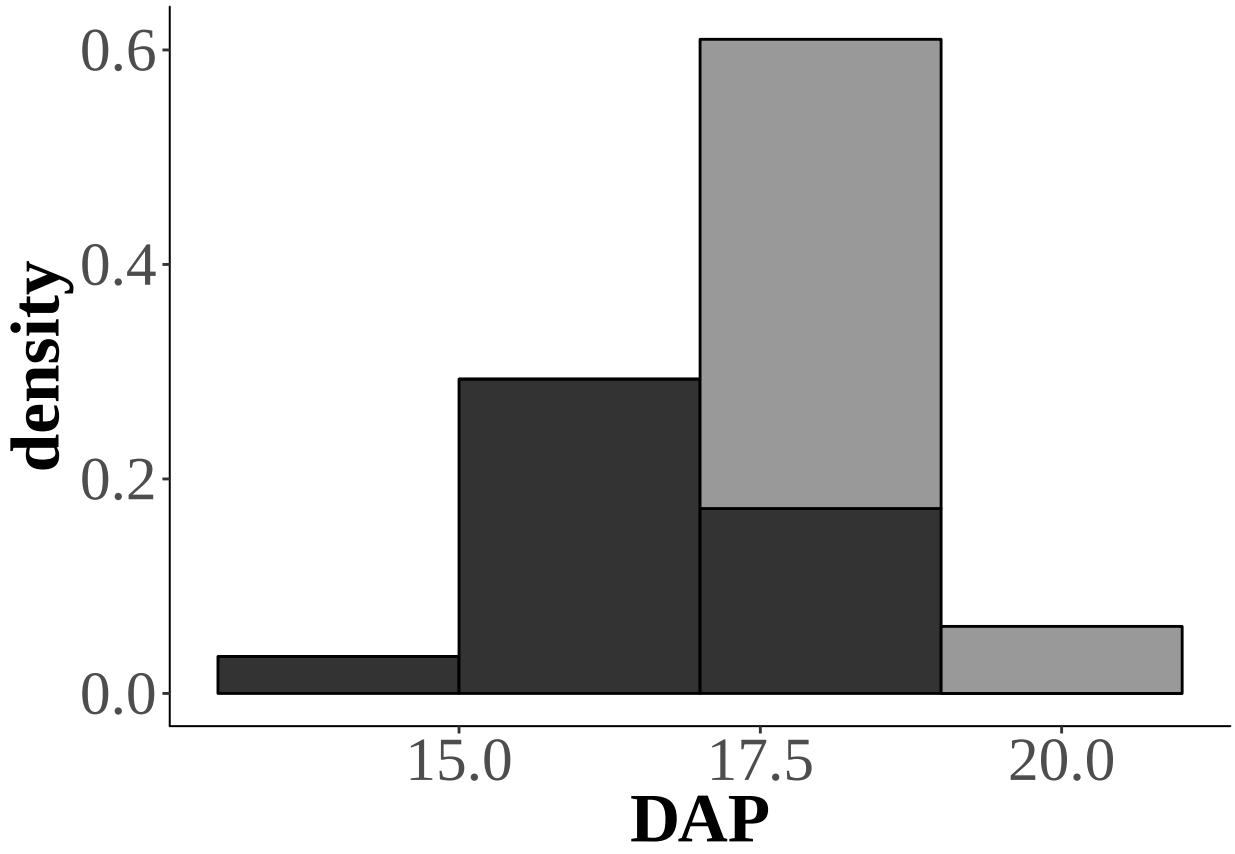
pode-se preencher os dados com uma variavel classificatoria

```
b + geom_histogram(binwidth = 2, aes(fill = DESCCATEGORIA), color = "black")
```



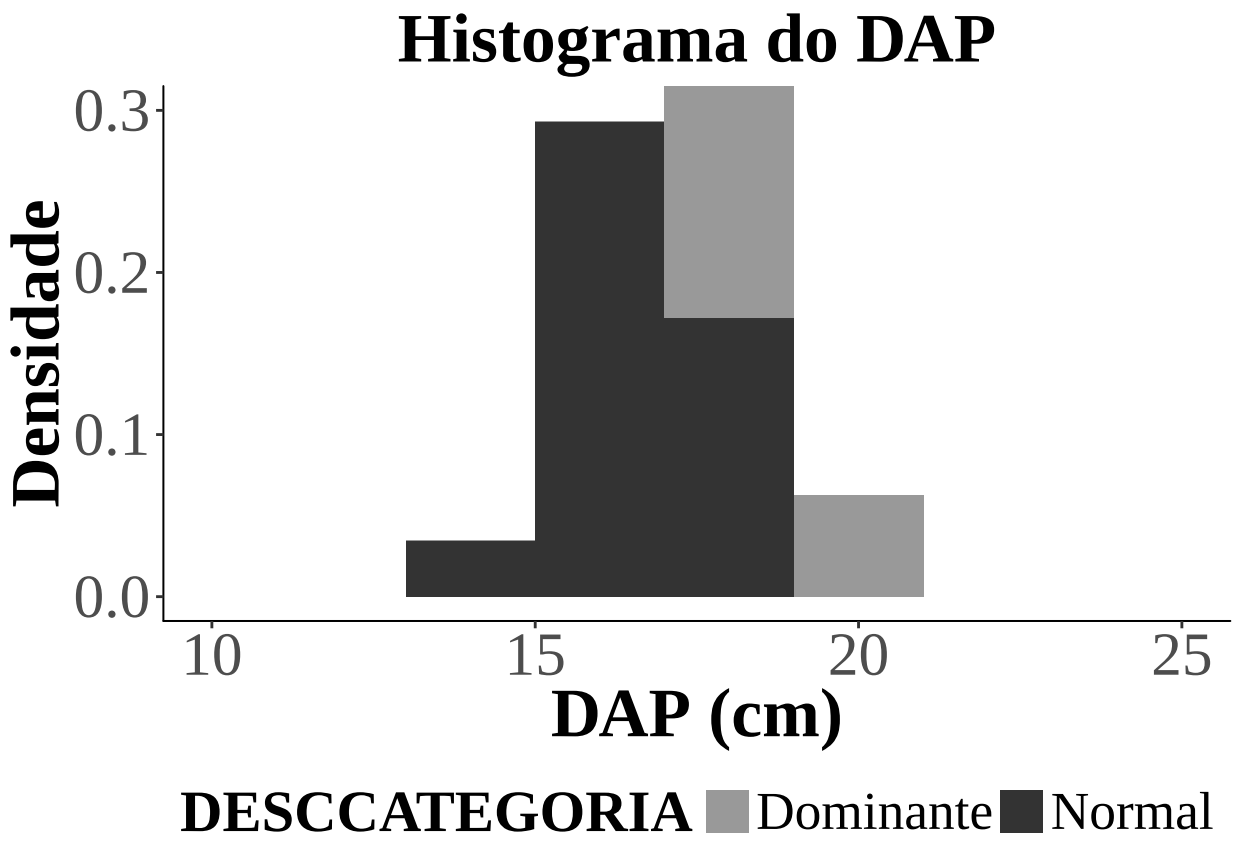
sem legenda

```
b + geom_histogram(binwidth = 2, aes(fill = DESCCATEGORIA), color = "black", show.legend = FALSE)
```



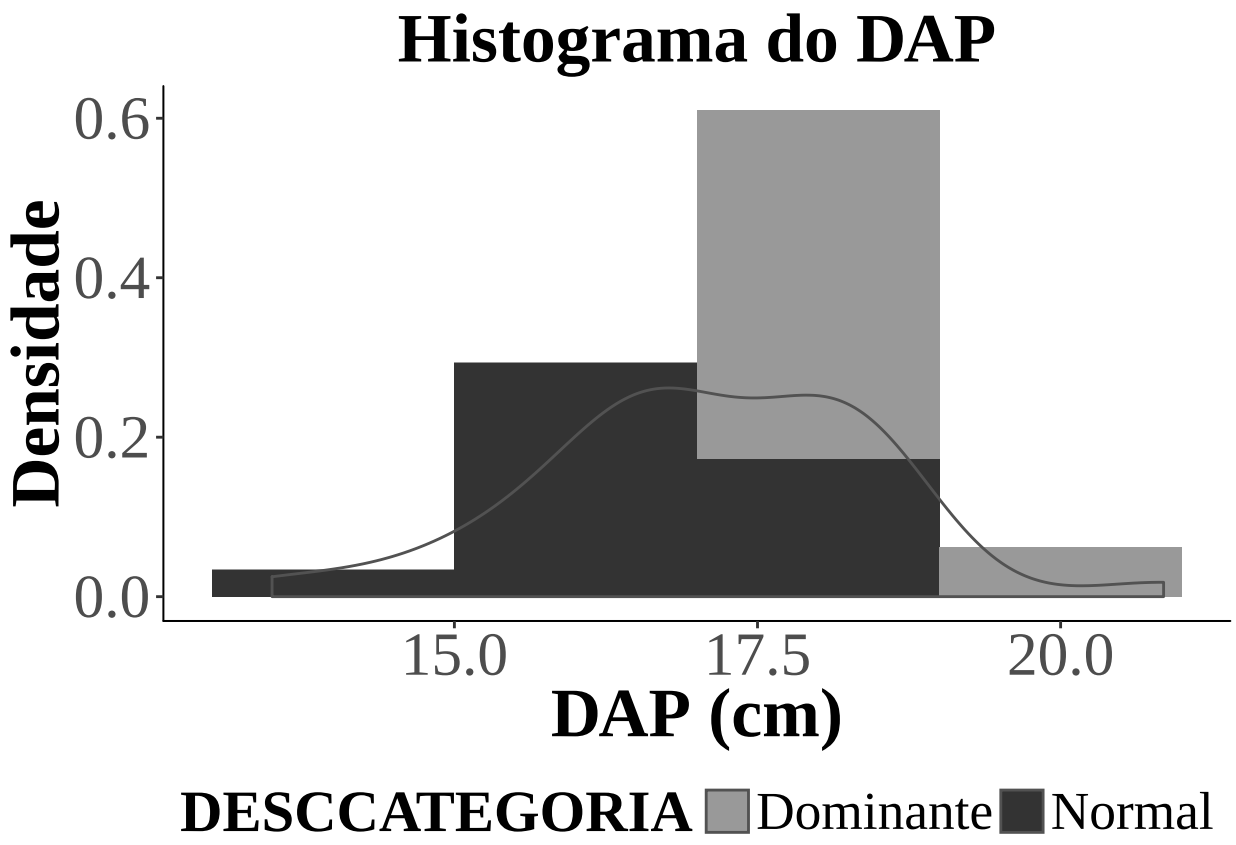
TItulo

```
b + geom_histogram(binwidth = 2, aes(fill = DESCCATEGORIA)) +
  labs(x = "DAP (cm)", y = "Densidade", title = "Histograma do DAP") +
  coord_cartesian(xlim = c(10,25), ylim = c(0, .3))
```

