1-Pengenalan R

Ashari Ramadhan

11/19/2020

1. Pengenalan

Pengenalan R

R merupakan bahasa yang digunakan dalam komputasi statistik.

Figure 1: Logo R

R bisa digunakan untuk membuat model linier dan nonlinier, hipotesis test, visualisasi, time series, klasifikasi, web-apps, pemetaan dan lain-lain.

Kenapa harus pakai R?

1. Gratis / Open Source
2. Packages / Library yang banyak
3. Digunakan oleh industri
4. Komunitas Besar

Case sensitif

Pada bahasa R, huruf kapital dan non kapital dianggap berbeda. Contoh angka berbeda dengan Angka berbeda pula dengan ANGKA

```r
"angka" == "Angka"
```

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

```r
"angka" == "angka"
```

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

Hasil FALSE menandakan objek berbeda, sedangkan TRUE menandakan objek sama

Komentar

Komentar digunakan untuk memberikan penjelasan pada program. Komentar tidak akan mempengaruhi jalannya program. Pada bahasa R semua text yang berada di di belakang # akan dianggap komentar dan tidak akan dieksukusi

```r
# ini adalah komentar
# 1 + 1
```

Kode diatas tidak menghasilkan output.

```r
# ini adalah komentar
1 + 1
```

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

baris 1 + 1 dieksekusi dan menghasilkan output 2

help

Setiap fungsi di R memiliki dokumentasi yang berisi cara penulisan, parameter dan penjelasan-penjelasan lainnya. misal mean adalah fungsi untuk mencari nilai rata-rata. Jika ingin menampilkan dokumentasi mean ketikkan

```r
help("mean")
```

atau

```r
?mean
```

Say Hello World

```r
print('Hello World')
```

```
## [1] "Hello World"
```

2. Operator pada R

Operator aritmatika

- tanda + adalah penjumlahan
- tanda - adalah pengurangan
- tanda / adalah pembagian
- tanda * perkalian
- tanda ^ pangkat
- tanda %% modulos, sisa bagi

```
print('1 + 1 sama dengan')
```

```
## [1] "1 + 1 sama dengan"
```

```
1 + 1
```

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

```
print('2 - 3 sama dengan')
```

```
## [1] "2 - 3 sama dengan"
```

```
2 - 3
```

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

```
print('8/3 sama dengan')
```

```
## [1] "8/3 sama dengan"
```

```
8/3
```

```
## [1] 2.666667
```

```
print('9 * 0.5 sama dengan')
```

```
## [1] "9 * 0.5 sama dengan"
```

```
9*0.5
```

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

```
print('6 ^ 9 sama dengan')
```

```
## [1] "6 ^ 9 sama dengan"
```

```
6^9
```

```
## [1] 10077696
```

```
print('9 %% 3')
```

```
## [1] "9 %% 3"
```

```
9%%3
```

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

tambahan operator

```
numbers = c(1:7)
numbers
```

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

3. Variabel

Variabel tempat untuk menyimpan sebuah nilai

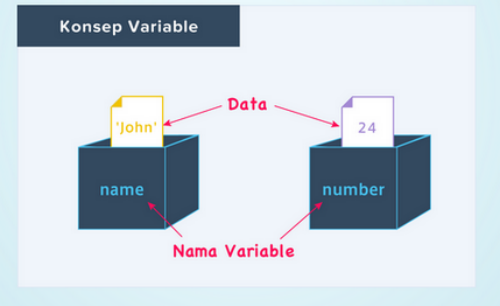


Figure 2: Konsep Variabel

Membuat variabel kita memerlukan operator assigmnent

```
number <- 24

print(number)
```

```
## [1] 24
```

```
print(number + 2)
```

```
## [1] 26
```

```r
name <- 'John'

print(paste('Nama saya', name, 'umur saya', number - 2, 'tahun'))
```

```
## [1] "Nama saya John umur saya 22 tahun"
```

Operator assigment

```r
name1 <- 'John' # tapi ini yang disarankan untuk di pake, RStyle
name2 = 'John'
name3 <<- 'John'
'John' -> name4
```

Penamaaan Variabel

- Diberi nama yang jelas, sesuai dengan isinya.

Sebagai contoh akan lebih mudah dibaca jika menulis:

umurku = 22

dibanding

x = 22

Meskipun maksud x adalah umurku

Penamaaan Variabel

- Tidak diawali angka atau _

```r
_x = 3
```

```
Error: unexpected input in "_"
```

Penamaan variabel

- jangan menggunakan tanda operasi untuk penghubungan atau tanda spasi. Sebagai pengganti gunakan tanda _ sebagai penghubungan
- disarankan menggunakan huruf kecil, menggunakan Snake case (stylized as snake_case)

```r
umur-ku = 20
Error in umur - ku = 20 : object 'umur' not found
```

Memasukkan banyak nilai pada variabel

Tentu tidak efetif jika menulis kode secara berulang seperti ini

```r
food1 <- 'banana'
food2 <- 'orange'
food3 <- 'grape'

print(food1)
```

```
## [1] "banana"
```

```
print(food2)
```

```
## [1] "orange"
```

```
print(food3)
```

```
## [1] "grape"
```

Lebih baik di tulis seperti ini

```
# lebih baik di tulis seperti ini

foods = c('banana', 'orange', 'grape')
print(foods)
```

```
## [1] "banana" "orange" "grape"
```

4. Percabangan

Percabangan adalah fitur dari bahasa pemrograman yang melakukan perhitungan atau tindakan yang berbeda tergantung pada apakah kondisi boolean yang ditentukan pemrogram mengevaluasi benar atau salah.

