ALGORITMA DAN PEMROGRAMAN

Fungsi

Dr. Acep Purqon, M.Si
Tirta Setiawan, M.Si
Febri Dwi Irawati, S.Si., M.Si

RA

RB

Fungsi

- Sebuah function dapat dikenali dengan adanya tanda kurung buka dan kurung tutup “()”.
- Function dapat diartikan sebagai suatu objects yang berisi kumpulan perintah. Ketika dipangil aka R akan mengeksekusi kode-kode program yang sudah dideklarasikan dalam function tersebut.
- R menyediakan cukup banyak function siap pakai. Sebagiann besar function suddah tersedia saat R diinstal. Sebagian lagi harus diinstal via Internet.
- Function (dalam matematika) → pemetaan setiap anggota sebuah himpunan dinamakan sebagai domain) kepada tepat sebuah anggota himpunan yang lain (dinamakan kodomain).
- Sebuah function dapat menerimpa input, memproses input, dan kemudian menghasilkan output.
- Function (dalam pemrograman) → sekumpulan instruksi atau statement (pernyataan) yang disusun bersama untuk melakukkan tugas tertentu.
- Function (dalam R) dapat menerima satu atau beberapa input yang disebut parameter, daapt menghasilkan output berupa satu atau beberapa buah value (nilai).
- Pada R terdapat 2 jenis fungsi, yaitu: *build in fuction* dan *user define function*. *build in function* merupakan fungsi bawaan R saat pertama kita menginstall R. Contohnya adalah `mean()`, `sum()`, `ls()`, `rm()`, dll. Sedangkan *user define fuction* merupakan fungsi-fungsi yang dibuat sendiri oleh pengguna.

Membuat function dalam R

- Function → object
- Function akan disimpan pada R environment dan sewaktu-waktu dapat digunakan dengan cara memanggil nama functionnya.
- Membuat R function dapat dilakukan dengan pola sebagai berikut. Function buatan sendiri disebut UDF (User Define Functions)
- Fungsi-fungsi buatan pengguna haruslah dideklarasikan (dibuat) terlebih dahulu sebelum dapat dijalankan. Pola pembentukan fungsi adalah sebagai berikut:

```
function_name <- function(argument_1, argument_2, ...){
  function body
}
```

Catatan:

`function_name` : Nama dari fungsi R. R akan menyimpan fungsi tersebut sebagai objek

`argument_1, argument_2, …` : Argument bersifat opsional (tidak wajib). Argument dapat digunakan untuk memberi inputan kepada fungsi

`function body` : Merupakan inti dari fungsi. Fuction body dapat terdiri atas 0 statement (kosong) hingga banyak statement.

`return` : Fungsi ada yang memiliki output atau return value ada juga yang tidak. Jika fungsi memiliki return value maka return value dapat diproses lebih lanjut

Membuat function dalam R

- Berikut adalah contoh penerapan user define function:

```
# Fungsi tanpa argument
bilang <- function(){
  print("Hello World!!")
}

# Print
bilang()
```

```
HitungLuas<-function(sisi)
{
sisi*sisi
}
```

```
# Fungsi dengan argumen
tambah <- function(a,b){
  print(a+b)
}

# Print
tambah(5,3)
```

```
# Fungsi dengan return value
kali <- function(a,b){
  return(a*b)
}

# Print
kali(4,3)
```

Built-in vs UDF

- R dapat menggabungkan built-n fucction dengan UDF.
- Misalkan ingin membuat function yang dapat menghitung panjang sisi miring pada sebuah segitiga siku-siku (salah satu sudutnya 90 derajat).
- Perhitungan akar kuadrat dapat diselesaikan oleh built-in function bernama `sqrt()`

```
HitungSisiMiring <- function(a,b)
{
sqrt(a*a+b*b)
}
```

Function di dalam Function

- Function dapat dimodifikasi sesuai kebutuhan.
- Contoh, untuk menghitunng kuadrat dari masing-masing sisi dapat digunakan function HitungLuas() yan sudah dibuat sebelumnya.
- Kita boleh memanggil function di dalam function.

```
HitungSisiMiring <-function(a,b)
{
sqrt(HitungLuas(a)+HitungLuas(b))
}
> HitungSisiMiring1(6,8)
```