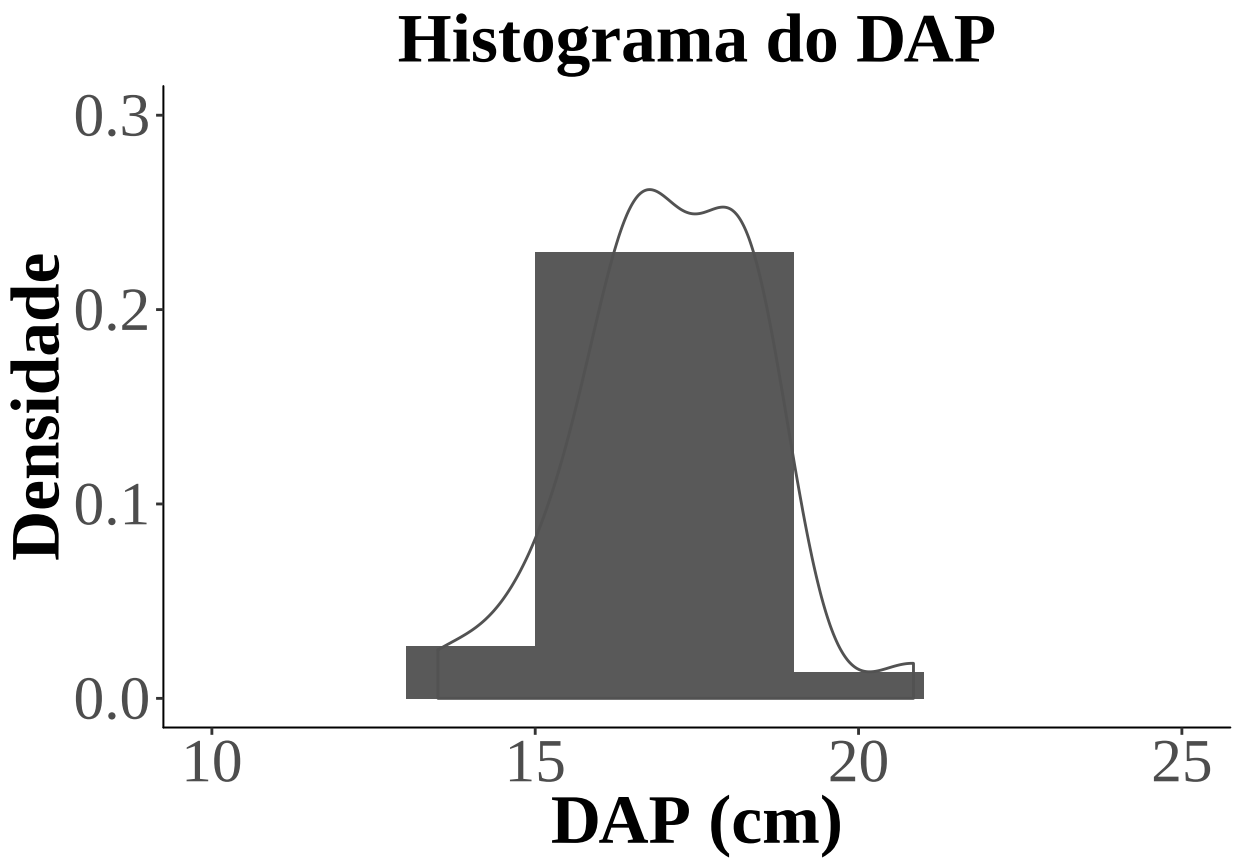


Curva de densidade

```
b + geom_histogram(binwidth = 2, aes(fill = DESCCATEGORIA)) +
  labs(x = "DAP (cm)", y = "Densidade", title = "Histograma do DAP") +
  geom_density(color = "grey32")
```

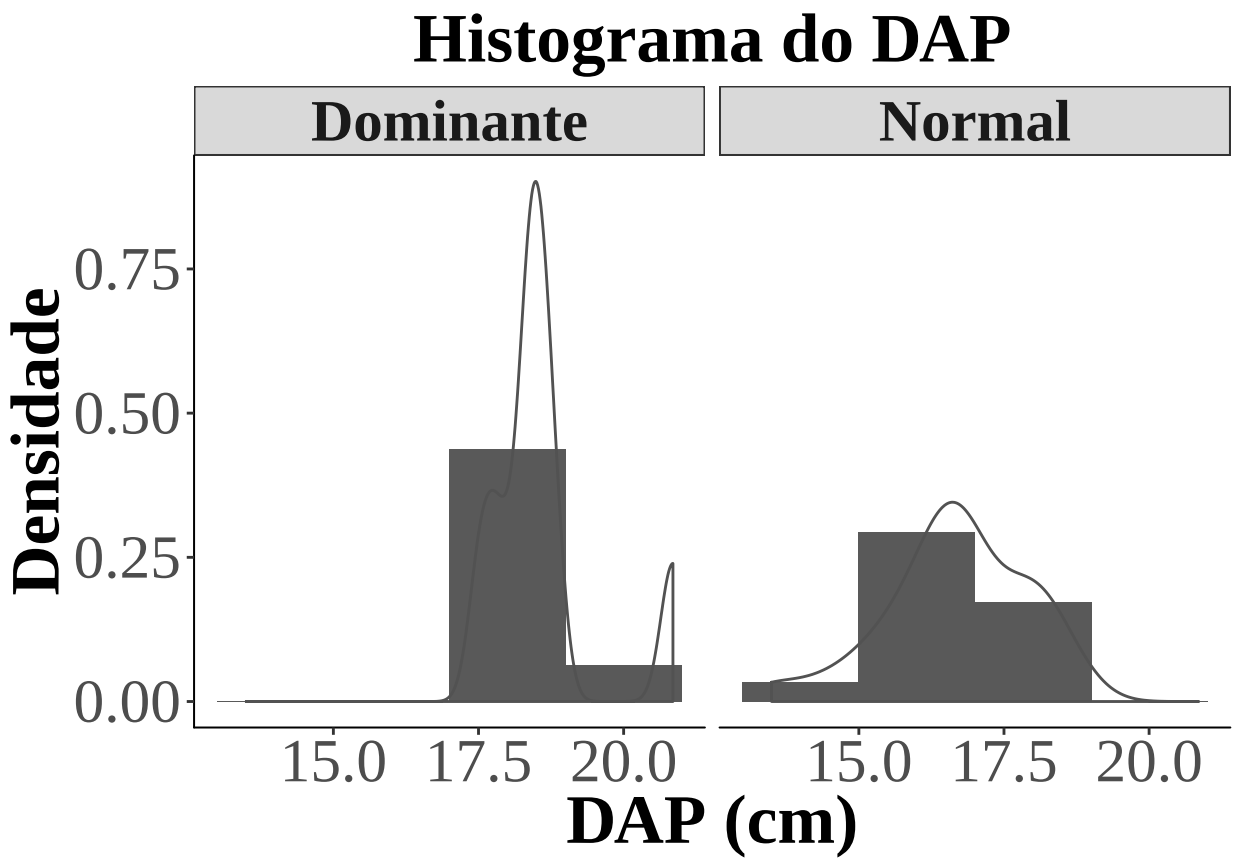



```
b + geom_histogram(binwidth = 2) +
  labs(x = "DAP (cm)", y = "Densidade", title = "Histograma do DAP") +
  coord_cartesian(xlim = c(10,25), ylim = c(0, .3)) +
  geom_density(color = "grey32")
```

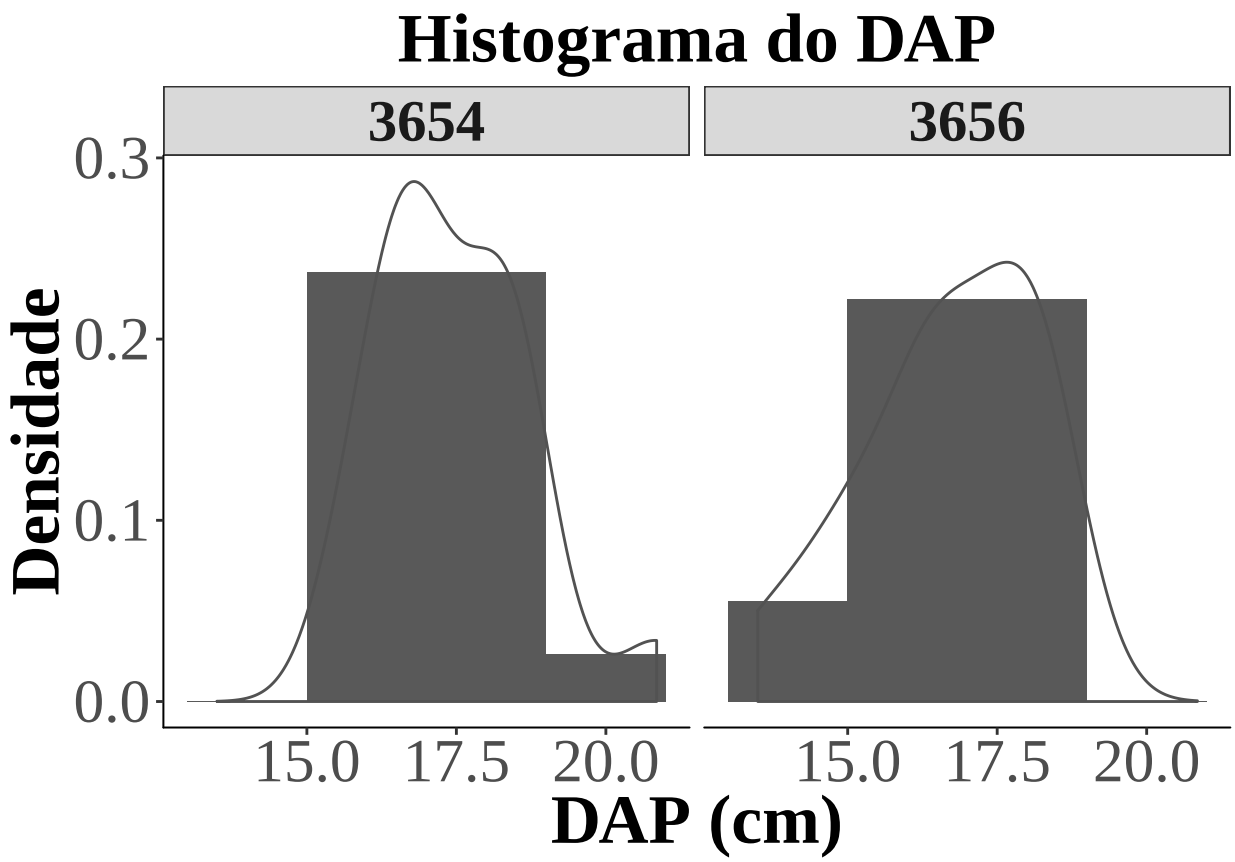


Podemos utilizar subsetting com uma variavel categorica

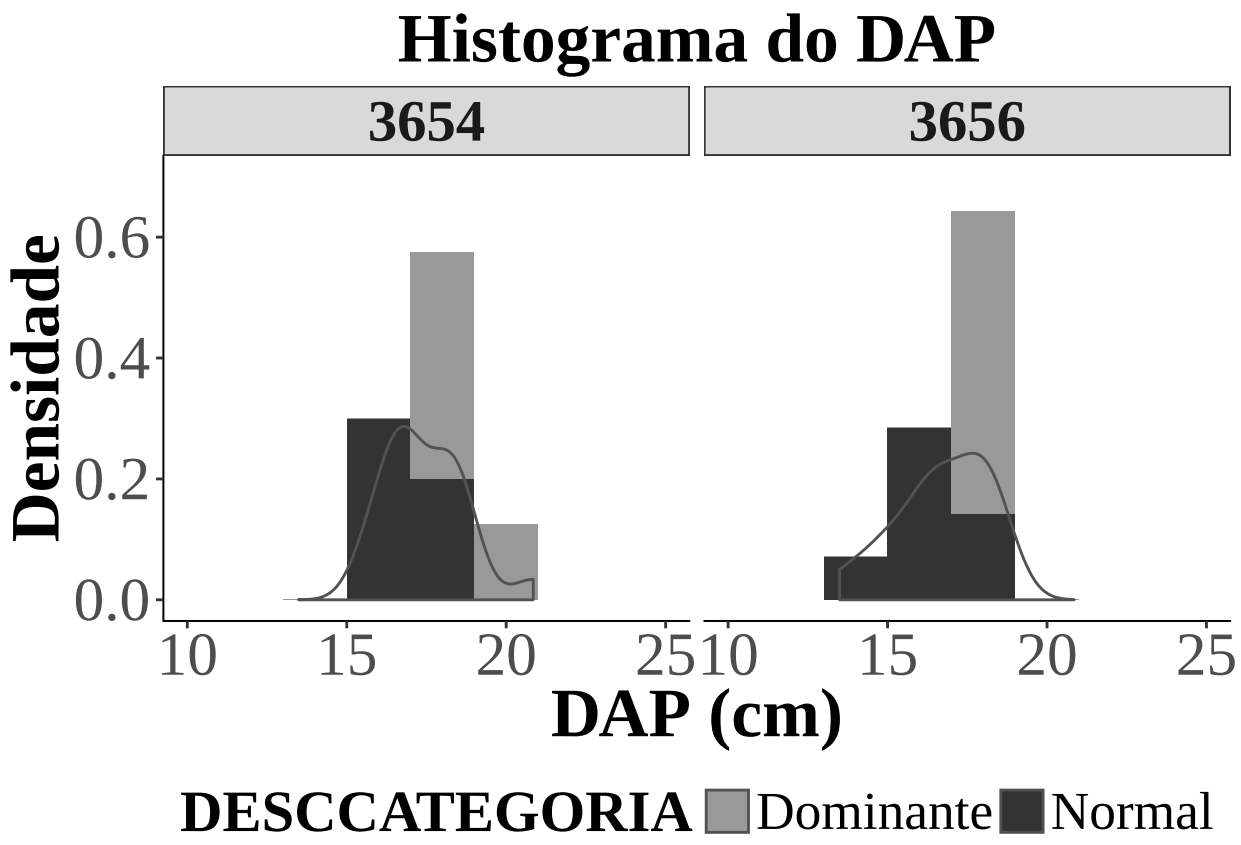
```
b + geom_histogram(binwidth = 2) +
  labs(x = "DAP (cm)", y = "Densidade", title = "Histograma do DAP") +
  geom_density(color = "grey32") +
  facet_grid(.~DESCCATEGORIA)
```



