

PENGANTAR BAHASA R

Bahasa R merupakan salah satu software gratis yang sangat populer di Indonesia. Kemudahan penggunaan serta banyaknya besarnya dukungan komunitas membuat R menjadi salah satu bahasa pemrograman paling populer di dunia. Keunggulan bahasa R yaitu :

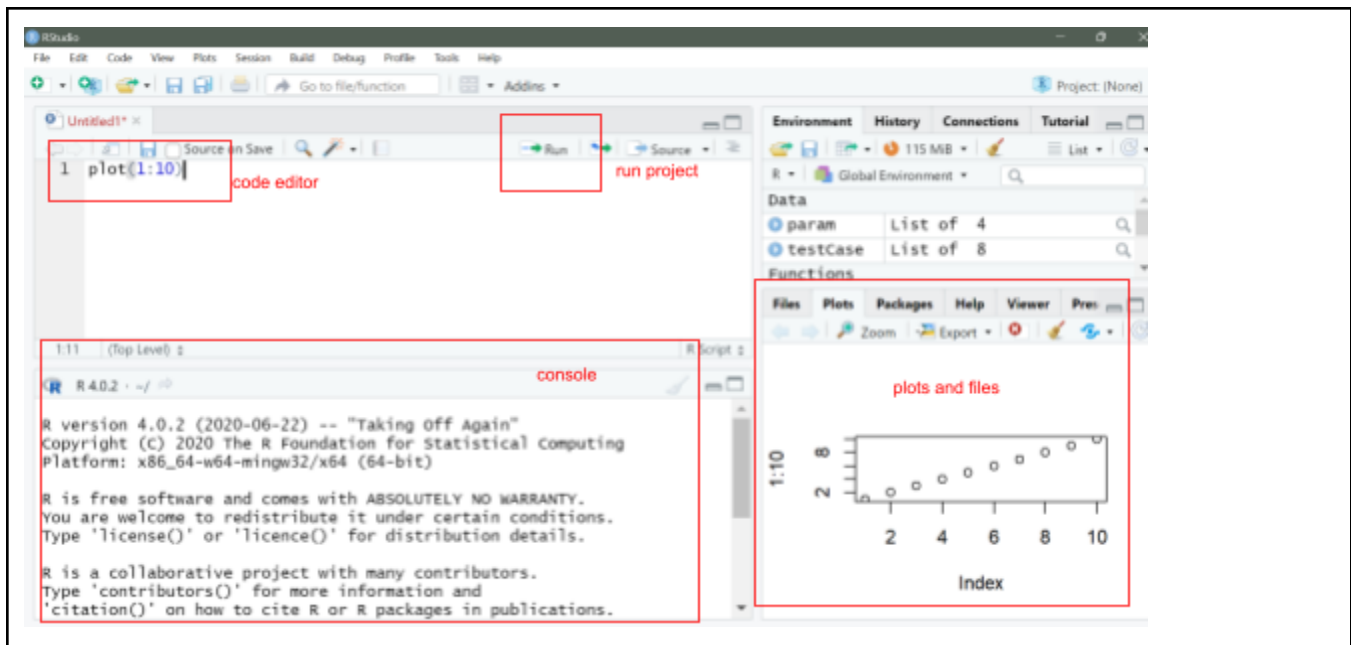
1. R adalah open source dan tersedia secara bebas.
2. R tersedia untuk sistem operasi Windows, Mac dan Linux.
3. R memiliki seperangkat alat yang luas dan koheren untuk analisis statistik.
4. R memiliki fasilitas grafis yang luas dan sangat fleksibel yang mampu memproduksi angka kualitas publikasi.
5. R memiliki serangkaian 'paket' yang tersedia secara bebas untuk memperluas kemampuan R.
6. R memiliki jaringan dukungan yang luas dengan banyak online dan tersedia secara bebas dokumen.

1. Instalasi R

Sebelum menjalankan program R, yang pertama dilakukan adalah menginstal R. R tersedia secara gratis untuk sistem operasi Windows, Mac dan Linux dari situs web [Comprehensive R Archive Network \(CRAN\)](https://cran.r-project.org/). Untuk pengguna Windows dan Mac, disarankan untuk mengunduh dan menginstal versi biner yang telah dikompilasi sebelumnya. Install sesuai sistem operasi yang kalian gunakan.

Kemudian kita akan menggunakan Integrated Development Environment (IDE) populer yang disebut RStudio. RStudio dapat dianggap sebagai add-on untuk R yang menyediakan antarmuka yang lebih ramah pengguna, menggabungkan Konsol R, editor script, dan fungsionalitas berguna lainnya (seperti penurunan harga R dan integrasi Git Hub). RStudio tersedia secara gratis untuk sistem operasi Windows, Mac dan Linux dan dapat diunduh dari situs [RStudio](https://www.rstudio.com/). Anda harus memilih versi 'RStudio Desktop'.

Tampilan jendela R Studio



2. Dasar-dasar Bahasa R

2.1 Operasi dasar

- Operasi aritmatika sederhana

```
> #Proses aritmatika sederhana
> 6 + 2
[1] 8
> 7 - 8
[1] -1
> 9 / 2
[1] 4.5
> 6 * 2
[1] 12
```

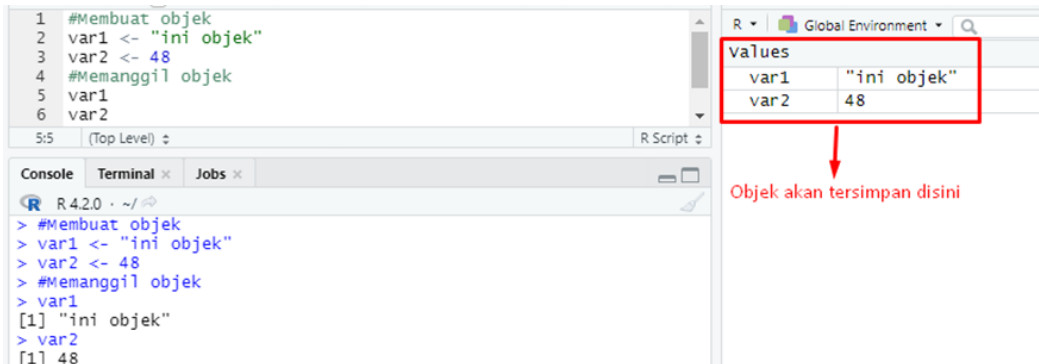
- Fungsi aritmatika dasar

- Fungsi Logaritma dan Eksponensial : $\log_2(x)$, $\log_{10}(x)$, $\exp(x)$
- Fungsi Trigonometri: $\cos(x)$, $\sin(x)$, $\tan(x)$, $\arccos(x)$, $\arcsin(x)$, $\arctan(x)$
- Fungsi Matematika Lainnya: $\text{abs}(x)$: absolute value; $\text{sqrt}(x)$: square root

```
> log(1)
[1] 0
> log10(10)
[1] 1
> exp(1)
[1] 2.718282
> sqrt(4)
[1] 2
> 4^2
[1] 16
> pi
[1] 3.141593
```

2.2 Objek-objek pada R

2.2.1 Membuat objek



The screenshot shows the R Studio interface. The script editor on the left contains the following code:

```
1 #Membuat objek
2 var1 <- "ini objek"
3 var2 <- 48
4 #Memanggil objek
5 var1
6 var2
```

The console on the bottom left shows the output of the commands:

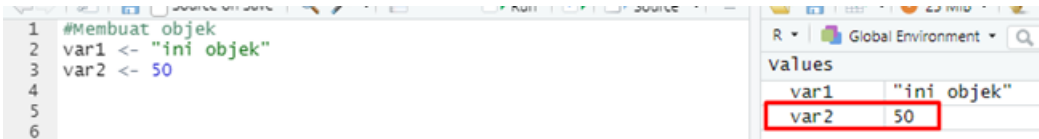
```
> #Membuat objek
> var1 <- "ini objek"
> var2 <- 48
> #Memanggil objek
> var1
[1] "ini objek"
> var2
[1] 48
```

The Global Environment pane on the right shows the values of the objects:

values	
var1	"ini objek"
var2	48

A red box highlights the values table, and a red arrow points to it with the text "Objek akan tersimpan disini".

