MATERI 3
Pengembangan Fungsi & Pemrograman Berorientasi Objek

STK571 KOMPUTASI STATISTIK

Outline

- Menciptakan Fungsi
- Pemrograman berorientasi object (OOP)
- Ilustrasi

Menciptakan Fungsi Sendiri

- Fungsi yang tidak ada dalam bahasa S dapat diciptakan sendiri.
- Syntax :

  namafungsi <- function(argumen) isifungsi
- Teladan:

  std.dev <- function (x) sqrt(var(x))
- Komentar bisa dituliskan menggunakan #

Menciptakan Fungsi Sendiri

- Output dari fungsi adalah objek → umumnya menggunakan objek list
- Untuk memanggil fungsi :

  namafungsi (arg1, arg2, ….)

Penanganan Kesalahan

- Untuk menangani kesalahan dalam fungsi disediakan fungsi-fungsi:
  - try
  - tryCatch
  - warnings
  - stop

Argumen Fungsi

- Fungsi umumnya menggunakan argumen
- Argumen fungsi dalam R dapat diberikan sebuah nilai default
- Argumen dapat tak terhingga (menggunakan …)
- Sebuah fungsi di dalam R dapat dijadikan sebagai argumen

Teladan

- Membuat Fungsi untuk Melakukan Pengujian Hipotesis Nilai Tengah untuk Dua Populasi dengan ragam sama.
- Algoritma :
  - hitung masing-masing n
  - hitung galat baku
  - hitung statistik uji
  - hitung nilai-p
  - tetapkan keputusan penerimaan atau penolakan H_0

```
ttest <- function(y1, y2, test = "dua-arah", alpha = 0.05)
{
  n1 <- length(y1); n2 <- length(y2)
  ndf <- n1+n2-2
  s2 <- ((n1-1)*var(y1) + (n2-1)*var(y2))/ndf
  tstat <- (mean(y1)-mean(y2))/sqrt(s2*(1/n1+1/n2))
  tail.area <-
    switch(test,
       "dua-arah" = 2 * (1-pt(abs(tstat), ndf)),
       kecil=pt(tstat,ndf),
       besar= 1-pt(tstat,ndf),
       {
         warning ("uji harus 'dua-arah', 'kecil' atau 'besar'")
         NULL
       }
    )
  list(tstat=tstat,df=ndf,reject=if(!is.null(tail.area))
       tail.area < alpha, tail.area=tail.area)
}
```

Pemrograman Berorientasi Objek

Objek dan Class

- **Objek**
  - **The state** of an object encompasses all of the (static) properties of the object plus the current (dynamic) values of each of these properties
  - **Behavior** is how an object acts and reacts, in terms of state changes and interactions with other objects.
  - **Identity** is the property of an object that distinguishes it from all other objects.
- **Class** adalah kumpulan objek yang berbagi struktur dan tingkah laku yg sama/umum
- **Class** merupakan abstraksi dari objek

Konsep Pemrograman OO

- Tidak semua konsep dari pemrograman berorientasi objek diterapkan dalam R
- Konsep yang diimplementasikan:
  - Encapsulation
  - Inheritance
  - Polymorphism

Encapsulation

Encapsulation means that a group of related properties, methods, and other members are treated as a single unit or object. Objects can control how properties are changed and methods are executed.

Why: Makes it easier to change your implementation at a later date by letting you hide implementation details of your objects, a practice called *data hiding*.

Inheritance

Inheritance describes the ability to create new classes based on an existing class. The new class inherits all the properties and methods and events of the base class, and can be customized with additional properties and methods.

Why: Promotes code reuse since the code for the methods of the subclasses do not need to be rewritten.

Polymorphism

Polymorphism means that you can have multiple classes that can be used interchangeably, even though each class implements the same properties or methods in different ways. Polymorphism is essential to object-oriented programming because it allows you to use items with the same names, no matter what type of object is in use at the moment.

Why: Polymorphism becomes more flexible. Subclasses can keep some methods inherited from their super classes and *override* others.