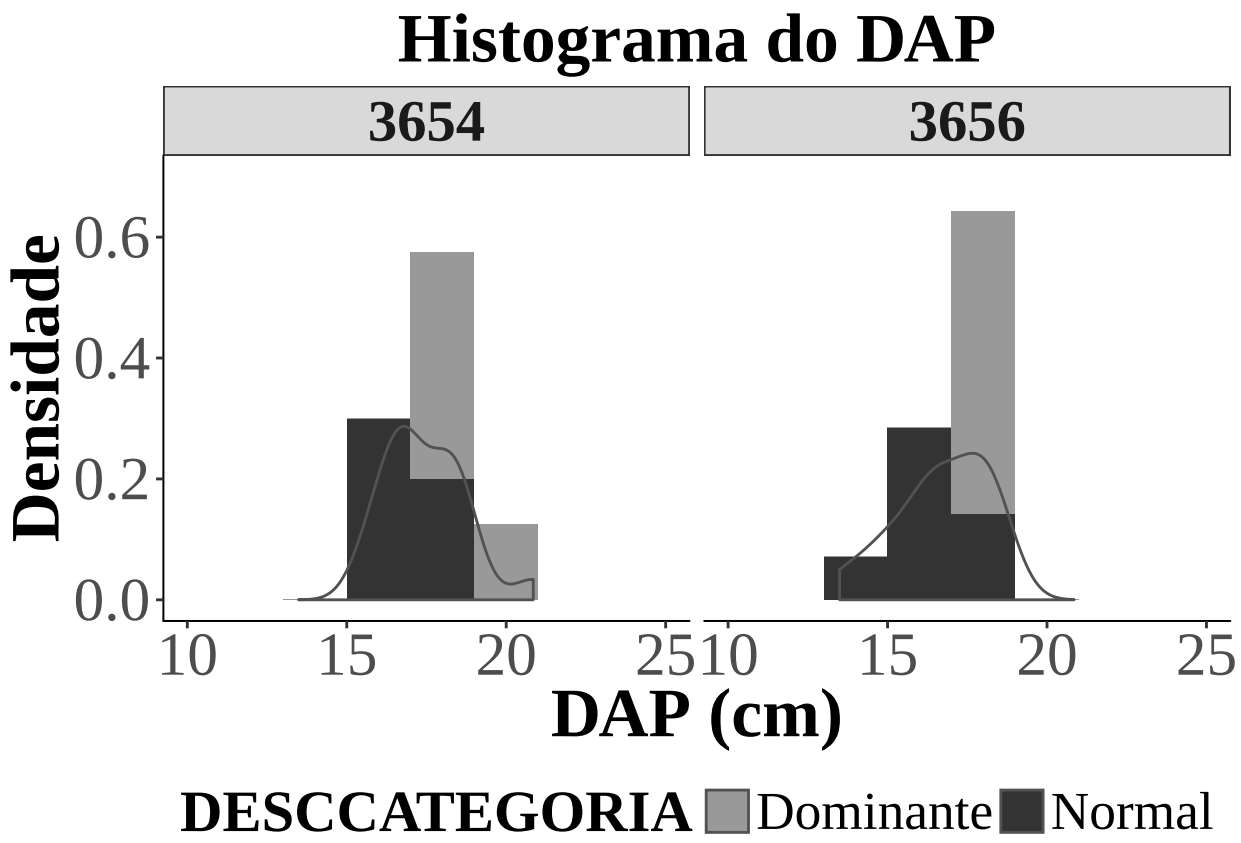

```
b + geom_histogram(binwidth = 2) +
  labs(x = "DAP (cm)", y = "Densidade", title = "Histograma do DAP") +
  geom_density(color = "grey32") +
  facet_grid(.~CODTALHAO)
```




```
b + geom_histogram(binwidth = 2, aes(fill = DESCCATEGORIA)) +
  labs(x = "DAP (cm)", y = "Densidade", title = "Histograma do DAP") +
  coord_cartesian(xlim = c(10,25), ylim = c(0, .7)) +
  geom_density(color = "grey32") +
  facet_grid(.~CODTALHAO)
```




```
b + geom_histogram(binwidth = 2, aes(fill = DESCCATEGORIA)) +
  labs(x = "DAP (cm)", y = "Densidade", title = "Histograma do DAP") +
  coord_cartesian(xlim = c(10,25), ylim = c(0, .7)) +
  geom_density(color = "grey32") +
  facet_grid(.~CODTALHAO)
```



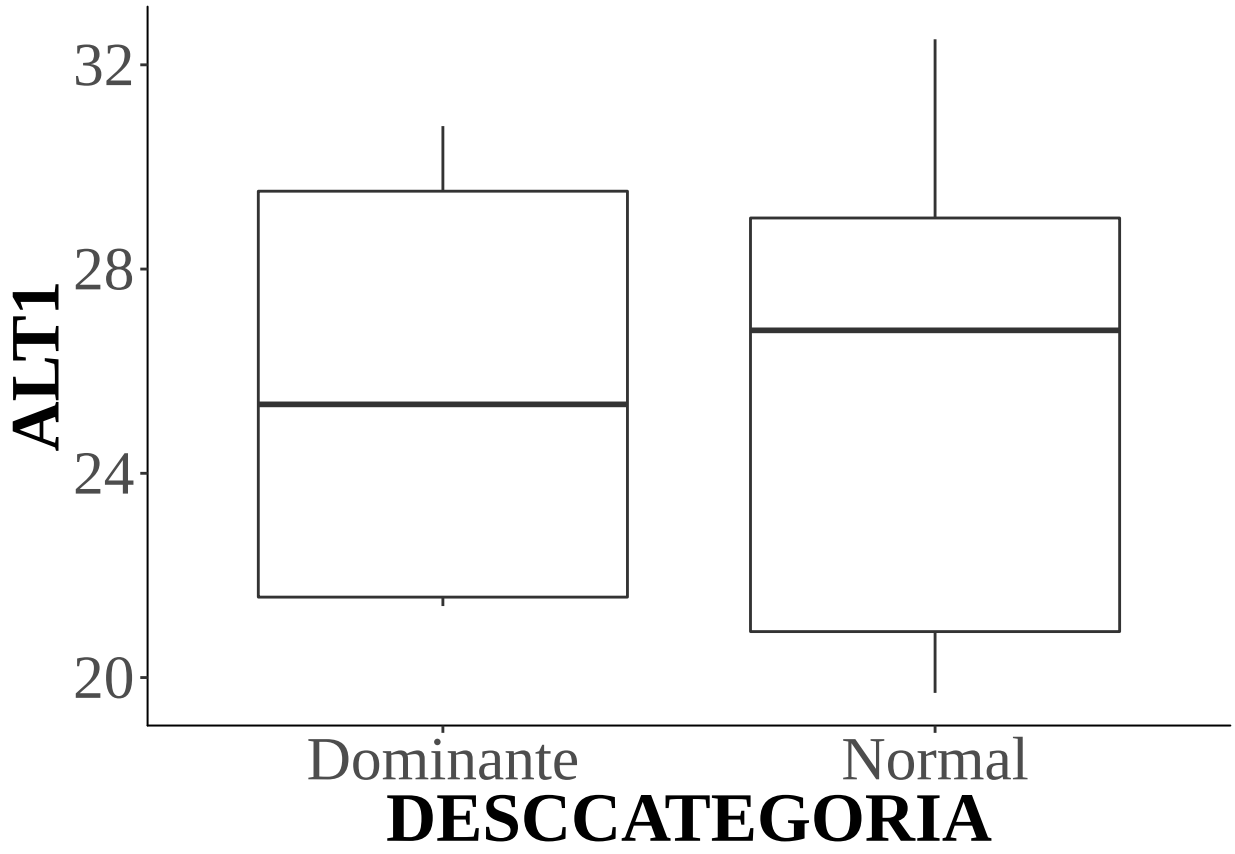
10.3) Boxplot

Boxplot da altura em função da qualidade da árvore

```
c <- ggplot(dados_g, aes(x = DESCCATEGORIA, y = ALT1)) +
  theme_bw(base_family = "serif") +
  theme(
    legend.position = "bottom",
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.border = element_blank(),
    plot.title = element_text(size = 26, face = "bold", hjust = 0.5),
    axis.title = element_text(size = 26, face = "bold"),
    axis.text = element_text(size = 23),
    axis.line.x = element_line(color = "black"),
    axis.line.y = element_line(color = "black"),
    strip.text.x = element_text(size = 22,face = "bold"),
    legend.text = element_text(size = 20),
    legend.title = element_text(size = 22, face="bold") ) +
  scale_colour_grey(start = 0.6, end = 0.2) +
  scale_fill_grey(start = 0.6, end = 0.2)
```