Kondisi boolean:

- TRUE, kondisi benar
- FALSE, kondisi salah

```
x = 4
y = 9

x < y
```

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

```
y <= x
```

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

```
x > y
```

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

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x | y	x OR y
x & y	x AND y
isTRUE(x)	test if X is TRUE

Figure 3: Operator Logika

```
y >= x
```

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

```
x == y
```

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

```
y != x
```

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

Percabangan

Percabangan adalah cara yang digunakan untuk mengambil keputusan apabila di dalam program dihadapkan pada kondisi tertentu. Contoh kita ingin mencetak 'Anda lulus ujian' jika nilai yang di peroleh >= 70

```
if ( __kondisi__ ){
    __statment__
}
```

Logika nya statement akan di eksekusi jika berniali TRUE (benar)

IF

```
nilai <- 70
if(nilai > 70){
    print('Anda lulus ujian')
}
```

if else

```
nilai <- 70
if(nilai > 70){
    print('Anda lulus ujian')
} else{
  print('anda tidak lulus ujian')
}
```

```
## [1] "anda tidak lulus ujian"
```

if elseif else

```
nilai <- 70
if(nilai > 70){
    print('anda lulus ujian')
} else if( nilai == 70){
    print('anda tidak lulus ujian, tapi boleh remedian')
} else{
    print('anda tidak lulus ujian')
}
```

```
## [1] "anda tidak lulus ujian, tapi boleh remedian"
```

Latihan

Buat program percabangan dengan kondisi

- jika nilai > 80 cetak "A"
- jika 70 < nilai <= 80 cetak "B"
- jika nilai 50 < nilai <=70 cetak "C"
- Selain itu cetak "D"

5. Perulangan

Perulangan atau yang sering disebut dengan "looping", merupakan proses yang dilakukan secara berulang-ulang dalam batas yang telah ditentukan.

Contoh jika kita ingin membuat tabel perkalian 9.

Cara manual

```
print(paste('9*1 =', 9*1))
```

```
## [1] "9*1 = 9"
```

```
print(paste('9*2 =', 9*2))
```

```
## [1] "9*2 = 18"
```

```
print(paste('9*3 =', 9*3))
```

```
## [1] "9*3 = 27"
```

```
print(paste('9*4 =', 9*4))
```

```
## [1] "9*4 = 36"
```

```
print(paste('9*5 =', 9*5))
```

```
## [1] "9*5 = 45"
```

```
print(paste('9*6 =', 9*6))
```

```
## [1] "9*6 = 54"
```

```
print(paste('9*7 =', 9*7))
```

```
## [1] "9*7 = 63"
```

```
print(paste('9*8 =', 9*8))
```

```
## [1] "9*8 = 72"
```

```
print(paste('9*9 =', 9*9))
```

```
## [1] "9*9 = 81"
```

```
print(paste('9*10 =', 9*10))
```

```
## [1] "9*10 = 90"
```

Cara looping

```
numbers <- c(1:10)

for(i in numbers){
    print(paste0('9*', i, ' = ', 9*i))
}
```

```
## [1] "9*1 = 9"
## [1] "9*2 = 18"
## [1] "9*3 = 27"
## [1] "9*4 = 36"
## [1] "9*5 = 45"
## [1] "9*6 = 54"
## [1] "9*7 = 63"
## [1] "9*8 = 72"
## [1] "9*9 = 81"
## [1] "9*10 = 90"
```

Contoh lain

```
numbers = c(1:10)
perkalian = c()
perkalian
```

```
## NULL
```

Mengisi variabel perkalian

```
for(i in numbers){
    perkalian[i] = 9*i
}
perkalian
```

```
##  [1]  9 18 27 36 45 54 63 72 81 90
```

Perulangan dengan break

Proses looping akan berhenti jika i sama dengan 5

```
numbers = c(1:10)
perkalian = c()
for(i in numbers){
  perkalian[i] = 9*i
  if(i == 5){
    break
  }
}
perkalian
```

```
## [1]  9 18 27 36 45
```

Perulangan dengan

Proses looping akan di next, di lewati pada kondisi i sama dengan 5

```
numbers = c(1:10)
perkalian = c()
for(i in numbers){
  if(i == 5){
    next
  }
  perkalian[i] = 9*i
}
perkalian
```

```
##  [1]  9 18 27 36 NA 54 63 72 81 90
```

