# Pengembangan Fungsi dalam R

KOMPUTASI STATISTIK – PERTEMUAN KE-9

# Membuat fungsi

```
func_name <- function (argument) {
statement
}
```

- function: Fungsi dasar dalam membuat suatu fungsi
- statement yang berada di dalam kurung kurawal, bisa diabaikan jika statement berbentuk satu baris (kode satu baris)
- func_name: Nama dari fungsi
- Gunakan fungsi return() agar fungsi memberikan suatu hasil.

# Membuat fungsi

```
salam <- function() “Halo semuanya”

salam()
```

```
op <- function(val) return (val^2+val)

op(9)
```

- Banyaknya variabel harus sama dengan banyaknya argument dalam fungsi, kalau tidak akan muncul error
- Contoh: op(), akan muncul error seperti apa?
- Contoh: op(8,5) akan muncul error seperti apa?

# Default Argument

```
func_name <- function (argument=val) {
statement
}
```

- Di dalam mendeklarasikan suatu fungsi, argumen bisa dibuat sedemikian sehingga nilainya default
- Jika dalam pemanggilan fungsi tidak ada deklarasi untuk variabel, maka variabel yang akan digunakan adalah variabel default

# Default Argument

```
jumlah <- function(x=8, y=8){
   return(x+y)
}
```

Periksalah nilai dari:

- jumlah(5,6)
- jumlah(,9)
- jumlah(4,)
- jumlah()

Apakah kesimpulan yang bisa diambil dari fungsi ini?

# Fungsi Rekursi

```
func_name <- function (argument=val) {
   func_name(val)
}
```

- Rekursi: Fungsi yang memanggil dirinya sendiri.
- Harus berhati-hati agar fungsi yang berada di dalam fungsi tersebut tidak memunculkan output error

# Fungsi Rekursi

```
jum <- function(n){
  if(n==1){
    return (1)
  } else {
    return (n+jum(n-1))
  }
}
```

- Fungsi jum memberikan jumlahan n bilangan asli pertama
- Nilai jum(n) = n + jum(n-1).

# Latihan Fungsi

- Buatlah fungsi untuk menghitung nilai n faktorial, diberikan n bilangan asli, dengan argumen n.
- Buatlah fungsi untuk menghitung nilai kombinasi $C_r^n$, dengan argumen n dan r.
- Buatlah fungsi root(a,b,c), dengan outputnya adalah akar-akar dari persamaan kuadrat ax^2+bx+c = 0.

# Pemrograman berbasis objek di R

- R telah mengimplementasi pemrograman berorientasi objek
- Semua dalam R adalah objek
- Implementasi objek dalam R berbeda dengan pemrograman berorientasi objek lain seperti: C++, python, dll
- R memiliki tiga sistem kelas, S3, S4, dan kelas referensi

# Kelas S3

- Kelas yang sederhana dan primitive
- Kelas ini yang paling popular dalam pemrograman R.
- Tidak ada definisi formal untuk kelas ini
- Objek dari kelas ini dapat dibuat dengan menambahkan atribut kelas ke dalam objek tersebut
- Suatu list dengan atribut kelasnya diberikan suatu nama kelas, adalah objek kelas S3

# Kelas S3

```
m <- list(nama="andi", umur=19, GPA=3.2)
class(m) <- "mahasiswa itera"
m
```

- Bagaimanakah hasil dari objek m?
- Apakah nilai dari m$nama, m$umur, dan m$GPA?

# Kelas S3

```
b <- list(x=runif(10), y=runif(10,1,3))
class(b) <- "bilangan acak"
b
```

- Bagaimanakah hasil dari objek m?
- x dan y adalah suatu vektor, sehingga dapat dipanggil elemen indeks tertentu dari vektor x dan y
- b$x[1]=?, b$y[9]=?

# Konstruktor

- Langkah sederhana dalam membuat objek dari suatu kelas sebelumnya sangat tidak dianjurkan karena nilai-nilainya mungkin berupa sesuatu yang tidak tepat.
- Fungsi konstruktor dibutuhkan untuk mengecek kesesuaian nilai-nilai pada objek
- Contoh: GPA haruslah berada pada rentang [0,4]
- Umur (mungkin) haruslah kurang dari 25, dan sebagainya.