- Sebuah function lazimnya akan menyertakan built-in function bernama return(). Output dari function ditangani oleh return() ini. Keberadaan return() juga memudahkan untuk memahami alur program dan apa yang hendak dihasilkan oleh program tersebut.

```
fahrenheit_to_kelvin1 <- function(tempt_F){
temp_K <- ((tempt_F-32))*(5/9))+273.15
}
>fahrenheit_to_kelvin1(212)
```

Function di dalam Function

```
fahrenheit_to_kelvin1 <- function(tempt_F){
temp_K <- ((tempt_F-32))*(5/9))+273.15
return(tempt-K)
}
>fahrenheit_to_kelvin1(212)
```

- Kedua program di atas menghasilkan output yang sama, yang membedakan adalah salah satunya menggunakan return().
- Kita tidak harus mencantumkan return() pada functon, karena R akan mengatasinya secara oromatis. Namun akan lebih baik lagi jika return() selalu dicantumkan. Khususnya pada program yang rumit.

```
recur_factorial<- function(n) {
if(n<=1){
return(1)
}else{
return(n*recur_factorial(n-1))
}
}
```

Debugging

- Sering kali fungsi atau sintaks yang kita tulis menghasilkan error sehingga output yang kita harapkan tidak terjadi. Debugging merupakan langkah untuk mengecek error yang terjadi. Untuk lebih memahami proses debugging, berikut contoh error pada suatu fungsi dapat terjadi:

```
f1 <- function(x){
  xsq <- x^2
  xsqminus4 <- xsq - 4
  print(xsqminus4)
  log(xsqminus4-4)
}
f1(6:1)
## [1] 32 21 12  5  0 -3
## Warning in log(xsqminus4 - 4): NaNs produced
## [1] 3.332 2.833 2.079 0.000   NaN   NaN
```

Untuk mengecek error yang terjadi dari sintaks tersebut, kita dapat menggunakan fungsi debug(). Tinggal memasukkan nama fungsi kedalam fungsi debug(). Fungsi tersebut akan secara otomatis akan menampilkan hasil samping dari pengaplikasian fungsi f1() untuk melihat sumber atau tahapan dimana error mulai muncul.

Debugging

```
debug(f1)
f1(1:6)
## debugging in: f1(1:6)
## debug at <text>#1: {
##     xsq <- x^2
##     xsqminus4 <- xsq - 4
##     print(xsqminus4)
##     log(xsqminus4 - 4)
## }
## debug at <text>#2: xsq <- x^2
## debug at <text>#3: xsqminus4 <- xsq - 4
## debug at <text>#4: print(xsqminus4)
## [1] -3  0  5 12 21 32
## debug at <text>#5: log(xsqminus4 - 4)
## Warning in log(xsqminus4 - 4): NaNs produced
## exiting from: f1(1:6)
## [1]   NaN   NaN 0.000 2.079 2.833 3.332
```

Berdasarkan hasil debugging, NaN (missing value) muncul pada tahapan debug ke-4 (kita dapat melakukan enter terus menerus sampai proses debug selesai). Hal ini disebabkan karena terdapat nilai negatif pada objek xsqminu4-4 yang selanjutnya dilakukan transformasi logaritmik. Untuk menghentikan proses debugging pembaca dapat mengetikkan undebug(f1).

Menggunakan Apply Family Function

Berikut adalah sebuah sintaks yang digunakan untuk menghitung nilai mean pada suatu dataset:

```
# subset data iris
sub_iris <- iris[,-5]
# membuat vektor untuk menyimpan hasil loop
a <- rep(NA,4)
# loop
for(i in 1:length(sub_iris)){
  a[i]<-mean(sub_iris[,i])
}
# print
a
## [1] 5.843 3.057 3.758 1.199
class(a) # cek kelas objek
## [1] "numeric"
```

Metode alternatif lain untuk melakukan loop suatu fungsi adalah dengan menggunakan Apply function family. Metode ini memungkinkan untuk melakukan loop suatu fungsi tanpa perlu menuliskan sintaks loop.