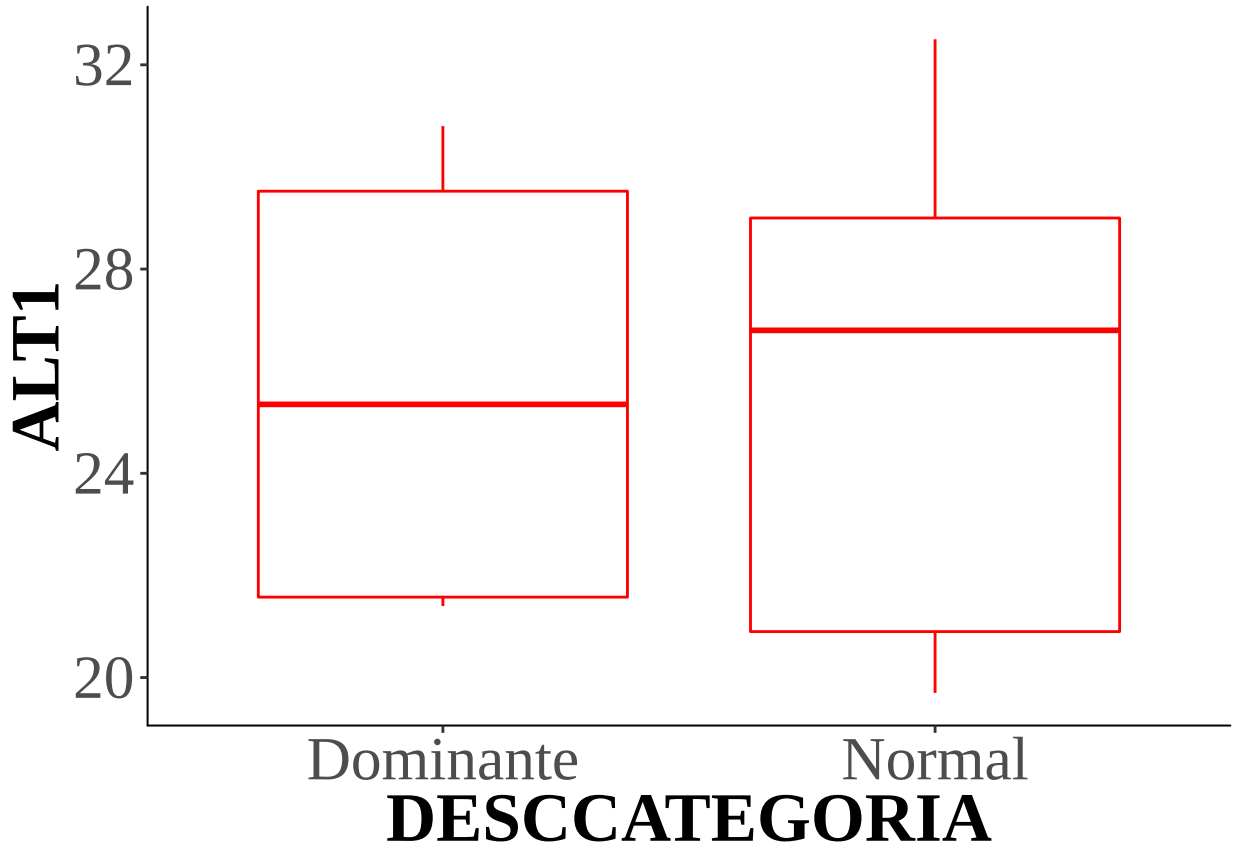
Boxplot basico

```
c + geom_boxplot()
```



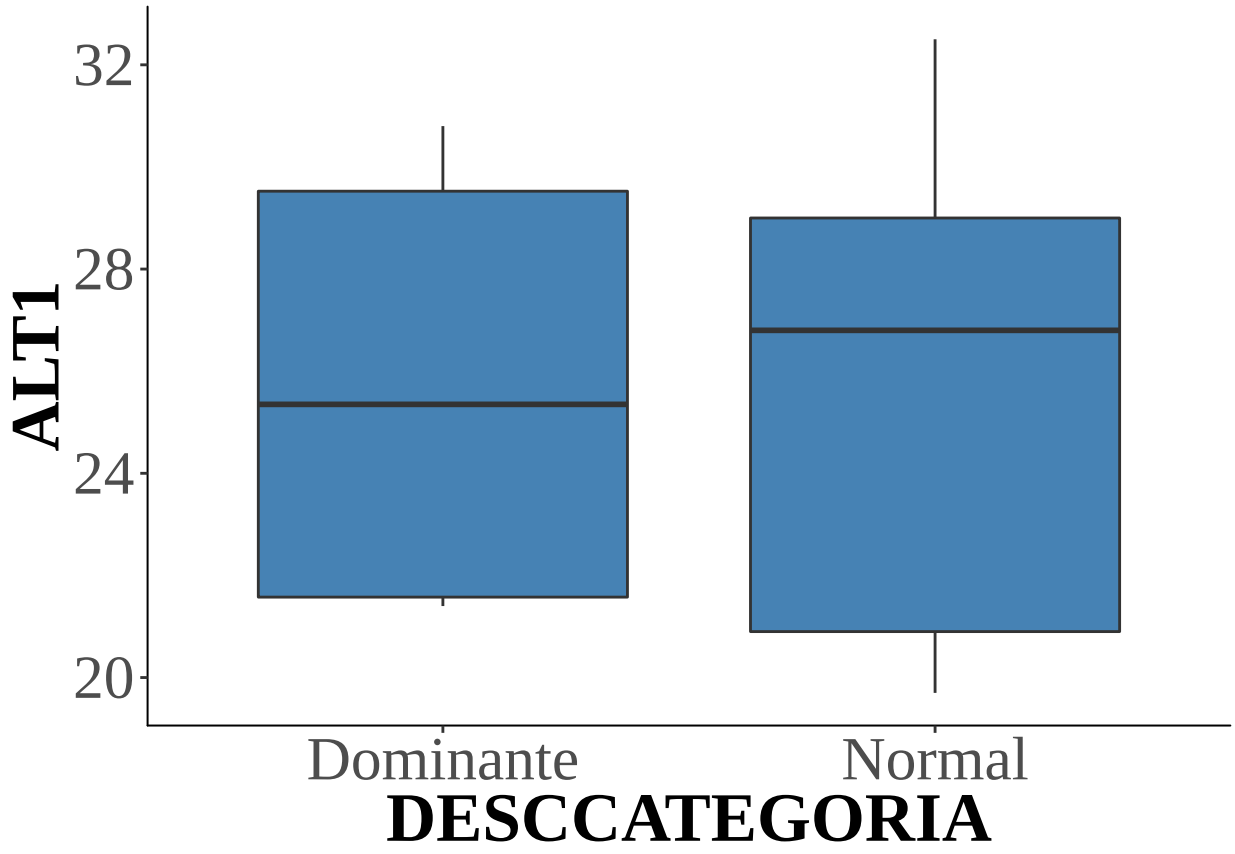
cor personalizada

```
c + geom_boxplot(color = "red")
```



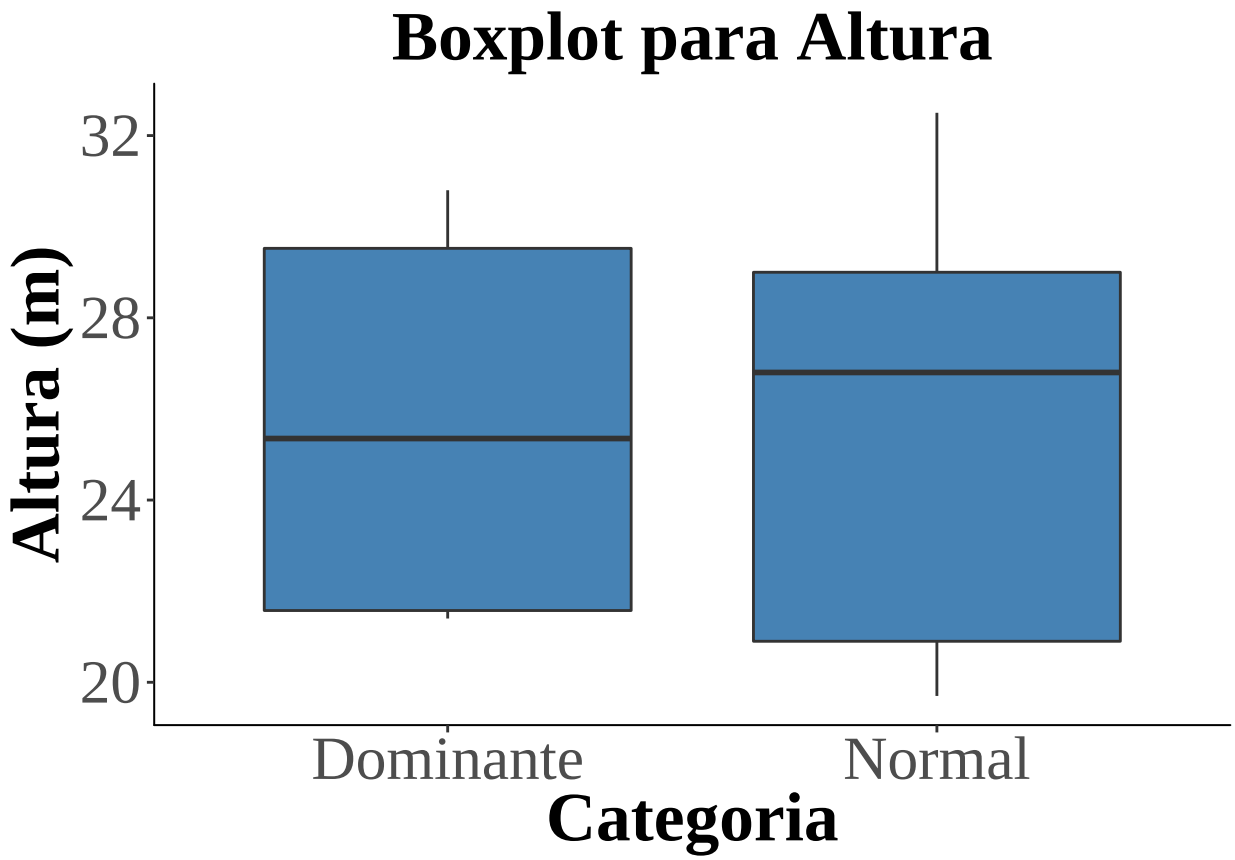
Preenchimento personalizado

```
c + geom_boxplot(fill = "steelblue")
```



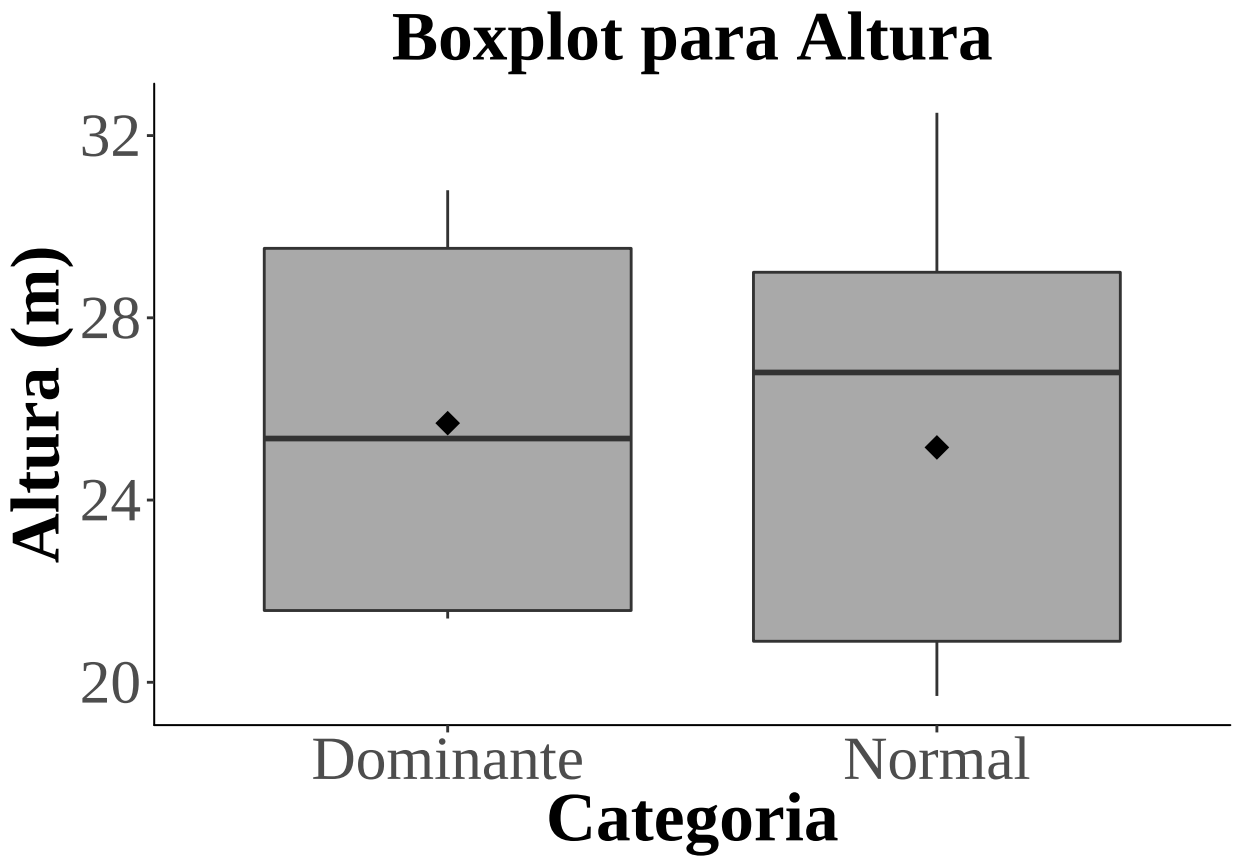
TItulo

```
c + geom_boxplot(fill = "steelblue") +
  labs(x = "Categoria", y = "Altura (m)", title = "Boxplot para Altura")
```