Untuk mengganti nilai dari suatu objek, kita tinggal meng-assign kembali nilainya ke dalam objek tersebut



The screenshot shows the R Studio interface. The script editor on the left contains the following code:

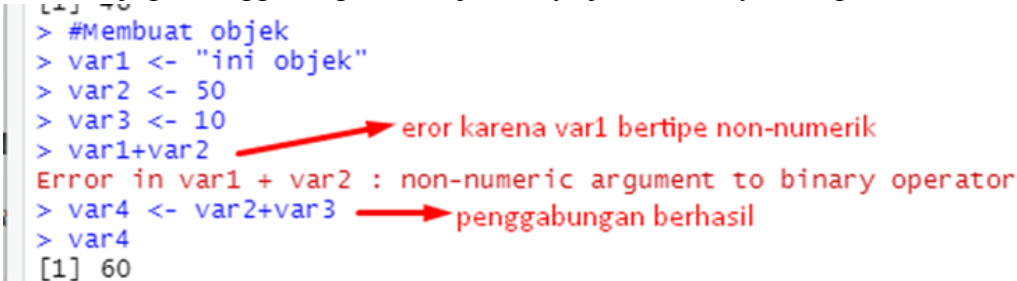
```
1 #Membuat objek
2 var1 <- "ini objek"
3 var2 <- 50
4
5
6
```

The Global Environment pane on the right shows the values of the objects:

values	
var1	"ini objek"
var2	50

A red box highlights the values table.

Kita bisa juga menggabungkan 2 objek hanya jika keduanya bertipe numerik



The screenshot shows the R Studio console with the following code and output:

```
> #Membuat objek
> var1 <- "ini objek"
> var2 <- 50
> var3 <- 10
> var1+var2
Error in var1 + var2 : non-numeric argument to binary operator
> var4 <- var2+var3
> var4
[1] 60
```

Red arrows point to the error message and the successful operation with the text "eror karena var1 bertipe non-numerik" and "penggabungan berhasil".

2.2.2 Penamaan objek

Berikut aturan penamaan objek di R.

- Menggunakan kombinasi alfabet (a-z, A-Z), angka (0-9), titik atau *underscore*.
- Diawali alphabet, titik atau *underscore*. Tidak boleh diawali dengan angka.
- Tidak mengandung spasi, tab atau karakter khusus seperti !, @, # dan lainnya.
- Sebaiknya tidak menggunakan beberapa penamaan atau nilai yang sudah digunakan oleh R (function dan keyword lainnya). Misalnya c, q, TRUE, FALSE, df, dt, rnorm, runif, rf, exp, dan lain-lain. Untuk mengetahui nama-nama yang sudah digunakan oleh R Anda dapat mengetikkan perintah [?reserved](#) di console RStudio Anda.



```
R 4.2.0 ~/  
> obj1 <- "objek"  
> obj1 <- "objek"  
> obj.1 <- "objek"  
> obj_1 <- "objek"  
> 1obj <- "objek"  
Error: unexpected symbol in "1obj" → Diawali dengan angka  
> !obj <- "objek"  
Error in !obj <- "objek" : object 'obj' not found → Diawali dengan spesial karakter !  
> function <- "objek"  
Error: unexpected assignment in "function <-" → Menggunakan kata yang sudah ada di R
```

2.3 Menggunakan function di R

Function dapat dikatakan sebagai sebuah objek yang mengandung serangkaian instruksi untuk menjalankan sebuah tugas spesifik. Salah satu contoh function sederhana adalah `c()` function. Function ini merupakan singkatan dari *concatenate* dan digunakan untuk menggabungkan serangkaian nilai dan menyimpannya di data struktur yang disebut **vektor** (lebih lanjutnya ada di Bab 3)

```
my_vec <- c(2, 3, 1, 6, 4, 3, 3, 7)
```

```
> my_vec <- c(2, 3, 1, 6, 4, 3, 3, 7)  
> my_vec  
[1] 2 3 1 6 4 3 3 7
```

Selain itu, kita bisa mengisi nilai vektor dengan angka-angka secara berurutan dengan simbol “:”

```
> vec1 <- 1:10  
> vec1  
[1] 1 2 3 4 5 6 7 8 9 10  
> vec2 <- 10:1  
> vec2  
[1] 10 9 8 7 6 5 4 3 2 1
```

Adapun function lain untuk membuat vektor urutan yaitu `seq()` dan `rep()`
`seq()` digunakan untuk membuat vektor urutan dengan kelipatan tertentu

```
vec <- seq(from = 1, to = 5, by = 0.5)
```

```
> vec <- seq(from = 1, to = 5, by = 0.5) Dimulai dari 1 sampai 5  
> vec dengan kelipatan 0.5  
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

rep() digunakan untuk membuat vektor urutan dengan repetisi tertentu

```
vec1 <- rep(2, times = 10) #Angka 2 sebanyak 10x  
vec2 <- rep('abc', times = 5) #String ab sebanyak 5x  
vec3 <- rep(1:5, times = 3) #Urutan 1 s.d 5 di-assign sebanyak 3x  
vec4 <- rep(1:5, each = 3) #Setiap angka di-assign sebanyak 3x  
vec5 <- rep(c(1,3,4,7), each = 3) #Setiap angka di-assign sebanyak 3x
```

```
> vec1 <- rep(2, times = 10) #Angka 2 sebanyak 10x  
> vec1  
[1] 2 2 2 2 2 2 2 2 2 2  
> vec2 <- rep('abc', times = 5) #String ab sebanyak 5x  
> vec2  
[1] "abc" "abc" "abc" "abc" "abc"  
> vec3 <- rep(1:5, times = 3) #Urutan 1 s.d 5 di-assign sebanyak 3x  
> vec3  
[1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5  
> vec4 <- rep(1:5, each = 3) #Setiap angka di-assign sebanyak 3x  
> vec4  
[1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5  
> vec5 <- rep(c(1,3,4,7), each = 3) #Setiap angka di-assign sebanyak 3x  
> vec5  
[1] 1 1 1 3 3 3 4 4 4 7 7 7
```

2.4 Bekerja dengan vektor

2.4.1 Ekstraksi elemen

a. Positional index

Kita dapat mengecek nilai suatu vektor pada indeks posisi tertentu

```
> my_vec  
[1] 2 3 1 6 4 3 3 7  
> my_vec[3] #Mengecek nilai pada indeks ke-3  
[1] 1  
> val_3 <- my_vec[3] #Menyimpan nilai indeks ke-3 ke val_3  
> val_3  
[1] 1
```

Selain itu, kita bisa mengecek indeks posisi dari nilai yang ada pada vektor

```
my_vec[c(1, 5, 6, 7)] #Mengecek indeks posisi 1,5,6 dan 7  
> my_vec[c(1, 5, 6, 7)] #Mengecek indeks posisi 1,5,6 dan 7  
[1] 2 4 3 3
```

b. Logical Index

Kita bisa meng-ekstrak data dari vektor yang menggunakan logical expression sebagai index.

