Dasar-dasar Pemrograman *Datamining* di R: Pengelolaan Data

Sigit Wahyu Kartiko
gsigit[at]gmail[dot]com

Pendahuluan

R bukan hanya software paket statistik namun juga lingkungan pemrograman. R menyediakan lingkungan yang cukup lengkap bagi pengguna untuk memecahkan persoalan. Ketersediaan berbagai paket tambahan terutama interface dengan bahasa lain, koneksi database, metode statistika, dan matematika memudahkan pengguna dalam menyelesaikan persoalan tersebut.

Sebelum melangkah ke *datamining*, prasyarat yang harus dipenuhi oleh pengguna adalah mengetahui terlebih dahulu dasar-dasar pemrograman dalam lingkungan R. Salah satu pengetahuan dasar tentang pemrogaman di R adalah pengelolaan data yang meliputi impor data, pengoperasian data dan ekspor data.

Mengelola Data

Impor Data

Langkah pertama dalam mengolah data adalah mengimpor data dari file. R memiliki *package* tambahan agar dapat mengimpor dari beberapa format file. File yang dapat diimpor oleh R antara lain format : SPSS, SAS, STATA, SYSTAT, CSV (delimited text file), dan Excel. Tidak hanya file yang bisa diimpor, konek langsung ke DBMS pun (database) dapat dengan mudah dilakukan. Jika konek ke DBMS dapat dilakukan maka selanjutnya tinggal melakukan operasi database menggunakan bahasa SQL.

Text File

Text File yang dimaksud disini adalah data yang terdiri dari kolom dan baris yang disimpan dalam text file. Data text file dapat diperoleh melalui file excel yang biasa disebut CSV (comma separated value), artinya antar kolom dalam tabel excel terpisah (delimited) oleh tanda koma. Pemisah antar kolom tidak harus tanda koma, dapat juga tanda tab. Cara yang paling mudah untuk membuat text file ini adalah copy tabel berisi apa saja dari file excel lalu paste ke notepad, maka antar kolom terpisah oleh tanda tabulasi (tab).

Listing 1: Impor file text

```
data_text <- read.csv(file="./contoh_file.txt", header = T,
                      sep = "\t",
                      quote="\"",
                      dec=".",fill=TRUE)
```

Database

Database yang dimaksud adalah database yang menggunakan bahasa manipulasi SQL (structured query language). Contoh aplikasi database ini antara lain SQL server, MySQL, PostgreSQL, SQLite, MS Access, Oracle dan masih banyak lagi yang lain.

Kali ini penulis ingin menggunakan driver JDBC (Java DataBase Connectivity) untuk menghubungkan antara server dan client. Server yang dimaksud disini adalah server database dan client di sini tidak lain adalah R.

Listing 2: Impor data melalui JDBC

```
library(RJDBC) #load library RJDBC

#variabel driver menyimpan driver JDBC
driver <- JDBC("com.mysql.jdbc.Driver",
               "/etc/jdbc/mysql-connector-java-3.1.14-bin.jar")

#variabel koneksi menyimpan driver JDBC
koneksi <- dbConnect(driver,
                     "jdbc:mysql://localhost/test",
                     "username",
                     "pasword")

#menampilkan tabel-tabel
dbListTables(koneksi)

#meng-attach dataframe "iris" ke dalam konteks
data(iris)
#meng-insert dataframe "iris" ke dalam tabel "iris" di database
dbWriteTable(koneksi, "iris", iris)

#variabel d menyimpan dataframe hasil Query
d <- dbGetQuery(koneksi, "select count(*) from iris")

#variabel d menyimpan dataframe dari tabel "iris"
d <- dbReadTable(koneksi, "iris")
```

Langkah kerja dalam mengimpor data melalui JDBC adalah sebagai berikut:

1. Memanggil *package/library* RJDBC dengan fungsi ***library()***.
2. Simpan *driver handle* ke dalam variabel tertentu dengan fungsi ***JDBC()***.

   Pada **listing 2** koneksi dipanggil dengan fungsi JDBC() yang didalamnya mengandung argumen nama driver dan file driver jdbc mysql. Setelah berhasil, handling driver tersebut disimpan dalam variabel **driver.**
3. Melakukan koneksi dengan fungsi ***dbConnect()***

   Pada listing 2 dicontohkan bahwa fungsi dbConnect mengandung 4 argumen. Argumen pertama adalah variabel driver handler dan argumen kedua adalah url koneksi ke database dengan format url jdbc. Argumen ke-3 dan ke-4 adalah *username* dan *password* ke database.
4. Melakukan berbagai operasi pengelolaan data seperti ***dbListTables()*** yang berfungsi menampilkan nama tabel didalam database. Fungsi ***dbWriteTable()*** yang berguna dalam menyimpan dataframe/list ke dalam tabel database. Fungsi ***dbGetQuery()*** untuk melakukan perintah SELECT. Atau dapat langsung membaca seluruh isi tabel dengan fungsi ***dbReadTable()***.