Ilustrasi Objek

- Suatu objek untuk manipulasi data numerik yang direpresentasikan dalam suatu lokasi koordinat x y.
- Cara mudah untuk menyimpan vektor lokasi koordinat adalah dalam list

```
pts <- list (x=round(rnorm(5),2),
y=round(rnorm(5),2))
```

Menjadikan sebagai Class S3

- Untuk menjadikan sebagai objek gunakan fungsi class

```
> class(pts) = "coords"

> pts
$x
[1]  0.69 -0.34  1.59  0.12 -0.40

$y
[1] -1.20  0.68 -0.23  0.61  0.57

attr(,"class")
[1] "coords"
```

Fungsi Generik dan Metode

- Atribut Class adalah dasar dari **mekanisme** OO sederhana sistem objek "S3"
- **Mekanisme** menggunakan tipe fungsi yang disebut fungsi generik
- Fungsi generik bertindak untuk beralih memilih fungsi tertentu atau metode tertentu yang dijalankan
- Metode tertentu yang dipilih tergantung dalam class argumen pertama
- Contoh: Akan lebih baik jika fungsi print untuk class coords memiliki metode yang berbeda dengan print list

Metode print

```
> print.coords =
      function(obj)
      {
            print(paste("(",
                        format(xcoords(obj)),
                        ", ",
                        format(ycoords(obj)),
                        ")", sep = ""),
                  quote = FALSE)
      }

> pts
[1] ( 0.22,  0.83) (-0.03, -1.02)
[3] ( 1.27,  1.65) ( 0.37,  1.32)
[5] (-0.37,  0.40)
```

Menciptakan fungsi generik

- Metode hanya dapat didefinisikan untuk fungsi yang generik
- Untuk menciptakan fungsi generik

```
> bbox =
      function(obj)
      UseMethod("bbox")
```

Menciptakan Metode bbox

```
> bbox.coords =
      function(obj)
      matrix(c(range(xcoords(obj)),
               range(ycoords(obj))),
             nc = 2,
             dimnames = list(
               c("min", "max"),
               c("x:", "y:")))
```

```
> bbox(pts)
        x:    y:
min -0.37 -1.02
max  1.27  1.65
```

Masalah dalam Sistem Objek S3

- Sistem Objek S3 dalam R memberikan fasilitas object-oriented, tetapi terlalu longgar
- Ekspresi berikut diperbolehkan dalam R

```
> model = 1:10; class(model) = "lm"

> class(x) = sample(class(x))
```

- Sistem objek S3 tidak secara lengkap dapat dipercaya
- Masih banyak technical issue dalam sistem objek S3

Sistem Objek S4

- Mengatasi masalah dalam sistem objek S3 dengan sistem objek lebih formal
- Salah satu keuntungan: sistem penurunan dari class/objek
- Dalam sistem objek formal: setiap objek didefinisikan secara formal dalam suatu class
- Sebuah class terdiri dari slot dengan tipe atau class spesifik
- Class dideklarasikan dengan fungsi setClass

```
> setClass("coords",
           representation(x = "numeric",
                          y = "numeric"))
[1] "coords"
```

Menciptakan objek

- Dapat menggunakan fungsi new

```
> pts = new("coords",
            x = rnorm(5), y = rnorm(5))
```

- Tidak disarankan → lebih baik membuat suatu konstruktor

```
coords =
    function(x, y) {
        if (length(x) != length(y))
            stop("equal length x and y required")
        if (!is.numeric(x) || !is.numeric(y))
            stop("numeric x and y required")
        new("coords", x = as.vector(x),
                      y = as.vector(y))
    }
```

Ilustrasi

- Misal menciptakan objek pts

```
> pts = coords(round(rnorm(5), 2),
               round(rnorm(5), 2))
```

- Metode print → metode show

```
> pts
An object of class "coords"
Slot "x":
[1]  1.31  1.15 -1.15 -0.46 -0.97

Slot "y":
[1]  0.64 -0.31  0.87  0.51 -1.34
```

Akses terhadap slot

- Menggunakan operator @

```
> pts@x
[1]  1.31  1.15 -1.15 -0.46 -0.97

> pts@y
[1]  0.64 -0.31  0.87  0.51 -1.34
```

- Tidak disarankan secara langsung → Create fungsi aksesor

```
> xcoords = function(obj) obj@x
> ycoords = function(obj) obj@y
```