```
#Menampilkan nilai pada my_vec yang lebih dari 4  
my_vec[my_vec > 4]
```

```

> #Menampilkan nilai my_vec yang lebih dari 4
> my_vec[my_vec > 4]
[1] 6 7

> #contoh lain
> my_vec[my_vec < 4]
[1] 2 3 1 3 3
> my_vec[my_vec <= 4]
[1] 2 3 1 4 3 3
> my_vec[my_vec == 4]
[1] 4
> my_vec[my_vec != 4]
[1] 2 3 1 6 3 3 7

```

Selain itu, kita juga bisa menggunakan boolean expressions (AND, OR, dll)

```

> #Menyimpan nilai vektor yang kurang dari 6 dan lebih dari 2
> val26 <- my_vec[my_vec < 6 & my_vec > 2]
> val26
[1] 3 4 3 3
> #Menyimpan nilai vektor yang lebih dari 6 atau kurang dari 3
> val63 <- my_vec[my_vec > 6 | my_vec < 3]
> val63
[1] 2 1 7

```

2.4.2 Replacing elements

Kita bisa meng-replace suatu elemen vektor dengan notasi [] pada operator <-
Misal, meng-replace nilai ke-4 menjadi 500

my_vec[4] <- 500

```

> my_vec[4] <- 500
> my_vec
[1] 2 3 1 500 4 3 3 7

```

Contoh lain:

```

> #Replace nilai ke-6 dan 7 dengan 100
> my_vec[c(6,7)] <- 100
> my_vec
[1] 2 3 1 500 4 100 100 7
> #Replace nilai yang kurang dari 4 dengan 50
> my_vec[my_vec < 4] <- 50
> my_vec
[1] 50 50 50 500 4 100 100 7

```

2.4.3 Ordering elements (pengurutan)

1. Pengurutan dengan `sort()`

```

vec_sort <- sort(my_vec)
vec_sort
[1] 4 7 50 50 50 100 100 500

```

Atau

```

vec_sort2 <- sort(my_vec, decreasing = TRUE)

```

```

> vec_sort2 <- sort(my_vec , decreasing = TRUE)
> vec_sort2
[1] 500 100 100 50 50 50 7 4

```

Bisa juga melakukan

reverse urutan

```

> vec_sort3 <- rev(sort(my_vec))
> vec_sort3
[1] 500 100 100 50 50 50 7 4

```

2. Pengurutan dengan `order()`

```
Tinggi <- c(180, 155, 160, 167, 181)
```

```
Nama <- c('Tony', 'Nate', 'Banner', 'Steve', 'Clint')
```

```

> Tinggi <- c(180, 155, 160, 167, 181)
> Nama <- c('Tony', 'Nate', 'Banner', 'Steve', 'Clint')
>
> Tinggi
[1] 180 155 160 167 181
> Nama
[1] "Tony" "Nate" "Banner" "Steve" "Clint"

```

```
Urutan_tinggi <- order(Tinggi)
```

```

> Urutan_tinggi <- order(Tinggi)
> Urutan_tinggi
[1] 2 3 4 1 5

```

Penjelasan :

urutan terkecil adalah nilai ke-2 yaitu 155 dst

Kemudian, urutan nama dapat dibuat berdasarkan

Urutan_tinggi

```
Urutan_nama <- Nama[Urutan_tinggi]
```

```

> Urutan_nama <- Nama[Urutan_tinggi]
> Urutan_nama
[1] "Nate" "Banner" "Steve" "Tony" "Clint"

```

2.4.4 Vectorisation

- Membuat vektor

```
Vector1 <- c(1, 2, 3, 5, 6)
```

```

> Vector1 <- c(1, 2, 3, 5, 6)
> Vector1
[1] 1 2 3 5 6

```

- Mengalikan setiap elemen dengan 5

```
Vector1 * 5
```

```

> Vector1 * 5
[1] 5 10 15 25 30

```

- Membuat vektor kedua

```
Vector2 <- c(8, 9, 2, 5, 4)
```

```

> Vector2 <- c(8, 9, 2, 5, 4)
> Vector2
[1] 8 9 2 5 4

```

- Menambahkan 2 vektor

Vector1 + Vector2

```
> Vector1 + Vector2
[1] 9 11 5 10 10
```

- Mengalikan 2 vektor

*Vector1 * Vector2*

```
> Vector1 * Vector2
[1] 8 18 6 25 24
```

2.4.5 Missing data

Pada R, missing data sering direpresentasikan dengan *NA* atau '*Not Available*'. Data bisa menjadi *missing* karena berbagai alasan seperti mungkin mesinnya rusak, cuaca buruk ketika pengambilan data dll. Misal, kita mengumpulkan temperatur udara selama 10 hari, namun termometernya rusak pada hari ke-2 dan ke-9 sehingga tidak ada data pada hari itu.

```
temp <- c(7.2, NA, 7.1, 6.9, 6.5, 5.8, 5.8, 5.5, NA, 5.5)
```

```
> temp <- c(7.2, NA, 7.1, 6.9, 6.5, 5.8, 5.8, 5.5, NA, 5.5)
> temp
[1] 7.2 NA 7.1 6.9 6.5 5.8 5.8 5.5 NA 5.5
```

NA dapat mempengaruhi perhitungan jika tidak diatasi. Misal jika kita ingin menghitung rata-rata temperatur ketika masih ada NA

```
Mean_temp <- mean(temp)
```

```
> Mean_temp <- mean(temp)
> Mean_temp
[1] NA
```

hasil rata-ratanya adalah NA. Hal ini karena jika sebuah vektor memiliki *missing value*, maka hasil akhir yang akan keluar ketika digunakan dalam suatu kalkulasi adalah NA.

Oleh karena itu, NA dapat diatasi dengan `na.rm()` atau singkatan dari *remove NA*