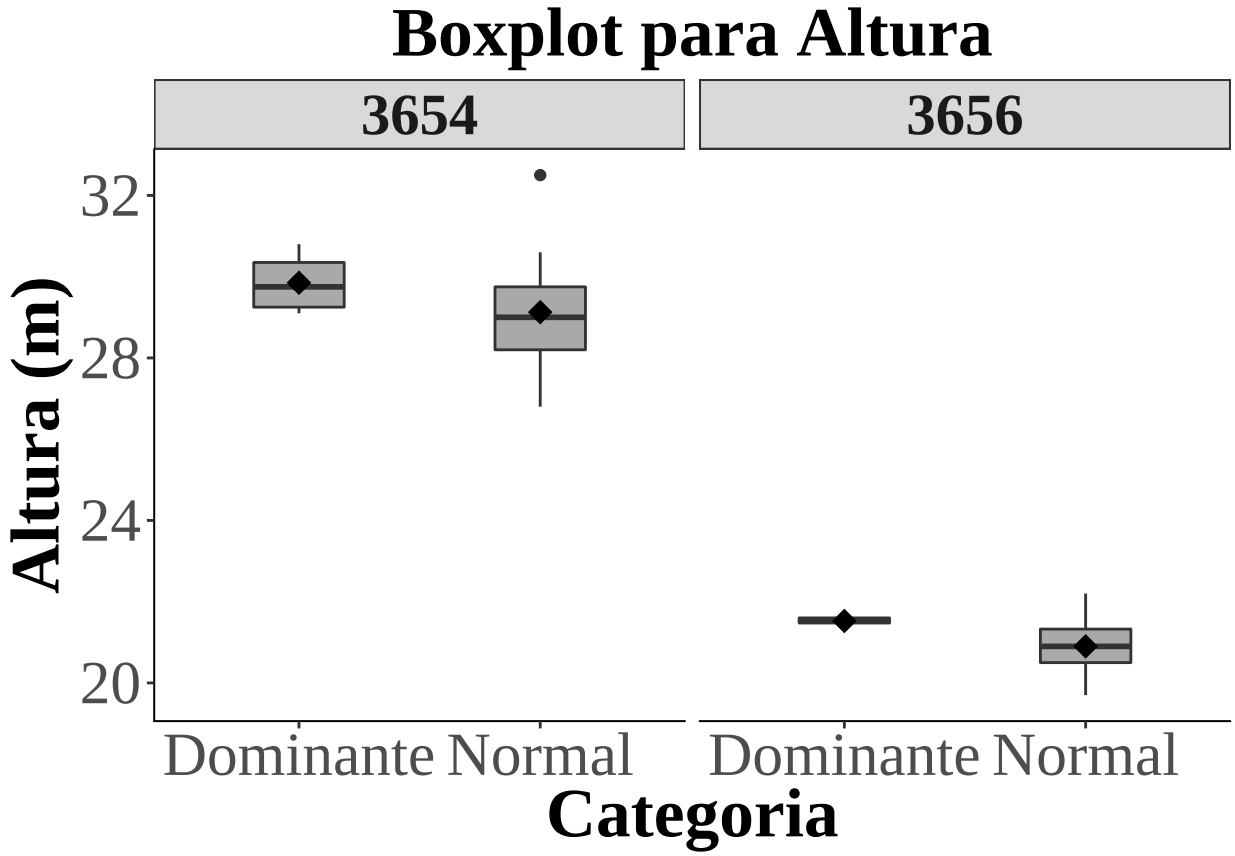
Adicionar ponto de media

```
c + geom_boxplot(fill = "darkgray") +
  labs(x = "Categoria", y = "Altura (m)", title = "Boxplot para Altura") +
  stat_summary(fun.y=mean, geom="point", shape=18, size = 4, show.legend=FALSE)
```



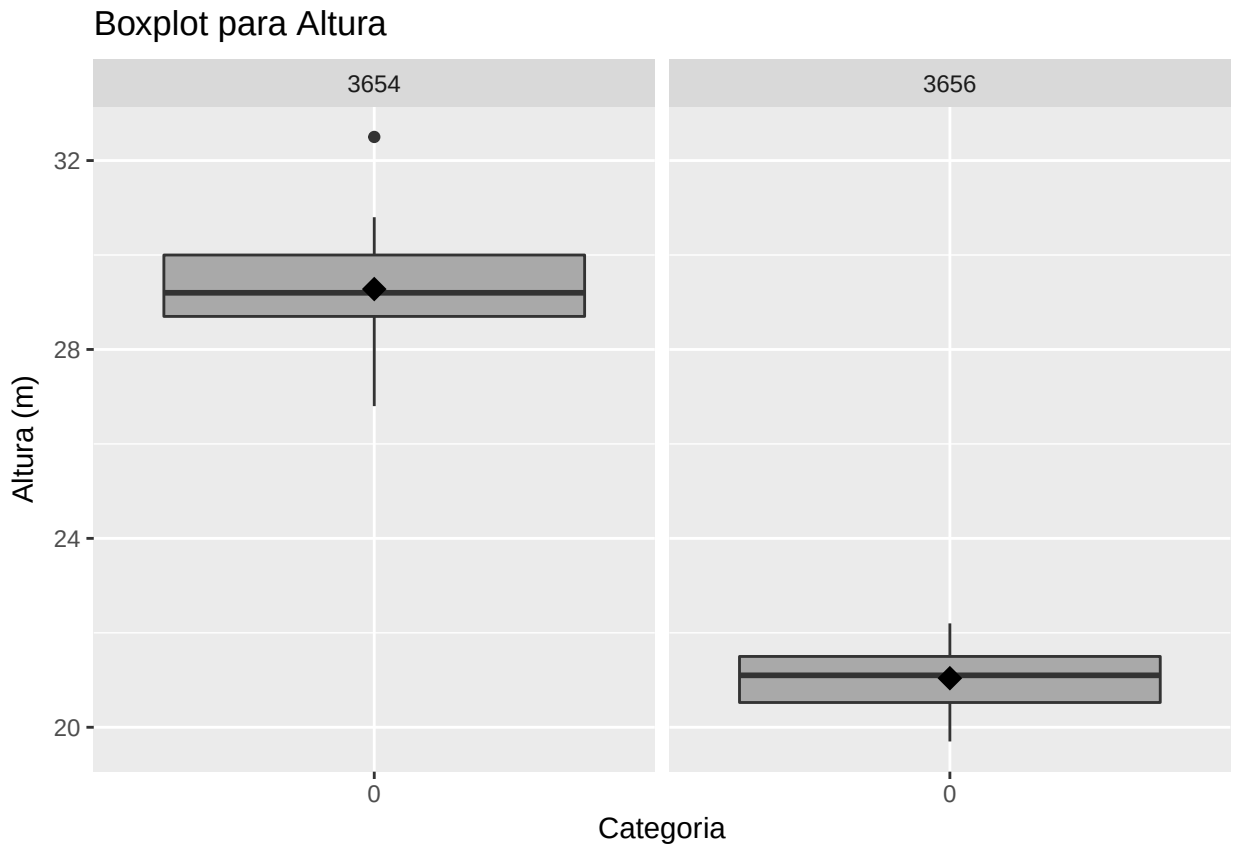
Podemos utilizar subsetting com uma variavel categorica (factor)

```
c + geom_boxplot(fill = "darkgray") +
  labs(x = "Categoria", y = "Altura (m)", title = "Boxplot para Altura") +
  stat_summary(fun.y=mean, geom="point", shape=18, size = 4, show.legend=FALSE) +
  facet_grid(.~CODTALHAO)
```



Normalmente, ggplot2 nao plota boxplots sem categoria; Podemos improvisar e criar uma categoria falsa

```
ggplot(dados_g2, aes(x = factor(0), y = ALT1)) +
  geom_boxplot(fill = "darkgray") +
  labs(x = "Categoria", y = "Altura (m)", title = "Boxplot para Altura") +
  stat_summary(fun.y=mean, geom="point", shape=18, size = 4, show.legend=FALSE) +
  facet_grid(.~CODTALHAO)
```



10.4) Linhas

Dado utilizado: Orange: Crescimento de árvores de laranja.

```
head(Orange,10)
```

Tree	age	circumference
1	118	30
1	484	58
1	664	87
1	1004	115
1	1231	120
1	1372	142
1	1582	145
2	118	33
2	484	69
2	664	111

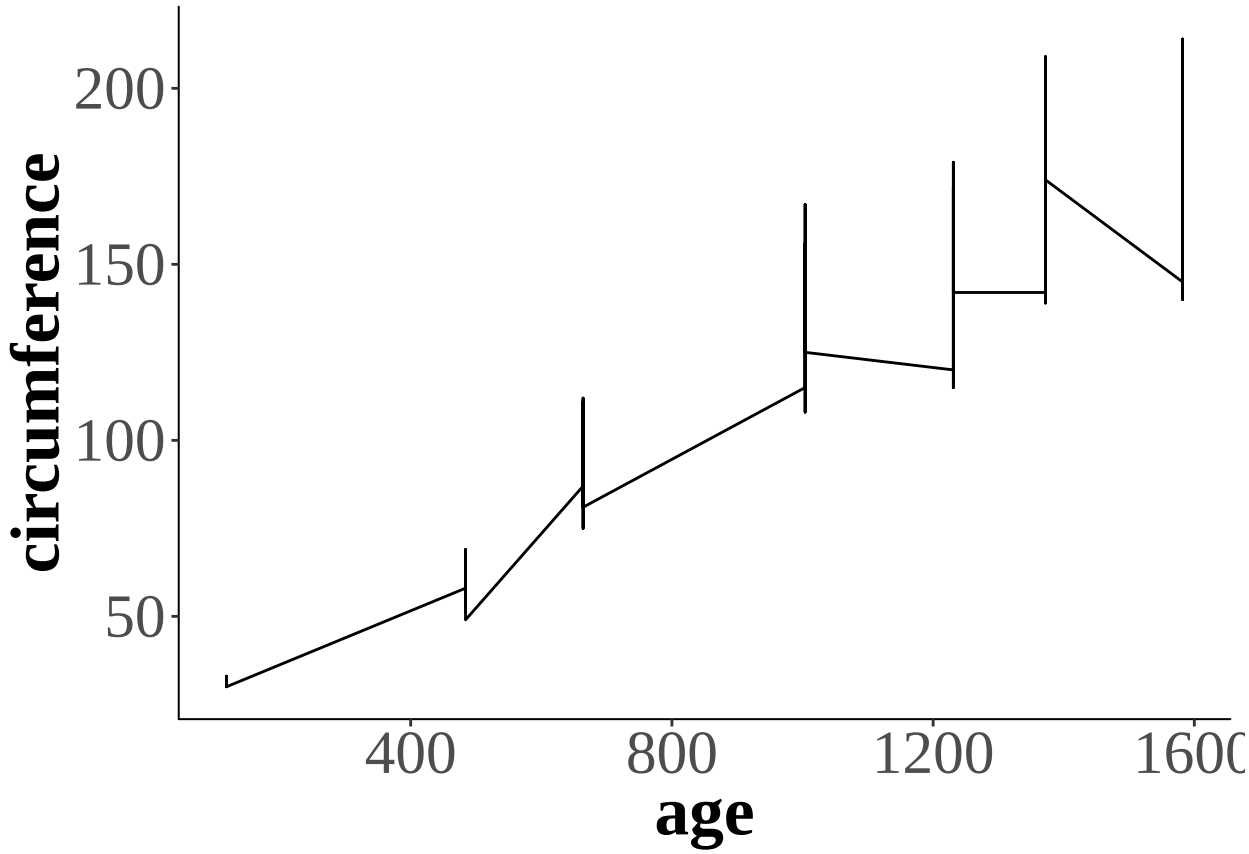
Gráfico da circunferência em função da idade.