- `apply()` : fungsi generik yang mengaplikasikan fungsi kepada kolom atau baris pada matriks atau secara lebih general aplikasi dilakukan pada dimensi untuk jenis data array.
- `lapply()` : fungsi apply yang bekerja pada jenis data list dan memberikan output berupa list juga.
- `sapply()` : bentuk sederhana dari lapply yang menghasilkan output berupa matriks atau vektor.
- `vapply()` : disebut juga verified apply (memungkinkan untuk menghasilkan output dengan jenis data yang telah ditentukan sebelumnya).
- `tapply()` : tagged apply dimana dimana tag menentukan subset dari data.

Apply

Fungsi apply() bekerja dengan jenis data matrik atau array (jenis data homogen). Kita dapat melakukan spesifikasi apakah suatu fungsi hanya akan bekerja pada kolom saja, baris saja atau keduanya. Format fungsi ini adalah sebagai berikut:

```
apply(X, MARGIN, FUN, ...)
```

Catatan:

X: matriks atau array
MARGIN: menentukan bagaimana fungsi bekerja terhadap matriks atau array. Jika nilai yang diinputkan 1, maka fungsi akan bekerja pada masing-masing baris pada matriks. Jika nilainya 2, maka fungsi akan bekerja pada tiap kolom pada matriks.
FUN: fungsi yang akan digunakan. Fungsi yang dapat digunakan dapat berupa fungsi dasar matematika atau statistika, serta user define function.
…: opsional argumen pada fungsi yang digunakan.

```
## membuat matriks
x <- cbind(x1 = 3, x2 = c(4:1, 2:5))
x # print
class(x) # cek kelas objek
## menghitung mean masing-masing kolom
apply(x, MARGIN=2 ,FUN=mean, trim=0.2, na.rm=TRUE)
```

Apply

```
## menghitung range pada masing-masing baris
## menggunakan user define function
apply(x, MARGIN=1,
      FUN=function(x){
        max(x)-min(x)
      })
```

lapply

Fungsi ini melakukan loop fungsi terhadap input data berupa list. Output yang dihasilkan juga merupakan list dengan panjang list yang sama dengan yang diinputkan. Format yang digunakan adalah sebagai berikut:

```
lapply(X, FUN, ...)
```

Catatan:
X: vektor, data frame atau list
FUN: fungsi yang akan digunakan. Fungsi yang dapat digunakan dapat berupa fungsi dasar matematika atau statistika, serta user define function. Subset juga dimungkinkan pada fungsi ini.
…: opsional argumen pada fungsi yang digunakan.

```
## Membuat list
x <- list(a = 1:10, beta = exp(-3:3), logic = c(TRUE,FALSE,FALSE,TRUE))
x # print
class(x) # cek kelas objek
## Menghitung nilai mean pada masing-masing baris lits
lapply(x, FUN=mean)
## Menghitung mean tiap kolom dataset iris
lapply(iris, FUN=mean)
## Mengalikan elemen vektor dengan suatu nilai
y <- c(1:5)
lapply(y, FUN=function(x){x*5})
## Mengubah output menjadi vektor
unlist(lapply(y, FUN=function(x){x*5}))
```

sapply

Fungsi sapply() merupakan bentuk lain dari fungsi lapply(). Perbedaanya terletak pada output default yang dihasilkan. Secara default sapply() menerima input utama berupa list (dapat pula dataframe atau vektor), namun tidak seperti lapply() jenis data output yang dihasilkan adalah vektor. Untuk mengubah output menjadi list perlu argumen tambahan berupa simplify=FALSE. Format fungsi tersebut adalah sebagai berikut:

```
sapply(X, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)
```