```
Mean_temp <- mean(temp, na.rm = TRUE)
```

```
> Mean_temp <- mean(temp, na.rm = TRUE)
> Mean_temp
[1] 6.2875
```

2.5 Getting Help

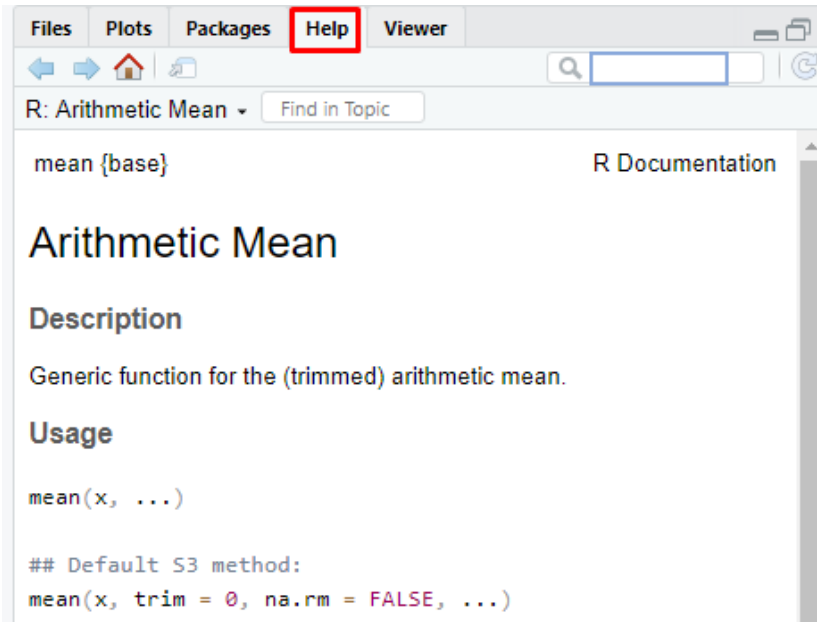
2.5.1 R Help

Untuk memanggil fitur HELP untuk sebuah function pada R dapat menggunakan

`help("nama_fungsi")` atau `?nama_fungsi`

Misal, fitur HELP untuk `mean()`

`?mean`

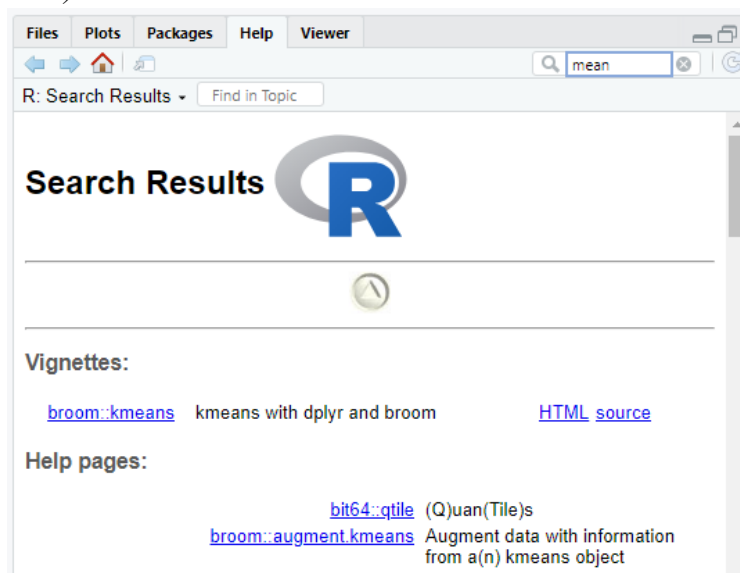


Penjelasan dari HELP tersebut

- Description : deskripsi dari function dan kegunaannya
- Usage : argumen-argumen yang terdapat pada function secara default
- Arguments : menyediakan detail dari argumen yang ada dan kegunaannya
- Details : memberikan detail function jika dibutuhkan
- Value : jika applicable, memberikan tipe dan struktur objek yang di-return by oleh function atau operator
- See also : menyediakan informasi *help page* yang relevan
- Examples : memberikan contoh penggunaan function. Bisa di akses dengan `example("nama_fungsi")`

Function `help()` berguna jika kita tau nama dari function itu. Jika kurang yakin dengan namanya dan hanya ingat *keyword*-nya, bisa menggunakan `help.search()` function.

Misal `help.search("mean")` atau `??mean`



Selain itu ada function lain seperti `apropos()` untuk menampilkan list function yang mengandung kata tertentu. Misal, menampilkan list function yang mengandung kata “mean”

`apropos("mean")`

```
> apropos("mean")
[1] ".colMeans"      ".rowMeans"      "colMeans"      "kmeans"         "mean"
[6] "mean.Date"      "mean.default"   "mean.difftime" "mean.POSIXct"   "mean.POSIXlt"
[11] "Mean_temp"      "rowMeans"       "weighted.mean"
```

2.5.2 Sumber HELP lainnya

1. General R resources

- [R-Project](#): User contributed documentation
- [The R Journal](#): Journal of the R project for statistical computing
- [Swirl](#): An R package that teaches you R from within R
- [RStudio's](#) : printable cheatsheets
- [Rseek](#) : A custom Google search for R-related sites

2. Getting help

- Google
- Stack Overflow

3. R markdown resources

- Basic markdown and R markdown reference
- [A good markdown reference](#)
- [A good 10-minute markdown tutorial](#)
- [RStudio's R markdown cheatsheet](#)
- [R markdown reference sheet](#)
- [The R markdown documentation](#) including a [getting started guide](#), [a gallery of demos](#), and several [articles](#) for more advanced usage.
- [The knitr website](#) has lots of useful reference material about how knitr works.

4. Git and GitHub resources

- [Happy Git](#): Great resource for using Git and GitHub
- [Version control with RStudio](#): RStudio document for using version control
- [Using Git from RStudio](#): Good 10 minute guide
- [The R Class](#): In depth guide to using Git and GitHub with RStudio

5. R Programming

- [R Programming for Data Science](#): In depth guide to R programming
- [R for Data Science](#): Fantastic book, tidyverse orientated

2.6 Menyimpan pekerjaan di R

Untuk menyimpan sebuah objek ke dalam sebuah file berformat .RData, kita bisa menggunakan `save()` function

`save(nameOfObject, file = "name_of_file.RData")`

```
> save(obj1, file="obj1.RData")
```

 obj1.RData


1/26/2023 12:29 AM

R Workspace

Jika ingin menyimpan semua objek yang ada di workspace menjadi satu file .RData, bisa menggunakan `save.image()`

`save.image(file = "name_of_file.RData")`

```
> save.image(file='latihan.RData')
```

 latihan.RData

1/26/2023 12:30 AM

R Workspace

Untuk meng-load file .RData kembali ke RStudio, bisa menggunakan `load()` function

`load(file = "name_of_file.RData")`

3. Data pada R

3.1 Tipe data

Tipe data dalam bahasa R meliputi numeric, complex, character, dan logical

- Numeric : 8, 9.8, 2.4e-2
- Complex: 3.5 + 2i
- Character: "Metode Kuantitatif"
- Logical : T,F,TRUE,FALSE

Mengecek tipe data suatu objek menggunakan `class()` function

```
> num <- 2.2
> class(num)
[1] "numeric"
> char <- "metkuan"
> class(char)
[1] "character"
> logi <- TRUE
> class(logi)
[1] "logical"
```

Mengecek tipe data dengan *logical test* menggunakan `is.[tipe_class]()` function

```
> is.numeric(num)
[1] TRUE
> is.numeric(char)
[1] FALSE
> is.character(char)
[1] TRUE
> is.logical(logi)
[1] TRUE
```

Mengubah tipe data dapat dilakukan menggunakan `as.[tipe_class]()` function

Mengubah tipe numeric menjadi character

Tabel logical test dan konversi tipe data

Tipe	Logical test	Konversi
------	--------------	----------

Character	is.character	as.character
Numeric	is.numeric	as.numeric
Logical	is.logical	as.logical
Factor	is.factor	as.factor
Complex	is.complex	as.complex

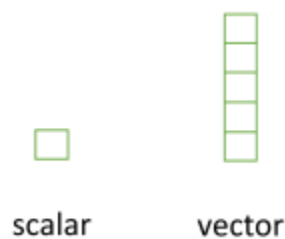
3.2 Struktur data (data structures)

Struktur data dalam bahasa R meliputi vektor, matriks, array, faktor, list, dan data frame.

3.2.1 Skalar dan vektor

Vektor : struktur data sederhana dengan semua elemen memiliki tipe yang sama

Skalar : vektor yang berisi *single value*



#Membuat vektor