Membuat program angka ganjil-genap

```
numbers <- c(1:10)
for(i in numbers){
    if(i %% 2 == 0){
```

```
        print(paste(i, 'adalah bilangan genap'))
    } else{
        print(paste(i, 'adalah bilangan ganjil'))
    }
}
```

```
## [1] "1 adalah bilangan ganjil"
## [1] "2 adalah bilangan genap"
## [1] "3 adalah bilangan ganjil"
## [1] "4 adalah bilangan genap"
## [1] "5 adalah bilangan ganjil"
## [1] "6 adalah bilangan genap"
## [1] "7 adalah bilangan ganjil"
## [1] "8 adalah bilangan genap"
## [1] "9 adalah bilangan ganjil"
## [1] "10 adalah bilangan genap"
```

6. Fungsi

Fungsi bahasa pemrograman yaitu memerintah komputer untuk mengolah data sesuai dengan alur berpikir yang kita inginkan.

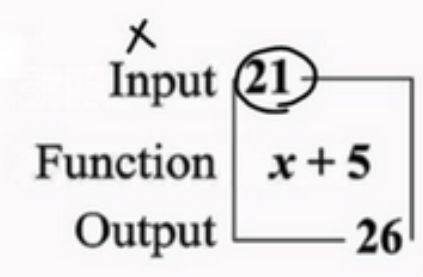


Figure 4: Konsep Fungsi

Penulisan fungsi

cari 1

```
seper_x <- function(x) 1/x
```

cara 2

```
seper_x <- function(x){
    1/x
}
```

Fungsi tanpa parameter

```
hay <- function(){
  print('hello')
}
```

Fungsi dengan parameter

```
luas_segitiga <- function(alas, tinggi){
  1/2 * alas * tinggi
}

luas_segitiga(5, 9)
```

```
## [1] 22.5
```

Funsgi dengan parameter yang di inisialisasi

```
luas_segitiga <- function(alas = 2, tinggi = 4){
  1/2 * alas * tinggi
}
luas_segitiga()
```

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

7.Packages

Packages pada dasarnya adalah fungsi yang sudah di program oleh orang lain dan kita gunakan.

Untuk menggunakan packages kita perli menginstallnya terlebih dahulu

```
install.packages('nama_packages')
```

Contoh menginstall packages ggplot2

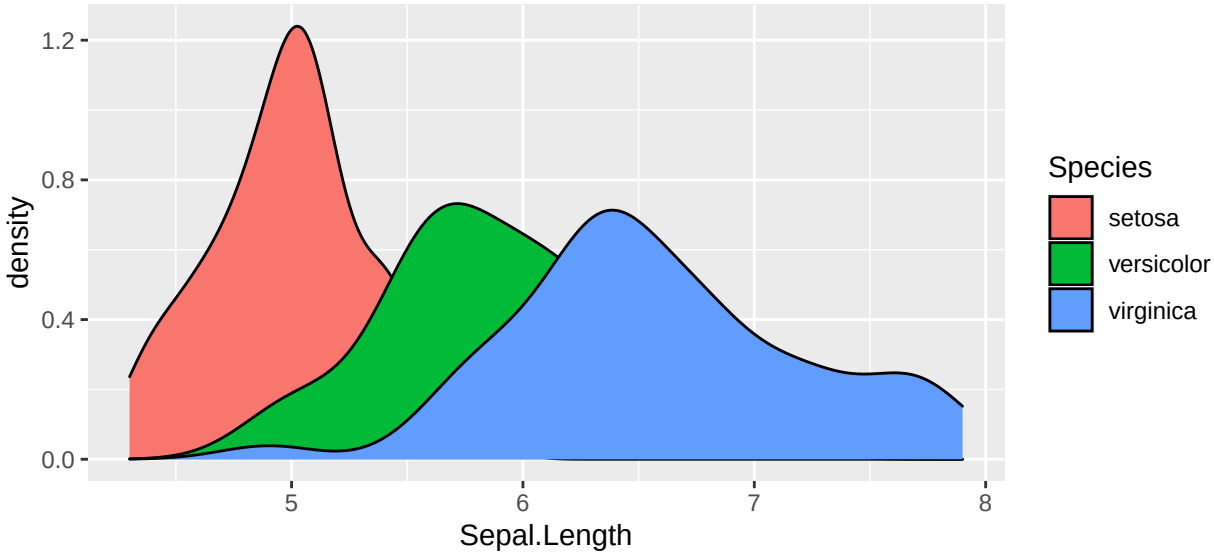
```
install.packages('ggplot2')
```

Untuk load packages agar bisa digunakan tuliskan kode berikut

```
library(ggplot2)
```

Maka packages siap digunakan

```
ggplot(iris, aes(Sepal.Length, fill = Species)) +
  geom_density()
```


1.2
0.8
0.4
0.0
density
5
6
7
8
Sepal.Length
Species
setosa
versicolor
virginica