Fungsi Generik show

- Fungsi generik show setara dengan fungsi generik print pada sistem objek S3
- Penciptaan fungsi generik menggunakan fungsi setMethod
- Argumen didefinisikan dalam signature

```
> setMethod(show, signature(object = "coords"),
            function(object)
            print(data.frame(x = xcoords(object),
                             y = ycoords(object))))
[1] "show"
attr(,"package")
[1] "methods"
```

```
> pts
      x     y
1  1.31  0.64
2  1.15 -0.31
3 -1.15  0.87
4 -0.46  0.51
5 -0.97 -1.34
```

Definisi Fungsi Generik

- Menggunakan setGeneric

```
> setGeneric("display",
             function(obj)
             standardGeneric("display"))
[1] "display"
> setMethod("display", signature(obj = "coords"),
            function(obj)
            print(paste("(",
                        format(xcoords(obj)),
                        ", ",
                        format(ycoords(obj)),
                        ")", sep = ""),
                  quote = FALSE))
[1] "display"
```

```
> display(pts)
[1] ( 1.31,  0.64) ( 1.15, -0.31)
[3] (-1.15,  0.87) (-0.46,  0.51)
[5] (-0.97, -1.34)
```

Bounding Box

- Ilustrasi metode untuk mendapatkan batas dari coords

```
> setGeneric("bbox",
             function(obj)
             standardGeneric("bbox"))
[1] "bbox"

> setMethod("bbox", signature(obj = "coords"),
            function(obj)
            matrix(c(range(xcoords(obj)),
                     range(ycoords(obj))),
                   nc = 2,
                   dimnames = list(
                     c("min", "max"),
                     c("x:", "y:"))))
[1] "bbox"
```

```
> pts
      x     y
1  1.31  0.64
2  1.15 -0.31
3 -1.15  0.87
4 -0.46  0.51
5 -0.97 -1.34

> bbox(pts)
        x:    y:
min -1.15 -1.34
max  1.31  0.87
```

Inheritance

- Terdapat class baru yang diturunkan dari coords dengan menambahkan slot nilai

```
> setClass("vcoords",
           representation(value = "numeric"),
           contains = "coords")
[1] "vcoords"
```

Fungsi konstruktor

- Fungsi konstruktor dan aksesor

```
> vcoords =
      function(x, y, value)
      {
          if (!is.numeric(x) ||
              !is.numeric(y) ||
              !is.numeric(value) ||
              length(x) != length(value) ||
              length(y) != length(value))
                  stop("invalid arguments")
          new("vcoords", x = x, y = y,
              value = value)
      }

> values = function(obj) obj@value
```

Ilustrasi

- Instanisasi class vcoords:

```
> vpts = vcoords(xcoords(pts), ycoords(pts),
                 round(100 * runif(5)))
```

- Print/show objek masih dari class coords

```
> vpts
      x     y
1  1.31  0.64
2  1.15 -0.31
3 -1.15  0.87
4 -0.46  0.51
5 -0.97 -1.34
```

Polymorphism dari metode show

- Metode show baru untuk vcoords:

```
> setMethod(show, signature(object = "vcoords"),
            function(object)
            print(data.frame(
                    x = xcoords(object),
                    y = ycoords(object),
                    value = values(object))))
[1] "show"
attr(,"package")
[1] "methods"
```

```
> vpts
      x     y value
1  1.31  0.64     7
2  1.15 -0.31    79
3 -1.15  0.87    19
4 -0.46  0.51    98
5 -0.97 -1.34    79
```

Fungsi Matematika

- Menciptakan fungsi cos untuk nilai

```
> setMethod("cos", signature(x = "vcoords"),
            function(x)
            vcoords(xcoords(x),
                    ycoords(x),
                    cos(values(x))))
[1] "cos"
> cos(vpts)
      x     y      value
1  1.31  0.64  0.7539023
2  1.15 -0.31 -0.8959709
3 -1.15  0.87  0.9887046
4 -0.46  0.51 -0.8192882
5 -0.97 -1.34 -0.8959709
```