Primeiro cira-se a base do gráfico

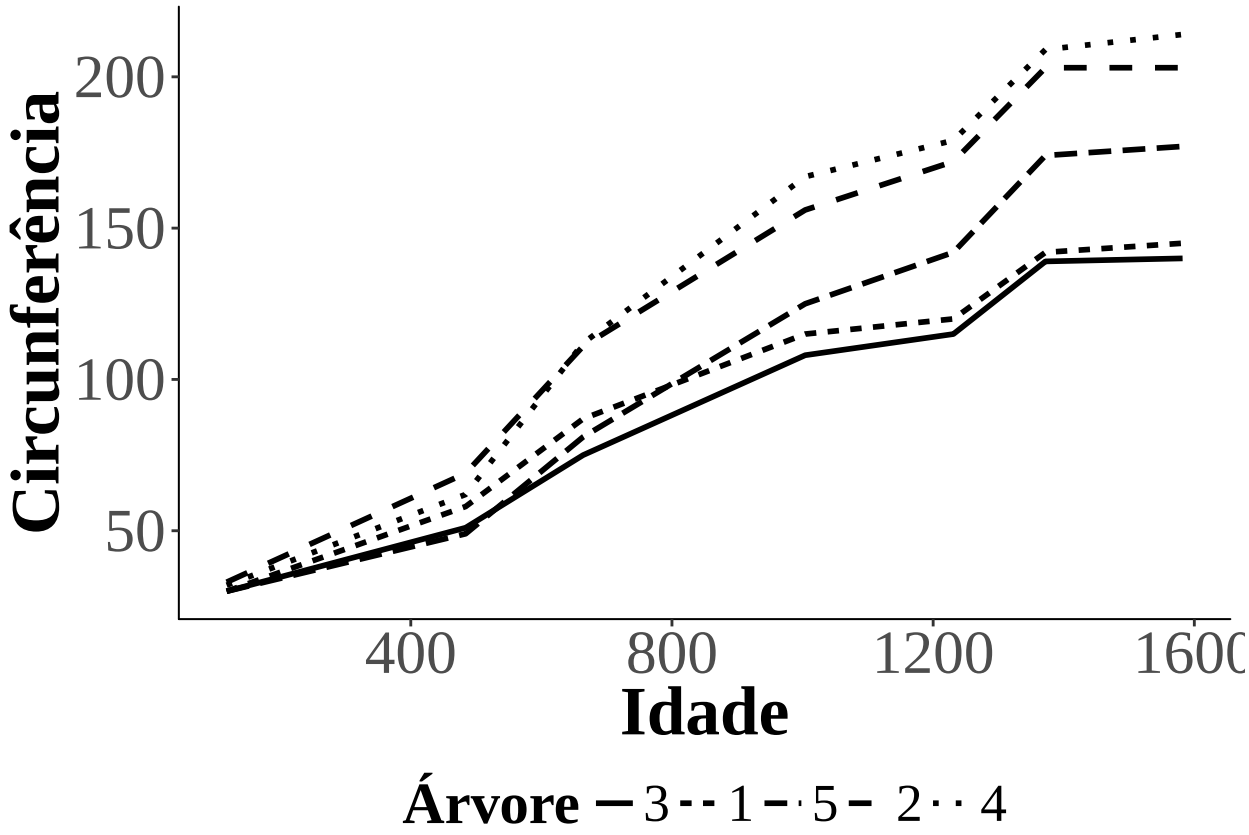
```
d <- ggplot(Orange, aes(x = age, y = circumference)) +
  theme_bw(base_family = "serif") +
  theme(
```

```
    legend.position = "bottom",
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.border = element_blank(),
    plot.title = element_text(size = 26, face = "bold", hjust = 0.5),
    axis.title = element_text(size = 26, face = "bold"),
    axis.text = element_text(size = 23),
    axis.line.x = element_line(color = "black"),
    axis.line.y = element_line(color = "black"),
    strip.text.x = element_text(size = 22,face = "bold"),
    legend.text = element_text(size = 20),
    legend.title = element_text(size = 22, face="bold") ) +
  scale_colour_grey(start = 0.6, end = 0.2) +
  scale_fill_grey(start = 0.6, end = 0.2)

d + geom_line()
```




```
# Linhas por árvore
d + geom_line(aes(linetype=Tree), size=1) +
  labs(x = "Idade", y = "Circunferência", linetype="Árvore" )
```



10.5) Exportar Graficos

Podemos combinar graficos feitos com o pacote ggplot2 pelo comando grid.arrange Para isso salvamos os graficos desejados em variaveis

```
disp <- a + geom_point(size = 4, aes(color=DESCCATEGORIA)) +
  labs(x="DAP (cm)", y="Altura (m)", colour = "Categoria", title = "Grafico de Dispersao") +
  theme_bw(base_family = "serif") +
  theme(
    legend.position = "bottom",
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.border = element_blank(),
    plot.title = element_text(size = 26, face = "bold", hjust = 0.5),
    axis.title = element_text(size = 26, face = "bold"),
    axis.text = element_text(size = 23),
    axis.line.x = element_line(color = "black"),
    axis.line.y = element_line(color = "black"),
    strip.text.x = element_text(size = 22,face = "bold"),
    legend.text = element_text(size = 20),
    legend.title = element_text(size = 22, face="bold") ) +
  coord_cartesian(ylim = c(15,35)) +
  geom_smooth(method = "lm", color = "red", se=FALSE) +
  facet_grid(.~CODTALHAO) +
  scale_colour_grey(start = 0.6, end = 0.2)
```

```
boxp <- c + geom_boxplot(fill = "darkgray") +
  labs(x = "Categoria", y = "Altura (m)", title = "Boxplot para Altura") +
  stat_summary(fun.y=mean, geom="point", shape=18, size = 4, show.legend=FALSE)

hist1 <- b + geom_histogram(binwidth = 2) +
  labs(x = "DAP (cm)", y = "Densidade", title = "Histograma do DAP") +
  geom_density(color = "grey32") +
  facet_grid(.~DESCCATEGORIA)

hist2 <- b + geom_histogram(binwidth = 2) +
  labs(x = "DAP (cm)", y = "Densidade", title = "Histograma do DAP") +
  coord_cartesian(xlim = c(10,25), ylim = c(0, .3)) +
  geom_density(color = "grey32")
```

Para salvarmos o grafico, basta usar o comando ggsave Por padrao este comando salva o ultimo grafico gerado para salvarmos um objeto especifico, basta indica-lo na função:

Salvar um dos graficos gerados pelo ggplot2

```
ggsave("disp.jpeg",  disp, width = 12, height = 10)
ggsave("boxp.jpeg",  boxp, width = 12, height = 10)
ggsave("hist1.jpeg",  hist1, width = 12, height = 10)
ggsave("hist2.jpeg",  hist2, width = 12, height = 10)
```

11) Atividade

Base de dados utilizada:

iris: Composta por comprimento e largura de sépalas e pétalas de 3 espécies de flores.

```
iris
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.4	3.4	1.7	0.2	setosa
5.1	3.7	1.5	0.4	setosa
4.6	3.6	1.0	0.2	setosa
5.1	3.3	1.7	0.5	setosa
4.8	3.4	1.9	0.2	setosa
5.0	3.0	1.6	0.2	setosa
5.0	3.4	1.6	0.4	setosa
5.2	3.5	1.5	0.2	setosa
5.2	3.4	1.4	0.2	setosa
4.7	3.2	1.6	0.2	setosa
4.8	3.1	1.6	0.2	setosa
5.4	3.4	1.5	0.4	setosa
5.2	4.1	1.5	0.1	setosa
5.5	4.2	1.4	0.2	setosa
4.9	3.1	1.5	0.2	setosa
5.0	3.2	1.2	0.2	setosa
5.5	3.5	1.3	0.2	setosa
4.9	3.6	1.4	0.1	setosa
4.4	3.0	1.3	0.2	setosa
5.1	3.4	1.5	0.2	setosa
5.0	3.5	1.3	0.3	setosa
4.5	2.3	1.3	0.3	setosa
4.4	3.2	1.3	0.2	setosa
5.0	3.5	1.6	0.6	setosa
5.1	3.8	1.9	0.4	setosa
4.8	3.0	1.4	0.3	setosa
5.1	3.8	1.6	0.2	setosa
4.6	3.2	1.4	0.2	setosa
5.3	3.7	1.5	0.2	setosa
5.0	3.3	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
5.5	2.3	4.0	1.3	versicolor
6.5	2.8	4.6	1.5	versicolor
5.7	2.8	4.5	1.3	versicolor
6.3	3.3	4.7	1.6	versicolor
4.9	2.4	3.3	1.0	versicolor
6.6	2.9	4.6	1.3	versicolor
5.2	2.7	3.9	1.4	versicolor
5.0	2.0	3.5	1.0	versicolor
5.9	3.0	4.2	1.5	versicolor
6.0	2.2	4.0	1.0	versicolor
6.1	2.9	4.7	1.4	versicolor
5.6	2.9	3.6	1.3	versicolor
6.7	3.1	4.4	1.4	versicolor
5.6	3.0	4.5	1.5	versicolor
5.8	2.7	4.1	1.0	versicolor
6.2	2.2	4.5	1.5	versicolor
5.6	2.5	3.9	1.1	versicolor
5.9	3.2	4.8	1.8	versicolor
6.1	2.8	4.0	1.3	versicolor

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
6.3	2.5	4.9	1.5	versicolor
6.1	2.8	4.7	1.2	versicolor
6.4	2.9	4.3	1.3	versicolor
6.6	3.0	4.4	1.4	versicolor
6.8	2.8	4.8	1.4	versicolor
6.7	3.0	5.0	1.7	versicolor
6.0	2.9	4.5	1.5	versicolor
5.7	2.6	3.5	1.0	versicolor
5.5	2.4	3.8	1.1	versicolor
5.5	2.4	3.7	1.0	versicolor
5.8	2.7	3.9	1.2	versicolor
6.0	2.7	5.1	1.6	versicolor
5.4	3.0	4.5	1.5	versicolor
6.0	3.4	4.5	1.6	versicolor
6.7	3.1	4.7	1.5	versicolor
6.3	2.3	4.4	1.3	versicolor
5.6	3.0	4.1	1.3	versicolor
5.5	2.5	4.0	1.3	versicolor
5.5	2.6	4.4	1.2	versicolor
6.1	3.0	4.6	1.4	versicolor
5.8	2.6	4.0	1.2	versicolor
5.0	2.3	3.3	1.0	versicolor
5.6	2.7	4.2	1.3	versicolor
5.7	3.0	4.2	1.2	versicolor
5.7	2.9	4.2	1.3	versicolor
6.2	2.9	4.3	1.3	versicolor
5.1	2.5	3.0	1.1	versicolor
5.7	2.8	4.1	1.3	versicolor
6.3	3.3	6.0	2.5	virginica
5.8	2.7	5.1	1.9	virginica
7.1	3.0	5.9	2.1	virginica
6.3	2.9	5.6	1.8	virginica
6.5	3.0	5.8	2.2	virginica
7.6	3.0	6.6	2.1	virginica
4.9	2.5	4.5	1.7	virginica
7.3	2.9	6.3	1.8	virginica
6.7	2.5	5.8	1.8	virginica
7.2	3.6	6.1	2.5	virginica
6.5	3.2	5.1	2.0	virginica
6.4	2.7	5.3	1.9	virginica
6.8	3.0	5.5	2.1	virginica
5.7	2.5	5.0	2.0	virginica
5.8	2.8	5.1	2.4	virginica
6.4	3.2	5.3	2.3	virginica
6.5	3.0	5.5	1.8	virginica
7.7	3.8	6.7	2.2	virginica
7.7	2.6	6.9	2.3	virginica
6.0	2.2	5.0	1.5	virginica
6.9	3.2	5.7	2.3	virginica
5.6	2.8	4.9	2.0	virginica
7.7	2.8	6.7	2.0	virginica
6.3	2.7	4.9	1.8	virginica

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
6.7	3.3	5.7	2.1	virginica
7.2	3.2	6.0	1.8	virginica
6.2	2.8	4.8	1.8	virginica
6.1	3.0	4.9	1.8	virginica
6.4	2.8	5.6	2.1	virginica
7.2	3.0	5.8	1.6	virginica
7.4	2.8	6.1	1.9	virginica
7.9	3.8	6.4	2.0	virginica
6.4	2.8	5.6	2.2	virginica
6.3	2.8	5.1	1.5	virginica
6.1	2.6	5.6	1.4	virginica
7.7	3.0	6.1	2.3	virginica
6.3	3.4	5.6	2.4	virginica
6.4	3.1	5.5	1.8	virginica
6.0	3.0	4.8	1.8	virginica
6.9	3.1	5.4	2.1	virginica
6.7	3.1	5.6	2.4	virginica
6.9	3.1	5.1	2.3	virginica
5.8	2.7	5.1	1.9	virginica
6.8	3.2	5.9	2.3	virginica
6.7	3.3	5.7	2.5	virginica
6.7	3.0	5.2	2.3	virginica
6.3	2.5	5.0	1.9	virginica
6.5	3.0	5.2	2.0	virginica
6.2	3.4	5.4	2.3	virginica
5.9	3.0	5.1	1.8	virginica

1)

Calcular o comprimento médio das pétalas da espécie virginica