```
vec <- c(80, 50, 20, 92)
```

```
> vec <- c(80, 50, 20, 92)
> vec
[1] 80 50 20 92
```

3.2.2 Matriks dan array

Matriks : struktur data dua dimensi terdiri dari baris dan kolom. Matriks dapat tersusun dari gabungan beberapa vektor. Tiap elemen dalam matriks memiliki tipe yang sama.

Array : multidimensional matriks

- Membuat matrix

```
my_mat <- matrix(1:16, nrow = 4, byrow = TRUE)
```

```
> my_mat <- matrix(1:16, nrow = 4, byrow = TRUE)
> my_mat
      [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
[4,]   13   14   15   16
```

- Membuat array

```
my_array <- array(1:16, dim = c(2,4,2))
```

```

> my_array <- array(1:16, dim = c(2,4,2))
> my_array
, , 1
      [,1] [,2] [,3] [,4]
[1,]     1     3     5     7
[2,]     2     4     6     8

, , 2
      [,1] [,2] [,3] [,4]
[1,]     9    11    13    15
[2,]    10    12    14    16

```

Jika ingin mengubah nama baris dan kolom dari matriks dapat menggunakan `rownames()` dan `colnames()` functions

```
rownames(my_mat) <- c('A', 'B', 'C', 'D')
```

```
colnames(my_mat) <- c('X', 'Y', 'W', 'Z')
```

```

> rownames(my_mat) <- c('A', 'B', 'C', 'D')
> colnames(my_mat) <- c('X', 'Y', 'W', 'Z')
> my_mat
  X Y W Z
A 1 2 3 4
B 5 6 7 8
C 9 10 11 12
D 13 14 15 16

```

Selain itu, jika ingin melakukan *transpose* untuk menukar baris dan kolom dapat menggunakan `t()` function

```
My_mat_t <- t(my_mat)
```

```

> My_mat_t <- t(my_mat)
> My_mat_t
  A B C D
X 1 5 9 13
Y 2 6 10 14
W 3 7 11 15
Z 4 8 12 16

```

Untuk meng-ekstrak diagonal matriks, dapat menggunakan `diag()`

```
My_mat_diag <- diag(my_mat)
```

```

> My_mat_diag <- diag(my_mat)
> My_mat_diag
[1] 1 6 11 16

```

Selanjutnya, kita bisa melakukan operasi matriks seperti penjumlahan, pengurangan dan perkalian
#penjumlahan (`mat.1 + mat.2`)

```

> mat.1 <- matrix(c(2, 0, 1, 1), nrow = 2)
> mat.2 <- matrix(c(1, 1, 0, 2), nrow = 2)
> mat.1
      [,1] [,2]
[1,]    2    1
[2,]    0    1
> mat.2
      [,1] [,2]
[1,]    1    0
[2,]    1    2
> mat.1 + mat.2
      [,1] [,2]
[1,]    3    1
[2,]    1    3

```

#pengurangan (*mat.1 - mat.2*)

```

> mat.1 - mat.2
      [,1] [,2]
[1,]    1    1
[2,]   -1   -1

```

#perkalian (*mat.1 %*% mat.2*)

```

> mat.1 %*% mat.2
      [,1] [,2]
[1,]    3    2
[2,]    1    2

```

3.2.3 List

List : struktur data yang elemen didalamnya dapat memiliki tipe yang berbeda. Umumnya digunakan sebagai tempat menyimpan suatu nilai

```

list_1 <- list(c("black", "yellow", "orange"),
              c(TRUE, TRUE, FALSE, TRUE, FALSE, FALSE),
              matrix(1:6, nrow = 3))

```

```

> list_1 <- list(c("black", "yellow", "orange"),
+               c(TRUE, TRUE, FALSE, TRUE, FALSE, FALSE),
+               matrix(1:6, nrow = 3))
> list_1
[[1]]
[1] "black" "yellow" "orange"

[[2]]
[1] TRUE TRUE FALSE TRUE FALSE FALSE

[[3]]
      [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6

```

3.2.4 Data frame

Dataframe : list tipe khusus digunakan untuk menyimpan data dengan tipe yang berbeda dalam bentuk

matriks. Umumnya data tabular memiliki objek Data Frame. Data frame bisa dibayangkan seperti membuat tabel di Excel

```
Tinggi <- c(180, 155, 160, 167, 181)
```

```
Nama <- c('Tony', 'Nate', 'Banner', 'Steve', 'Clint')
```

```
Berat <- c(85, 70, 100, 80, 90)
```

```
df <- data.frame(Tinggi, Berat, Nama)
```

```
> Tinggi <- c(180, 155, 160, 167, 181)
> Nama <- c('Tony', 'Nate', 'Banner', 'Steve', 'Clint')
> Berat <- c(85, 70, 100, 80, 90)
> df <- data.frame(Tinggi, Berat, Nama)
> df
  Tinggi Berat  Nama
1    180    85  Tony
2    155    70  Nate
3    160   100 Banner
4    167    80  Steve
5    181    90  Clint
```

Mengecek dimensi dataframe

```
dim(df)
```

```
> dim(df)
[1] 5 3
```

Melihat ringkasan struktur objek dataframe

```
str(df)
```

```
> str(df)
'data.frame':  5 obs. of  3 variables:
 $ Tinggi: num  180 155 160 167 181
 $ Berat : num   85  70 100  80  90
 $ Nama  : chr  "Tony" "Nate" "Banner" "Steve" ...
```

3.3 Importing data

3.3.1 Saving file untuk di import

Biasanya file yang sering di-import adalah file berformat .csv

Caranya :

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

New File
New Project...
Open File... Ctrl+O
Open File in New Column...
Reopen with Encoding...
Recent Files
Open Project...
Open Project in New Session...
Recent Projects
Import Dataset
Save Ctrl+S
Save As...
Save with Encoding...
Save All Ctrl+Alt+S
Compile Report...
Publish...
Print...
Close Ctrl+W
Close All Ctrl+Shift+W
Close All Except Current Ctrl+Alt+Shift+W
Close Project
Quit Session... Ctrl+Q

From Text (base)...
From Text (readr)... → untuk file format .csv
From Excel... → untuk file format.xlsx
From SPSS...
From SAS...
From Stata...

Import Text Data

File/URL:
Path file\namafile.csv Browse...

Data Preview:

Import Options:

Name: dataset
Skip: 0
☒ First Row as Names
☒ Trim Spaces
☒ Open Data Viewer
Delimiter: Comma
Quotes: Default
Locale: Configure...
Escape: None
Comment: Default
NA: Default

Code Preview:

```
library(readr)  
dataset <- read_csv(NULL)  
view(dataset)
```

→ kode snippet untuk mengimport data

Reading rectangular data using readr

Import Cancel

3.3.2 Import functions

- Import file .txt

```
flowers <- read.table(file = 'data/flower.txt', header = TRUE, sep = "\t", stringsAsFactors = FALSE)
```

```
str(flowers)
## 'data.frame':   96 obs. of  8 variables:
## $ treat   : chr  "tip" "tip" "tip" "tip" ...
## $ nitrogen: chr  "medium" "medium" "medium" "medium" ...
## $ block   : int   1 1 1 1 1 1 1 1 2 2 ...
## $ height  : num   7.5 10.7 11.2 10.4 10.4 9.8 6.9 9.4 10.4 12.3 ...
## $ weight  : num   7.62 12.14 12.76 8.78 13.58 ...
## $ leafarea: num   11.7 14.1 7.1 11.9 14.5 12.2 13.2 14 10.5 16.1 ...
## $ shootarea: num   31.9 46 66.7 20.3 26.9 72.7 43.1 28.5 57.8 36.9 ...
## $ flowers : int    1 10 10 1 4 9 7 6 5 8 ...
```

- Import file .csv