# Konstruktor

```
mhs <- function(n,u,g){
  if(g>4 || g<0){
    stop("GPA harus berada pada rentang 0 sampai
4")
  }
  mahasiswa <- list(nama=n,umur=u,gpa=g)
  class(mahasiswa) <- "mhs itera"
  mahasiswa
}
```

# Konstruktor

- Cobalah: m <- mhs(“Jakob”,19,3)
- Apa yang akan terjadi jika memasukkan mhs(“Ingrid”,24,4.5)?
- Latihan: Buatlah langkah pemberhentian untuk umur, berikan batasan bahwa nilai u harus berada pada rentang [13,25]
- Coba kode berikut ini: m$gpa <- 8
- Apa yang terjadi? apa yang dapat anda simpulkan?

# Fungsi Generik

- Merupakan suatu method dari suatu class objek dalam R
- Fungsi generic bertindak untuk beralih memilih fungsi tertentu atau metode tertentu yang dijalankan sesuai dengan classnya
- Untuk mendefinisi ulang suatu fungsi generic digunakan syntax: `method.class <- function() expression`
- Terdapat beberapa fungsi generic yang sudah ada: print, length,dll
- Namun bisa juga membuat fungsi generic sendiri.

# Fungsi Generik

```
mhs <- function(n,u,g){
  if(g>4 || g<0){
    stop("GPA harus berada pada rentang 0 sampai
4")
  }
  mahasiswa <- list(nama=n,umur=u,gpa=g)
  class(mahasiswa) <- "mahasiswa"
  mahasiswa
}
```

# Fungsi Generik

```
print.mahasiswa <- function(obj){
  cat(obj$nama, "\n")
  cat(obj$umur, "\n")
  cat(obj$gpa, "\n")
}
```

- Bagaimanakah tampilan dari objek m sekarang?

# Fungsi Generik

Membuat fungsi generic sendiri dengan fungsi UseMethod

```
grade<-function(obj){
  UseMethod("grade")
}
grade.mahasiswa <- function(obj){
  cat("IPK", obj$nama, "adalah", obj$gpa)
}
```

# Fungsi Generik

- Dari fungsi generic terakhir, tentukan hasil dari grade(m)
- Latihan: Buatlah fungsi generic isCumlaude untuk menentukan apakah mahasiswa m mendapatkan predikat cumlaude (kriteria hanya dipandang ketika ipk>=3.5)

# Kelas S4

- Mengatasi masalah dalam kelas S3 dengan sistem objek lebih formal
- Setiap objek didefinisikan secara formal dalam suatu class
- Sebuah class terdiri dari slot dengan tipe atau class spesifik
- Kelas S4 didefinisikan menggunakan fungsi setClass()

```
setClass(“mahasiswa”,
slots=list(nama=“character”, umur=“numeric”,
gpa=“numeric”
```

# Kelas S4

- Mengatasi masalah dalam kelas S3 dengan sistem objek lebih formal
- Setiap objek didefinisikan secara formal dalam suatu class
- Sebuah class terdiri dari slot dengan tipe atau class spesifik
- Kelas S4 didefinisikan menggunakan fungsi setClass()

```
setClass(“mahasiswa”,
slots=list(nama=“character”, umur=“numeric”,
gpa=“numeric”))
```

# Kelas S4

- Objek S4 dapat dibuat dengan menggunakan fungsi new()

```
s <- new("mahasiswa", nama="Matt Murdock",
umur=21, gpa=3.8)

s
```

- Bagaimanakah susunan dari objek s?

# Aksesor S4

- Akses terhadap slot menggunakan fungsi slot atau operator @

```
s@nama

s@umur

s@gpa
```

- Bisa diupdate dengan nilai yang baru: s@nama <- “Jon Snow”
- Bagaimana dengan objek s sekarang?

# Fungsi Generik

- Fungsi generic yang ekuivalen dengan print() pada S3 adalah fungsi show()
- Pendeklarasian fungsi generic menggunakan fungsi setMethod()

```
setMethod("show","mahasiswa",
   function(obj){
   cat(obj@nama, "\n")
   cat(obj@umur, "\n")
})
```

# Fungsi Generik

- Membuat fungsi generic baru

```
setMethod(“grade","mahasiswa",
   function(obj){
   cat(“IP anda adalah”, obj@gpa)
})
```

Terima Kasih.