```
mean(iris[iris$Species=="virginica", "Petal.Length"], na.rm=T)
```

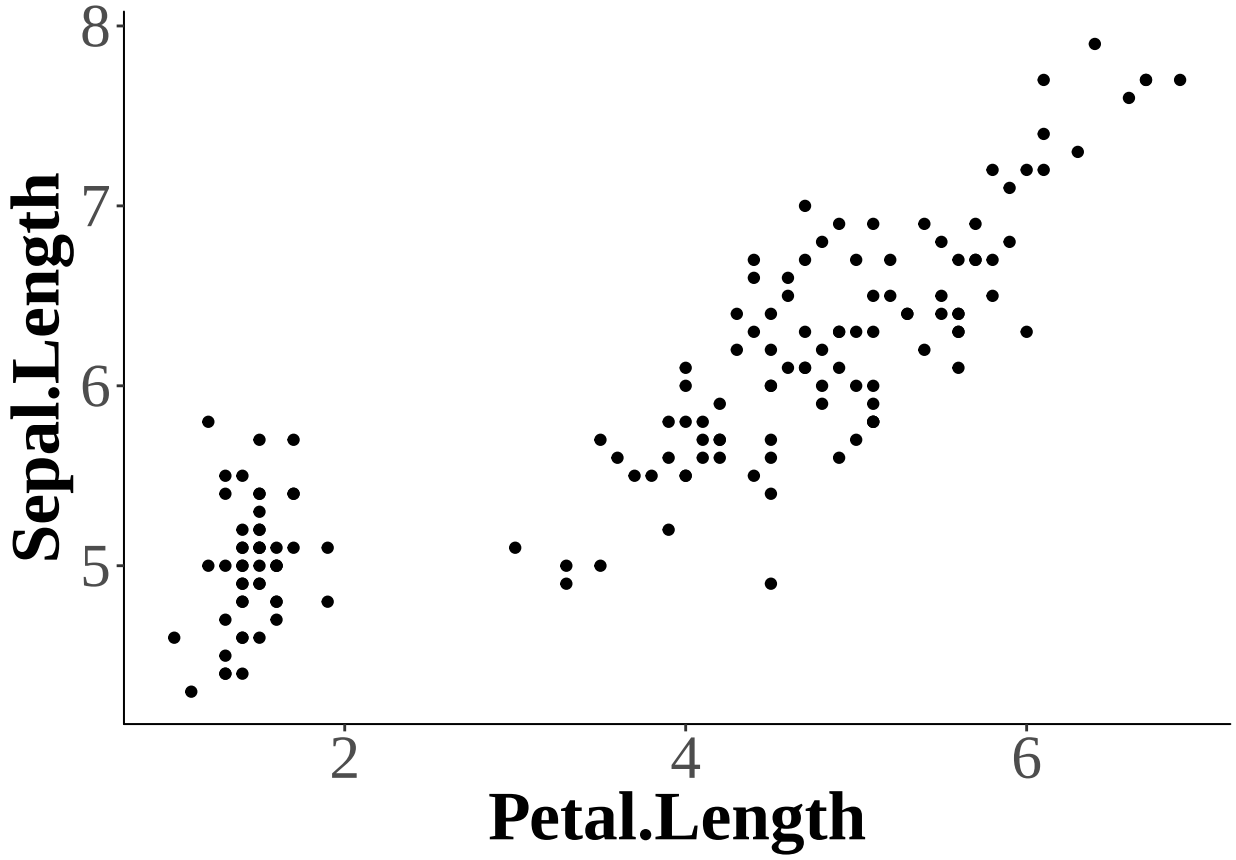
```
## [1] 5.552
```

2)

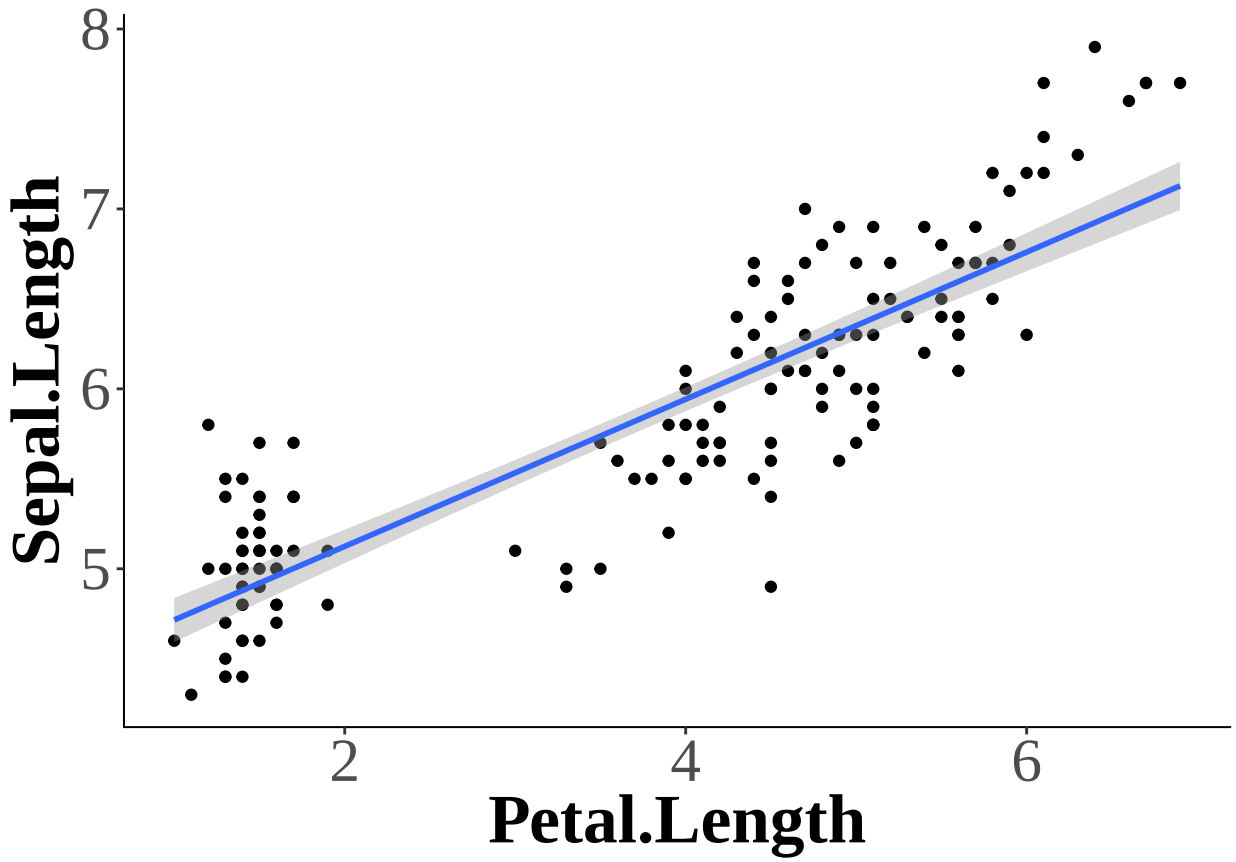
Criar um Gráfico de dispersão do comprimento da pétala em função do comprimento da sépala, por espécie, com linha de tendência linear