```
flowers <- read.csv(file = 'data/flower.csv', header = FALSE, sep = ',')
```

3.3.3 Pesan error yang sering ditemui

```
Error in file(file, "rt") : cannot open the connection
In addition: Warning message:
In file(file, "rt") :
  cannot open file 'flower.txt': No such file or directory
```

Beberapa penyebab error ketika melakukan importing data adalah kesalahan huruf (typo, case sensitive), salah path file ataupun salah format file (.txt, .csv dll)

3.3.4 Cara lain import data

Ada banyak function lain untuk import data dari berbagai sumber dan format. Sebagian besar function ini terdapat dalam package yang perlu diinstall pada RStudio. Contohnya function `fread()` pada package `read.table` lebih bagus untuk mengimport data yang besar dengan lebih efisien dan cepat daripada function `read.table()`

```
library(read.table)
```

```
all_data <- fread(file = 'data/flower.txt')
```

Kemudian ada package `readr`

```
library(readr)
```

```
# import white space delimited files
```

```
all_data <- read_table(file = 'data/flower.txt', col_names = TRUE)
```

```
# import comma delimited files
```

```
all_data <- read_csv(file = 'data/flower.txt')
```

```
# import tab delimited files
```

```
all_data <- read_delim(file = 'data/flower.txt', delim = "\t")
```

3. 4 Wrangling data frames / Manipulasi data frame

R mampu melakukan manipulasi, summarising dan visualisasi data. Manipulasi data atau sering disebut dengan *wrangling* atau *munging*. Sebagai contoh, kita bisa menggunakan data frame yang sudah di import sebelumnya.

```
flowers <- read.table(file = 'data/flower.txt', header = TRUE, sep = "\t")
str(flowers)
```

```
## 'data.frame':   96 obs. of  8 variables:
## $ treat      : chr  "tip" "tip" "tip" "tip" ...
## $ nitrogen   : chr  "medium" "medium" "medium" "medium" ...
## $ block      : int   1 1 1 1 1 1 1 1 2 2 ...
## $ height     : num   7.5 10.7 11.2 10.4 10.4 9.8 6.9 9.4 10.4 12.3 ...
## $ weight     : num   7.62 12.14 12.76 8.78 13.58 ...
## $ leafarea   : num   11.7 14.1 7.1 11.9 14.5 12.2 13.2 14 10.5 16.1 ...
## $ shootarea  : num   31.9 46 66.7 20.3 26.9 72.7 43.1 28.5 57.8 36.9 ...
## $ flowers    : int    1 10 10 1 4 9 7 6 5 8 ...
```

Kita bisa mengakses kolom data frame, misal kolom 'height'

```
flowers$height
```

```
## [1]  7.5 10.7 11.2 10.4 10.4  9.8  6.9  9.4 10.4 12.3 10.4 11.0  7.1  6.0  9.0
## [16]  4.5 12.6 10.0 10.0  8.5 14.1 10.1  8.5  6.5 11.5  7.7  6.4  8.8  9.2  6.2
## [31]  6.3 17.2  8.0  8.0  6.4  7.6  9.7 12.3  9.1  8.9  7.4  3.1  7.9  8.8  8.5
## [46]  5.6 11.5  5.8  5.6  5.3  7.5  4.1  3.5  8.5  4.9  2.5  5.4  3.9  5.8  4.5
## [61]  8.0  1.8  2.2  3.9  8.5  8.5  6.4  1.2  2.6 10.9  7.2  2.1  4.7  5.0  6.5
## [76]  2.6  6.0  9.3  4.6  5.2  3.9  2.3  5.2  2.2  4.5  1.8  3.0  3.7  2.4  5.7
## [91]  3.7  3.2  3.9  3.3  5.5  4.4
```

Selain itu, kita bisa menyimpan nilai suatu kolom kedalam suatu objek, kemudian menghitung rata-rata atau summary dari kolom tersebut dengan `summary()`

```
f_height <- flowers$height
```

```
mean(f_height)
```

```
summary(f_height)
```

```
f_height <- flowers$height
mean(f_height)
## [1] 6.839583
summary(f_height)
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      1.200  4.475   6.450   6.840   9.025  17.200
```

3.4.1 Positional indexes

Untuk menggunakan positional indexes, kita tinggal menuliskan posisi baris dan kolom yang diinginkan seperti mengakses indeks pada vektor. Misal untuk melihat nilai pada baris ke-1 dan kolom ke-4 :

```
flowers[1, 4]
```

Atau mengaksesnya melalui kolom tertentu

```
flowers$height[1]
```

```
flowers[1, 4]
## [1] 7.5

# this would give you the same
flowers$height[1]
## [1] 7.5
```

Kita juga bisa mengakses baris dan kolom lebih dari 1 dengan cara:
`flowers[1:10, 1:4]`

```
flowers[1:10, 1:4]
##      treat nitrogen block height
## 1    tip    medium     1    7.5
## 2    tip    medium     1   10.7
## 3    tip    medium     1   11.2
## 4    tip    medium     1   10.4
## 5    tip    medium     1   10.4
## 6    tip    medium     1    9.8
## 7    tip    medium     1    6.9
## 8    tip    medium     1    9.4
## 9    tip    medium     2   10.4
## 10   tip    medium     2   12.3
```

Bisa juga menggunakan `c()` function

```
flowers[c(1, 5, 12, 30), c(1, 3, 6, 8)]
##      treat block leafarea flowers
## 1    tip      1      11.7      1
## 5    tip      1      14.5      4
## 12   tip      2      12.6      6
## 30   tip      2      11.6      5
```

Kemudian untuk mengakses baris tertentu dan keseluruhan kolom, dapat dilakukan dengan mengosongkan bagian kolomnya pada `[]`. Begitu juga sebaliknya untuk baris.

`flowers[1:8,]` #Mengakses baris 1-8 pada semua kolom

```
flowers[1:8, ]
##      treat nitrogen block height weight leafarea shootarea flowers
## 1    tip    medium     1    7.5    7.62    11.7    31.9      1
## 2    tip    medium     1   10.7   12.14    14.1    46.0     10
## 3    tip    medium     1   11.2   12.76     7.1    66.7     10
## 4    tip    medium     1   10.4    8.78    11.9    20.3      1
## 5    tip    medium     1   10.4   13.58    14.5    26.9      4
## 6    tip    medium     1    9.8   10.08    12.2    72.7      9
## 7    tip    medium     1    6.9   10.11    13.2    43.1      7
## 8    tip    medium     1    9.4   10.28    14.0    28.5      6
```

`flowers[, 1:3]` #Mengakses semua baris pada kolom 1-3

Selain itu, bisa juga mengakses kolom menggunakan nama dari kolom tersebut

#Mengakses baris 1-5 pada kolom treat, nitrogen dan leafarea

flowers[1:5, c('treat', 'nitrogen', 'leafarea')]