Catatan:
X: vektor, data frame atau list
FUN: fungsi yang akan digunakan. Fungsi yang dapat digunakan dapat berupa fungsi dasar matematika atau statistika, serta user define function. Subset juga dimungkinkan pada fungsi ini.
…: opsional argumen pada fungsi yang digunakan.
simplify: logical. Jika nilainya TRUE maka output yang dihasilkan adalah bentuk sederhana dari vektor, matrix atau array.
USE.NAMES: jika list memiliki nama pada setiap elemennya, maka nama elemen tersebut akan secara default ditampilkan.

```
## membuat list
x <- list(a = 1:10, beta = exp(-3:3), logic = c(TRUE,FALSE,FALSE,TRUE))
## menghitung nilai mean setiap elemen
sapply(x, FUN=mean)
## menghitung nilai mean dengan output list
sapply(x, FUN=mean, simplify=FALSE)
## summary objek dataframe
sapply(mtcars, FUN=summary)
```

```
## summary objek list
a <- list(mobil=mtcars,
anggrek=iris)
sapply(a, FUN=summary)
```

vapply

Fungsi ini merupakan bentuk lain dari sapply(). Bedanya secara kecepatan proses fungsi ini lebih cepat dari sapply(). Hal yang menarik dari fungsi ini kita dapat menambahkan argumen FUN.VALUE. pada argumen ini kita memasukkan vektor berupa output fungsi yang diinginkan. Perbedaan lainnya adalah output yang dihasilkan hanya berupa matriks atau array. Format dari fungsi ini adalah sebagai berikut:

```
vapply(X, FUN, FUN.VALUE, ..., USE.NAMES = TRUE)
```

Catatan:
X: vektor, data frame atau list
FUN: fungsi yang akan digunakan. Fungsi yang dapat digunakan dapat berupa fungsi dasar matematika atau statistika, serta user define function. Subset juga dimungkinkan pada fungsi ini.
FUN.VALUE: vektor, template dari return value FUN.
…: opsional argumen pada fungsi yang digunakan.
USE.NAMES: jika list memiliki nama pada setiap elemennya, maka nama elemen tersebut akan secara default ditampilkan.

```
## membuat list
x <- sapply(3:9, seq)
x # print
## membuat ringkasan data pada tiap elemen list
vapply(x, fivenum,
       c(Min. = 0, "1st Qu." = 0,
         Median = 0, "3rd Qu." = 0, Max. = 0))
```

```
## membuat ringkasan data pada tiap kolom
dataframe
vapply(mtcars, summary,
       c(Min. = 0, "1st Qu." = 0,
         Median = 0, "3rd Qu." = 0, Max. = 0,
Mean=0))
```

tapply

Fungsi ini sangat berguna jika pembaca ingin menghitung suatu nilai misalnya mean berdasarkan grup data atau factor.
Format fungsi ini adalah sebagi berikut:

```
tapply(X, INDEX, FUN = NULL, ..., simplify = TRUE)
```

Catatan:
X: vektor, data frame atau list
INDEX: list satu atau beberapa factor yang memiliki panjang sama dengan X.
FUN: fungsi yang akan digunakan. Fungsi yang dapat digunakan dapat berupa fungsi dasar matematika atau statistika, serta user define function. Subset juga dimungkinkan pada fungsi ini.
…: opsional argumen pada fungsi yang digunakan.
simplify: logical. Jika nilainya TRUE maka output yang dihasilkan adalah bentuk skalar.

```
## membuat tabel frekuensi
groups <- as.factor(rbinom(32, n = 5, prob = 0.4))

tapply(groups, groups, length)
# atau
table(groups)
## membuat tabel kontingensi
# menghitung jumlah breaks berdasarkan faktor jenis wool
# dan tensi level
tapply(X=warpbreaks$breaks, INDEX=warpbreaks[,-1],
FUN=sum)
```

```
# menghitung mean panjang gigi babi hutan
berdasarkan
# jenis suplemen dan dosisnya
tapply(ToothGrowth$len, ToothGrowth[,-1], mean)
# menghitung mpg minimum berdasarkan jumlah
silinder pada mobil
tapply(mtcars$mpg, mtcars$cyl, min,
simplify=FALSE)
```