```
base_grafico <- ggplot(iris, aes(x=Petal.Length, y = Sepal.Length)) +
  theme_bw(base_family = "serif") +
  theme(
    legend.position = "bottom",
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.border = element_blank(),
    plot.title = element_text(size = 26, face = "bold", hjust = 0.5),
    axis.title = element_text(size = 26, face = "bold"),
    axis.text = element_text(size = 23),
    axis.line.x = element_line(color = "black"),
    axis.line.y = element_line(color = "black"),
    strip.text.x = element_text(size = 22,face = "bold"),
    legend.text = element_text(size = 20),
    legend.title = element_text(size = 22, face="bold") )
```

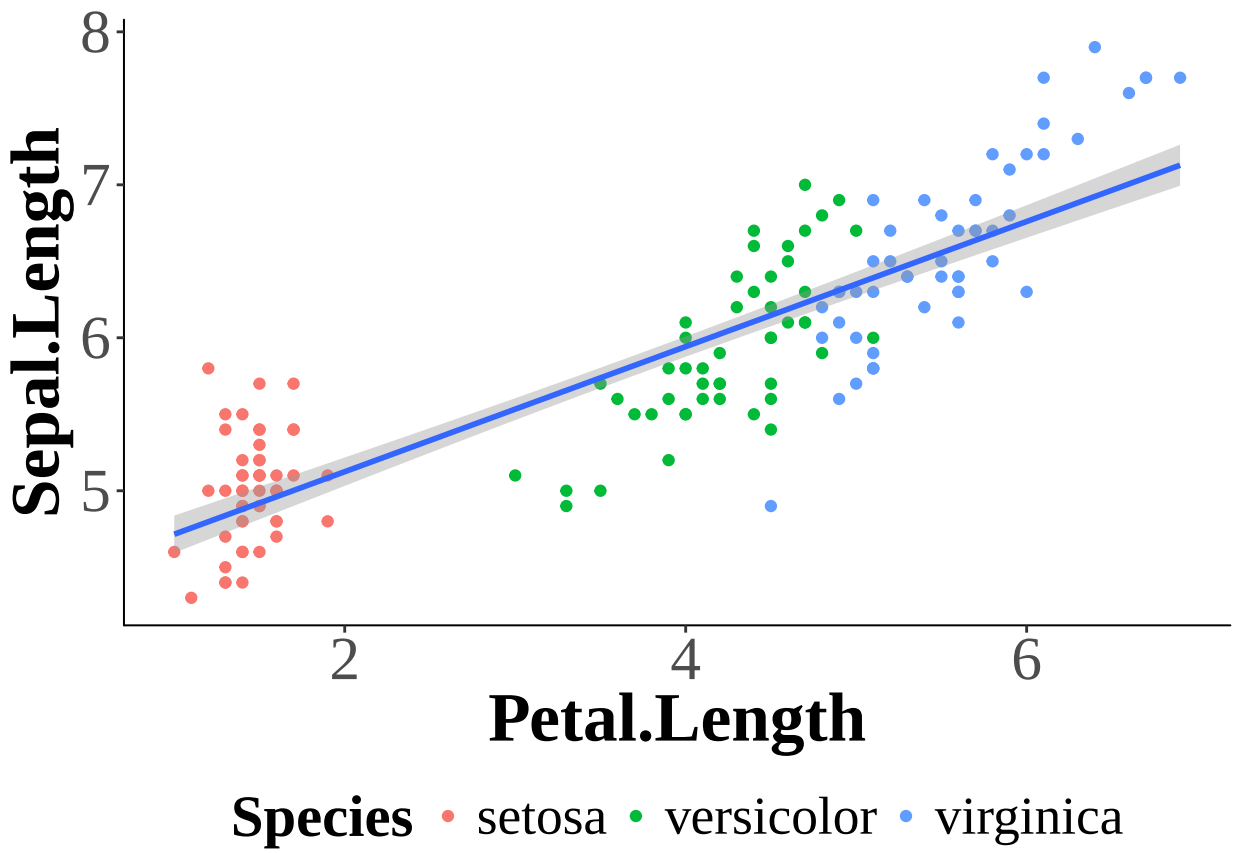
```
base_grafico + geom_point()
```



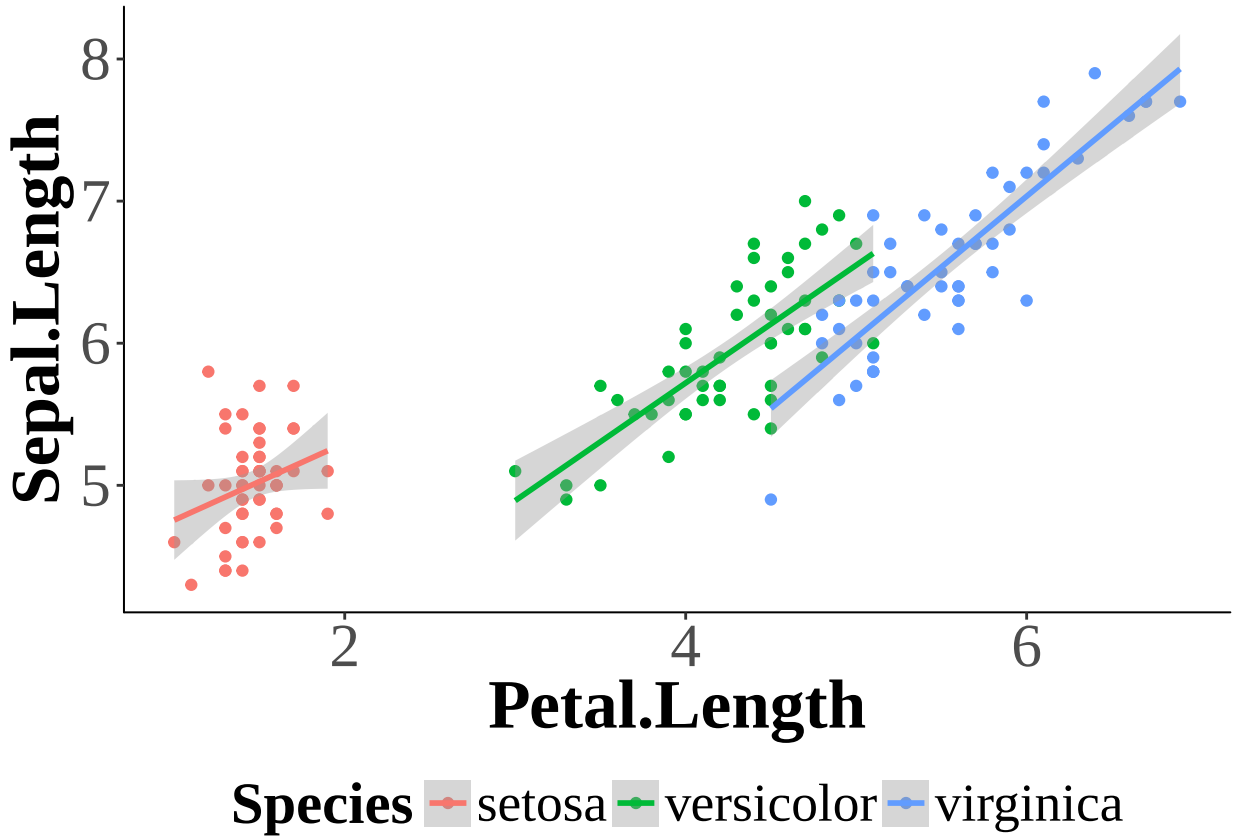

```
base_grafico + geom_point() + geom_smooth(method = "lm")
```




```
base_grafico + geom_point(aes(color=Species)) + geom_smooth(method = "lm")
```




```
base_grafico + geom_point(aes(color=Species)) + geom_smooth(aes(color=Species),method = "lm")
```




```
iris
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa
5.4	3.4	1.7	0.2	setosa
5.1	3.7	1.5	0.4	setosa
4.6	3.6	1.0	0.2	setosa

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.3	1.7	0.5	setosa
4.8	3.4	1.9	0.2	setosa
5.0	3.0	1.6	0.2	setosa
5.0	3.4	1.6	0.4	setosa
5.2	3.5	1.5	0.2	setosa
5.2	3.4	1.4	0.2	setosa
4.7	3.2	1.6	0.2	setosa
4.8	3.1	1.6	0.2	setosa
5.4	3.4	1.5	0.4	setosa
5.2	4.1	1.5	0.1	setosa
5.5	4.2	1.4	0.2	setosa
4.9	3.1	1.5	0.2	setosa
5.0	3.2	1.2	0.2	setosa
5.5	3.5	1.3	0.2	setosa
4.9	3.6	1.4	0.1	setosa
4.4	3.0	1.3	0.2	setosa
5.1	3.4	1.5	0.2	setosa
5.0	3.5	1.3	0.3	setosa
4.5	2.3	1.3	0.3	setosa
4.4	3.2	1.3	0.2	setosa
5.0	3.5	1.6	0.6	setosa
5.1	3.8	1.9	0.4	setosa
4.8	3.0	1.4	0.3	setosa
5.1	3.8	1.6	0.2	setosa
4.6	3.2	1.4	0.2	setosa
5.3	3.7	1.5	0.2	setosa
5.0	3.3	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
5.5	2.3	4.0	1.3	versicolor
6.5	2.8	4.6	1.5	versicolor
5.7	2.8	4.5	1.3	versicolor
6.3	3.3	4.7	1.6	versicolor
4.9	2.4	3.3	1.0	versicolor
6.6	2.9	4.6	1.3	versicolor
5.2	2.7	3.9	1.4	versicolor
5.0	2.0	3.5	1.0	versicolor
5.9	3.0	4.2	1.5	versicolor
6.0	2.2	4.0	1.0	versicolor
6.1	2.9	4.7	1.4	versicolor
5.6	2.9	3.6	1.3	versicolor
6.7	3.1	4.4	1.4	versicolor
5.6	3.0	4.5	1.5	versicolor
5.8	2.7	4.1	1.0	versicolor
6.2	2.2	4.5	1.5	versicolor
5.6	2.5	3.9	1.1	versicolor
5.9	3.2	4.8	1.8	versicolor
6.1	2.8	4.0	1.3	versicolor
6.3	2.5	4.9	1.5	versicolor
6.1	2.8	4.7	1.2	versicolor
6.4	2.9	4.3	1.3	versicolor

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
6.6	3.0	4.4	1.4	versicolor
6.8	2.8	4.8	1.4	versicolor
6.7	3.0	5.0	1.7	versicolor
6.0	2.9	4.5	1.5	versicolor
5.7	2.6	3.5	1.0	versicolor
5.5	2.4	3.8	1.1	versicolor
5.5	2.4	3.7	1.0	versicolor
5.8	2.7	3.9	1.2	versicolor
6.0	2.7	5.1	1.6	versicolor
5.4	3.0	4.5	1.5	versicolor
6.0	3.4	4.5	1.6	versicolor
6.7	3.1	4.7	1.5	versicolor
6.3	2.3	4.4	1.3	versicolor
5.6	3.0	4.1	1.3	versicolor
5.5	2.5	4.0	1.3	versicolor
5.5	2.6	4.4	1.2	versicolor
6.1	3.0	4.6	1.4	versicolor
5.8	2.6	4.0	1.2	versicolor
5.0	2.3	3.3	1.0	versicolor
5.6	2.7	4.2	1.3	versicolor
5.7	3.0	4.2	1.2	versicolor
5.7	2.9	4.2	1.3	versicolor
6.2	2.9	4.3	1.3	versicolor
5.1	2.5	3.0	1.1	versicolor
5.7	2.8	4.1	1.3	versicolor
6.3	3.3	6.0	2.5	virginica
5.8	2.7	5.1	1.9	virginica
7.1	3.0	5.9	2.1	virginica
6.3	2.9	5.6	1.8	virginica
6.5	3.0	5.8	2.2	virginica
7.6	3.0	6.6	2.1	virginica
4.9	2.5	4.5	1.7	virginica
7.3	2.9	6.3	1.8	virginica
6.7	2.5	5.8	1.8	virginica
7.2	3.6	6.1	2.5	virginica
6.5	3.2	5.1	2.0	virginica
6.4	2.7	5.3	1.9	virginica
6.8	3.0	5.5	2.1	virginica
5.7	2.5	5.0	2.0	virginica
5.8	2.8	5.1	2.4	virginica
6.4	3.2	5.3	2.3	virginica
6.5	3.0	5.5	1.8	virginica
7.7	3.8	6.7	2.2	virginica
7.7	2.6	6.9	2.3	virginica
6.0	2.2	5.0	1.5	virginica
6.9	3.2	5.7	2.3	virginica
5.6	2.8	4.9	2.0	virginica
7.7	2.8	6.7	2.0	virginica
6.3	2.7	4.9	1.8	virginica
6.7	3.3	5.7	2.1	virginica
7.2	3.2	6.0	1.8	virginica
6.2	2.8	4.8	1.8	virginica

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
6.1	3.0	4.9	1.8	virginica
6.4	2.8	5.6	2.1	virginica
7.2	3.0	5.8	1.6	virginica
7.4	2.8	6.1	1.9	virginica
7.9	3.8	6.4	2.0	virginica
6.4	2.8	5.6	2.2	virginica
6.3	2.8	5.1	1.5	virginica
6.1	2.6	5.6	1.4	virginica
7.7	3.0	6.1	2.3	virginica
6.3	3.4	5.6	2.4	virginica
6.4	3.1	5.5	1.8	virginica
6.0	3.0	4.8	1.8	virginica
6.9	3.1	5.4	2.1	virginica
6.7	3.1	5.6	2.4	virginica
6.9	3.1	5.1	2.3	virginica
5.8	2.7	5.1	1.9	virginica
6.8	3.2	5.9	2.3	virginica
6.7	3.3	5.7	2.5	virginica
6.7	3.0	5.2	2.3	virginica
6.3	2.5	5.0	1.9	virginica
6.5	3.0	5.2	2.0	virginica
6.2	3.4	5.4	2.3	virginica
5.9	3.0	5.1	1.8	virginica

12) Exportar dados

Para se exportar dados no R, utiliza-se a família de funções write, como write.table, para o formato .txt e write.csv para o formato .csv;

O modelo utilizado e: write.csv2(nome do objeto a exportar, “nome do arquivo.csv”).

Semelhante a família das funções read, utiliza-se write.csv2 para arquivos com separador “;” e dec “,” (padrão nacional/europeu); e write.csv para arquivos com separador “,” e dec “.” (padrão americano).

Deve-se atentar a extensão do arquivo no nome:

```
write.csv2(iris, "iris.csv")
```

Por padrão os nomes das linhas são inseridas no arquivo. Para que isto não aconteça, utiliza-se row.names = FALSE:

```
write.csv2(dados, "iris.csv", row.names = FALSE)
```