```
flowers[1:5, c("treat", "nitrogen", "leafarea")]
##   treat nitrogen leafarea
## 1   tip   medium    11.7
## 2   tip   medium    14.1
## 3   tip   medium     7.1
## 4   tip   medium    11.9
## 5   tip   medium    14.5
```

3.4.2 Logical indexes

Sama halnya pada vektor, kita bisa juga mengekstrak data dari data frame berdasarkan logical test. Contoh:

#Menampilkan tinggi bunga yang > 12

big_flowers <- flowers[flowers\$height > 12,]

```
big_flowers <- flowers[flowers$height > 12, ]
big_flowers
##   treat nitrogen block height weight leafarea shootarea flowers
## 10   tip   medium     2   12.3  13.48    16.1    36.9      8
## 17   tip    high     1   12.6  18.66    18.6    54.0      9
## 21   tip    high     1   14.1  19.12    13.1   113.2     13
## 32   tip    high     2   17.2  19.20    10.9    89.9     14
## 38   tip     low     1   12.3  11.27    13.7    28.7      5
```

Contoh lain:

```
flowers[flowers$height >= 6, ]      # values greater or equal to 6

flowers[flowers$height <= 6, ]      # values less than or equal to 6

flowers[flowers$height == 8, ]      # values equal to 8

flowers[flowers$height != 8, ]      # values not equal to 8
```

Kemudian, kita juga bisa mengekstrak menggunakan string tertentu dengan operator equals atau '==' maupun not equals atau '!='

nit_high <- flowers[flowers\$nitrogen == "high",]

```
##      treat nitrogen block height weight leafarea shootarea flowers
## 17    tip      high     1   12.6  18.66     18.6    54.0      9
## 18    tip      high     1   10.0  18.07     16.9    90.5      3
## 19    tip      high     1   10.0  13.29     15.8   142.7     12
## 20    tip      high     1    8.5  14.33     13.2    91.4      5
## 21    tip      high     1   14.1  19.12     13.1   113.2     13
## 22    tip      high     1   10.1  15.49     12.6    77.2     12
## 23    tip      high     1    8.5  17.82     20.5    54.4      3
## 24    tip      high     1    6.5  17.13     24.1   147.4      6
```

`nit_not_medium <- flowers[flowers$nitrogen != "medium", 1:4]`

```
nit_not_medium <- flowers[flowers$nitrogen != "medium", 1:4]
nit_not_medium
```

```
##      treat nitrogen block height
## 17    tip      high     1   12.6
## 18    tip      high     1   10.0
## 19    tip      high     1   10.0
## 20    tip      high     1    8.5
## 21    tip      high     1   14.1
## 22    tip      high     1   10.1
```

Bisa juga menggunakan boolean operator seperti AND atau OR

`low_notip_heigh6 <- flowers[flowers$height >= 6 & flowers$nitrogen == "medium" & flowers$treat == "notip",]`

```
low_notip_heigh6 <- flowers[flowers$height >= 6 & flowers$nitrogen == "medium" &
                             flowers$treat == "notip", ]
low_notip_heigh6
##      treat nitrogen block height weight leafarea shootarea flowers
## 51 notip    medium     1    7.5  13.60     13.6   122.2     11
## 54 notip    medium     1    8.5  10.04     12.3   113.6      4
## 61 notip    medium     2    8.0  11.43     12.6    43.2     14
```

`height2.2_12.3 <- flowers[flowers$height > 12.3 | flowers$height < 2.2,]`

```
height2.2_12.3 <- flowers[flowers$height > 12.3 | flowers$height < 2.2, ]
height2.2_12.3
```

##	treat	nitrogen	block	height	weight	leafarea	shootarea	flowers
## 17	tip	high	1	12.6	18.66	18.6	54.0	9
## 21	tip	high	1	14.1	19.12	13.1	113.2	13
## 32	tip	high	2	17.2	19.20	10.9	89.9	14
## 62	notip	medium	2	1.8	10.47	11.8	120.8	9
## 68	notip	high	1	1.2	18.24	16.6	148.1	7
## 72	notip	high	1	2.1	19.15	15.6	176.7	6
## 86	notip	low	1	1.8	6.01	17.6	46.2	4

Sebuah metode alternatif dalam penggunaan logical expression adalah dengan *subset()* function pada []. Keuntungannya adalah kita tidak perlu lagi menggunakan notasi \$ ketika memilih kolom spesifik. Kelemahannya kurang fleksibel daripada notasi [].

`tip_med_2 <- subset(flowers, treat == "tip" & nitrogen == "medium" & block == 2)`

```
tip_med_2 <- subset(flowers, treat == "tip" & nitrogen == "medium" & block == 2)
tip_med_2
```

##	treat	nitrogen	block	height	weight	leafarea	shootarea	flowers
## 9	tip	medium	2	10.4	10.48	10.5	57.8	5
## 10	tip	medium	2	12.3	13.48	16.1	36.9	8
## 11	tip	medium	2	10.4	13.18	11.1	56.8	12
## 12	tip	medium	2	11.0	11.56	12.6	31.3	6
## 13	tip	medium	2	7.1	8.16	29.6	9.7	2
## 14	tip	medium	2	6.0	11.22	13.0	16.4	3
## 15	tip	medium	2	9.0	10.20	10.8	90.1	6
## 16	tip	medium	2	4.5	12.55	13.4	14.4	6

Ketika ingin memilih kolom tertentu bisa menggunakan argumen *select* =

```
tipplants <- subset(flowers, treat == "tip" & nitrogen == "medium" & block == 2,
                    select = c("treat", "nitrogen", "leafarea"))
tipplants
```

##	treat	nitrogen	leafarea
## 9	tip	medium	10.5
## 10	tip	medium	16.1
## 11	tip	medium	11.1
## 12	tip	medium	12.6
## 13	tip	medium	29.6
## 14	tip	medium	13.0
## 15	tip	medium	10.8
## 16	tip	medium	13.4

3.4.3 Ordering data frames

Seperti pada vektor, kita menggunakan function *order()*.

#Mengurutkan berdasarkan tinggi bunga
height_ord <- flowers[order(flowers\$height),]

```
height_ord <- flowers[order(flowers$height), ]
height_ord
```

```
##      treat nitrogen block height weight leafarea shootarea flowers
## 68 notip      high      1   1.2  18.24    16.6   148.1      7
## 62 notip    medium      2   1.8  10.47    11.8   120.8      9
## 86 notip      low       1   1.8   6.01    17.6    46.2      4
## 72 notip      high      1   2.1  19.15    15.6   176.7      6
## 63 notip    medium      2   2.2  10.70    15.3    97.1      7
## 84 notip      low       1   2.2   9.97     9.6    63.1      2
## 82 notip      low       1   2.3   7.28    13.8    32.8      6
```

Kita juga bisa menambahkan tipe orderingnya secara descending menggunakan argumen `decreasing = TRUE`
leafarea_ord <- flowers[order(flowers\$leafarea, decreasing = TRUE),]

```
leafarea_ord <- flowers[order(flowers$leafarea, decreasing = TRUE), ]
leafarea_ord
```

```
##      treat nitrogen block height weight leafarea shootarea flowers
## 70 notip      high      1  10.9  17.22    49.2   189.6     17
## 13  tip      medium      2   7.1   8.16    29.6     9.7      2
## 24  tip      high       1   6.5  17.13    24.1   147.4      6
## 65 notip      high      1   8.5  22.53    20.8   166.9     16
## 23  tip      high       1   8.5  17.82    20.5    54.4      3
## 66 notip      high      1   8.5  17.33    19.8   184.4     12
```

Selain itu, bisa juga orderingnya berdasarkan lebih dari satu kolom
block_height_ord <- flowers[order(flowers\$block, flowers\$height),]