Fungsi Matematika

- Atau menggunakan group

```
> setMethod("Math", signature(x = "vcoords"),
            function(x)
            vcoords(xcoords(x),
                    ycoords(x),
                    callGeneric(values(x))))
[1] "Math"
```

```
> sqrt(vpts)
      x     y    value
1  1.31  0.64 2.645751
2  1.15 -0.31 8.888194
3 -1.15  0.87 4.358899
4 -0.46  0.51 9.899495
5 -0.97 -1.34 8.888194

> tan(vpts)
      x     y     value
1  1.31  0.64 0.8714480
2  1.15 -0.31 0.4956775
3 -1.15  0.87 0.1515895
4 -0.46  0.51 0.6998537
5 -0.97 -1.34 0.4956775
```

Fungsi dalam group Math

- Fungsi-fungsi yang termasuk dalam group Math

```
abs, sign, exp, sqrt, log, log10, log2,
cos, sin, tan, acos, asin, atan,
cosh, sinh, tanh, acosh, asinh, atanh,
ceiling, floor, trunc,
gamma, lgamma, digamma, trigamma
cumprod, cumsum, cummin, cummin.
```

Operasi Dua Buah Objek

- Terdapat dua group: Arith dan Compare
  - The arithmetic operators:

    `+, -, *, ^, %%, %/%, /`
  - The comparison operators:

    `==, >, <, !=, <=, >=`
- Keduanya berasal dari group yg lebih besar: Ops

Operasi Dua Buah Objek

- Operasi dilakukan untuk objek dari lokasi yang sama

```
> sameloc =
      function(e1, e2)
      (length(values(e1)) == length(values(e2))
      || any(xcoords(e1) == xcoords(e2))
      || any(ycoords(e1) == ycoords(e2)))
```

Operasi Dua Buah Objek

```
> setMethod("Arith", signature(e1 = "vcoords",
                               e2 = "vcoords"),
      function(e1, e2)
      {
          if (!sameloc(e1, e2))
              stop("identical locations required")
          vcoords(xcoords(e1),
                  ycoords(e1),
                  callGeneric(values(e1),
                              values(e2)))
      })
[1] "Arith"
```

```
> vpts
      x     y value
1  1.31  0.64     7
2  1.15 -0.31    79
3 -1.15  0.87    19
4 -0.46  0.51    98
5 -0.97 -1.34    79

> vpts + vpts
      x     y value
1  1.31  0.64    14
2  1.15 -0.31   158
3 -1.15  0.87    38
4 -0.46  0.51   196
5 -0.97 -1.34   158
```

```
> setMethod("Compare", signature(e1 = "vcoords",
                                 e2 = "vcoords"),
      function(e1, e2)
      {
          if (!sameloc(e1, e2))
              stop("identical locations required")
          callGeneric(values(e1), values(e2))
      })
[1] "Compare"
```

Penambahan Metode Arith

```
> setMethod("Arith",
      signature(e1 = "numeric",
                e2 = "vcoords"),
     function(e1, e2) {
         if (length(e1) > length(values(e2)))
             stop("incompatible lengths")
         vcoords(xcoords(e2),
                 ycoords(e2),
                 callGeneric(as.vector(e1),
                             values(e2)))
             })
[1] "Arith"
```

Memeriksa class

- Menggunakan fungsi is

```
> is(vpts, "vcoords")
[1] TRUE

> is(vpts, "coords")
[1] TRUE
```

Coercion objek

- Menggunakan fungsi as

```
> as(vpts, "coords")
      x     y
1  1.31  0.64
2  1.15 -0.31
3 -1.15  0.87
4 -0.46  0.51
5 -0.97 -1.34
```

Subset

- Mendefinisikan Operator Subset "["

```
> setMethod("[",
            signature(x = "vcoords",
                      i = "ANY",
                      j = "missing",
                      drop = "missing"),
            function(x, i, j)
            vcoords(xcoords(x)[i],
                    ycoords(x)[i],
                    values(x)[i]))
[1] "["
```

```
> vpts[1:3]
      x     y value
1  1.31  0.64     7
2  1.15 -0.31    79
3 -1.15  0.87    19

> vpts[values(vpts) > 50]
      x     y value
1  1.15 -0.31    79
2 -0.46  0.51    98
3 -0.97 -1.34    79
```

AKHIR MATERI