```
block_height_ord <- flowers[order(flowers$block, flowers$height), ]
block_height_ord
```

```
##      treat nitrogen block height weight leafarea shootarea flowers
## 68 notip      high      1   1.2  18.24    16.6   148.1      7
## 86 notip      low       1   1.8   6.01    17.6    46.2      4
## 72 notip      high      1   2.1  19.15    15.6   176.7      6
## 84 notip      low       1   2.2   9.97     9.6    63.1      2
## 82 notip      low       1   2.3   7.28    13.8    32.8      6
## 56 notip    medium      1   2.5  14.85    17.5    77.8     10
```

Bagaimana jika kita ingin mengurutkan ascending untuk kolom 'block' tapi descending untuk kolom 'height' ?

Kita bisa menambahkan simbol ' - ' ketika menggunakan `order()`

`block_revheight_ord <- flowers[order(flowers$block, -flowers$height),]`

```
block_revheight_ord <- flowers[order(flowers$block, -flowers$height), ]
block_revheight_ord
```

```
##      treat nitrogen block height weight leafarea shootarea flowers
## 21    tip      high     1   14.1  19.12    13.1    113.2     13
## 17    tip      high     1   12.6  18.66    18.6     54.0      9
## 38    tip      low      1   12.3  11.27    13.7     28.7      5
## 3     tip     medium     1   11.2  12.76     7.1     66.7     10
## 70 notip      high     1   10.9  17.22    49.2    189.6     17
```

Jika kita ingin mengurutkan kolom nitrogen dengan urutan low -> medium -> high, dapat menggunakan `factor()` function dan kemudian menggunakan `order()`

`flowers$nitrogen <- factor(flowers$nitrogen, levels = c("low", "medium", "high")) nit_ord <- flowers[order(flowers$nitrogen),]`

3.4.4 Menambahkan baris dan kolom

```
flowers$nitrogen <- factor(flowers$nitrogen,
                           levels = c("low", "medium", "high"))
nit_ord <- flowers[order(flowers$nitrogen),]
nit_ord
```

```
##      treat nitrogen block height weight leafarea shootarea flowers
## 33    tip      low      1    8.0   6.88     9.3     16.1      4
## 34    tip      low      1    8.0  10.23    11.9     88.1      4
## 35    tip      low      1    6.4   5.97     8.7      7.3      2
## 36    tip      low      1    7.6  13.05     7.2     47.2      8
## 37    tip      low      1    9.7   6.49     8.1     18.0      3
```

Penambahan baris menggunakan `rbind()` dan kolom menggunakan `cbind()`

`df_rcomb <- rbind(df1, df2)`

`df_ccomb <- cbind(df3, df4)`

```
df_rcomb <- rbind(df1, df2)
```

```
df_rcomb
```

```
##      id height weight
## 1  1    120     44
## 2  2    150     56
## 3  3    132     49
## 4  4    122     45
## 5  5    119     39
## 6  6    110     35
```

```
df_ccomb <- cbind(df3, df4)
```

```
df_ccomb
```

```
##      id height weight location
## 1  1    120     44      UK
## 2  2    150     56      CZ
## 3  3    132     49      CZ
## 4  4    122     45      UK
```


3.4.5 Merging data frames / menggabungkan data frame

Penggabungan dapat menggunakan `merge()` function

```
newdf <- merge(df1, df2)
```

Jika kita ingin memasukan semua data dari kedua dataframes, maka menambahkan argumen `all = TRUE`

```
newdf <- merge(df1, df2, all = TRUE)
```

3.5 Summarising data frames

Kita dapat mengekstrak ringkasan dari data frame kita dengan function `summary()`

```
summary(flowers)
```

```
summary(flowers)
##      treat      nitrogen      block      height      weight
## Length:96      low  :32   Min.   :1.0   Min.   : 1.200   Min.   : 5.790
## Class :character medium:32   1st Qu.:1.0   1st Qu.: 4.475   1st Qu.: 9.027
## Mode  :character high  :32   Median :1.5   Median : 6.450   Median :11.395
##                                     Mean   :1.5   Mean   : 6.840   Mean   :12.155
##                                     3rd Qu.:2.0   3rd Qu.: 9.025   3rd Qu.:14.537
##                                     Max.   :2.0   Max.   :17.200   Max.   :23.890
##      leafarea      shootarea      flowers
## Min.   : 5.80   Min.   : 5.80   Min.   : 1.000
## 1st Qu.:11.07   1st Qu.: 39.05   1st Qu.: 4.000
## Median :13.45   Median : 70.05   Median : 6.000
## Mean   :14.05   Mean   : 79.78   Mean   : 7.062
```

Atau bisa menggunakan indeks seperti:

```
summary(flowers[, 4:7])
```

```
summary(flowers[, 4:7])
##      height      weight      leafarea      shootarea
## Min.   : 1.200   Min.   : 5.790   Min.   : 5.80   Min.   : 5.80
## 1st Qu.: 4.475   1st Qu.: 9.027   1st Qu.:11.07   1st Qu.: 39.05
## Median : 6.450   Median :11.395   Median :13.45   Median : 70.05
## Mean   : 6.840   Mean   :12.155   Mean   :14.05   Mean   : 79.78
## 3rd Qu.: 9.025   3rd Qu.:14.537   3rd Qu.:16.45   3rd Qu.:113.28
## Max.   :17.200   Max.   :23.890   Max.   :49.20   Max.   :189.60
```

Kemudian untuk summarise 1 kolom saja bisa dengan cara:

```
summary(flowers$leafarea)
```

```
summary(flowers$leafarea)
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      5.80  11.07   13.45   14.05  16.45   49.20
```

Selain itu, kita bisa menghitung berapa banyak nilai di kolom tertentu

```
table(flowers$nitrogen) atau table(flowers$nitrogen, flowers$treat)
```

```
table(flowers$nitrogen)

##
##      low medium   high
##      32      32      32
```

```
table(flowers$nitrogen, flowers$treat)

##
##      notip tip
##      low    16  16
##      medium 16  16
##      high   16  16
```

3.6 Exporting data

3.6.1 Export functions

Data frame yang telah dibuat dapat di export dengan `write.table()` function

#Format file .txt

```
write.table(flowers_df2, file = 'data/flowers_04_12.txt', col.names = TRUE, row.names = FALSE, sep = "\t")
```

#Format file .csv

```
write.table(flowers_df2, file = 'data/flowers_04_12.csv', col.names = TRUE, row.names = FALSE, sep = ",")
```

atau

```
write.csv(flowers_df2, file = 'data/flowers_04_12.csv', row.names = FALSE)
```

3.6.2 Export function lainnya

Function export lainnya adalah `fwrite()` pada package `read.table` atau package `readr`

#Package read.table

#Format file .txt

```
library(read.table)
```

```
fwrite(flowers_df2, file = 'data/flowers_04_12.txt', sep = "\t")
```

#Format file .csv

```
library(read.table)
```

```
fwrite(flowers_df2, file = 'data/flowers_04_12.csv')
```

#Package readr

#Format file .txt

```
library(readr)
```

```
write_tsv(flowers_df2, path = 'data/flowers_04_12.txt')
```

#Format file .csv

```
write_csv(flowers_df2, path = 'data/flowers_04_12.csv')
```