Hal yang perlu dilakukan sebelum dapat meload data melalui JDBC adalah instalasi paket RJDBC. Langkah-langkah menginstall paket RJDBC adalah sebagai berikut:

1. Install Java Development Kit (JDK) versi terbaru dan tetapkan variabel *environment* $JAVA_HOME dalam terminal dengan cara edit file di ~/.bashrc
2. Edit konfigurasi kompilasi R dengan menggunakan perintah:

   ```
   root# R CMD javareconf
   ```

   Diwajibkan akses sebagai root.
3. *Compile* dan *Install* paket DBI (contoh versi 0.2-4) dengan perintah:

   ```
   $ R CMD INSTALL DBI_0.2-4.tar.gz
   ```
4. *Compile* dan *Install* paket rJava (contoh versi 0.8.1) dengan perintah:

   ```
   $ R CMD INSTALL rJava_0.8-1.tar.gz
   ```

Excel

Bagi pengguna windows maka dengan menggunakan paket RODBC hal ini dapat dengan mudah melakukan impor file excel. Sedangkan bagi pengguna linux, ODBC harus diinstall terlebih dahulu dengan aplikasi unix ODBC. Saran praktis dari penulis adalah sebaiknya file excel di save as saja menjadi file csv. Dengan demikian dapat dengan mudah diimpor sebagai file dalam format text biasa.

Listing 2: Impor file Excel

```
library(RODBC) #load library RODBC

#variabel koneksi menghandle file excel
koneksi <- odbcConnectExcel("c:/file_excel.xls")

#variabel d menyimpan isi sheet1
d <- sqlFetch(koneksi, "sheet1")
```

Operasi Data

Variabel

Untuk menetapkan variabel maka nama variabel tidak boleh berupa angka, namun huruf alfabetik a-z dan/atau campuran dari huruf alfabet, angka dan karakter diantaranya (tanpa tanda petik) "_" dan ".". Meng-*assign* variabel dapat menggunakan tanda panah ke kiri "<-" , ke kanan "->" atau sama dengan "=". Jika menggunakan tanda panah kiri atau sama dengan maka nama variabel harus berada di sebelah kiri, namun sebaliknya jika tanda panah kanan yang digunakan maka variabel harus berada di sebelah kanan.

Listing 3: Penamaan Variabel

```
#nama variabel var.1 terdiri dari huruf, titik dan angka
var.1 <- "hello"
var.2 <- "world"
# menggabung dua kata menjadi kalimat dengan fungsi paste
# sep (separator) adalah pemisah antara kata
paste(var.1,var.2,sep=" ")
[1] "hello world"
# nama variabel angka_1 terdiri dari huruf, _ dan angka
# assigment variabel menggunakan tanda = dan panah kanan ->
angka_1 = 1
2 -> angka_2
#penjumlahan
angka_1 + angka_2
[1] 3
```

Tipe Data

Seperti bahasa S, R memiliki beberapa jenis tipe data yang membantu dalam pemrograman statistika dan matematika yang meliput tipe data *scalar, vector* (numerik, karakter, logical), matriks, *dataframe/list* dan *factor/ordered.*

VECTOR

Data vector dibuat dengan cara mengapitkan kumpulan angka, karakter atau logical dengan fungsi c().

Listing 4: data Vector

```
# vector angka 1 s.d 10
v1 <- c(1:10) #atau c(1,2,3,4,5,6,7,8,9,10)
# vector karakter a,b,c,d,ef
v2 <- c("a","b","c","d","ef")
# vector logical TRUE atau T saja dan FALSE atau F saja.
v3 <- c(TRUE,T,F,FALSE)
# cara memanggil elemen vector
v1[c(1,5)]    # menggunakan vector indeks 1 5
[1] 1 5
v2[1:2]       # menggunakan urutan indeks a b
[1] "a" "b"
v3[2]  # TRUE
[1] TRUE
```

MATRIKS

Matriks sangat penting dalam operasi matematika terutama dalam memecahkan persamaan aljabar. Matrik mempunyai dimensi yaitu baris x kolom.

Listing 5: data Matriks

```
# membuat matriks dimensi 2 x 3
mtx.1 <- (1:6,nrow=2,ncol=3)

# membuat matriks 2 x 2 secara bertahap
nama_baris <- c("b1", "b2")        #menamai baris
nama_kolom <- c("k1", "k2")        #menamai kolom
sel <- c(10,20,30,40)              #membuat 4 sel matriks

# memasukkan variabel sel, nama_baris, nama_kolom di matrix
# byrow=TRUE artinya sel diisi dari baris ke barisi
mtx.2 <- matrix(sel, nrow=2, ncol=2, byrow=TRUE,
              dimnames=list(nama_baris, nama_kolom))

# memanggil matriks dengan subscripts [baris,kolom]
mtx.2[1,2]    # baris 1 kolom 2
[1] 20
mtx.2[2,]     # semua baris 2
k1 k2
30 40
mtx.2[,2]     # semua kolom 2
b1 b2
20 40
```

ARRAYS

Arrays dan matriks sebenarnya sama, namun arrays bisa memiliki dimensi lebih dari 2.

Listing 6: data Arrays

```
# array berdimensi 2 x 4 x 2 dengan isi sel 1 s.d 16
array(1:16, c(2,4,2))
, , 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
```

DATAFRAME/LIST

Dataframe/List merupakan tipe data yang siap digunakan dalam analisis statistika. Seperti halnya matriks, *dataframe* memiliki baris dan kolom. Namun, dataframe dapat memiliki tipe data yang berbeda antar kolom *(numeric, character, factor,* dan sebagainya) mirip halnya dengan yang dimiliki oleh dataset SAS dan SPSS. Dataframe dapat diibaratkan seperti tabel dalam

database, yang memiliki tipe *field* (kolom) yang bisa berbeda-beda.

Dataframe dapat dibuat melalui beberapa cara yaitu 1) Secara bertahap, 2) Menggunakan dataframe yang sudah ada dalam *library/package*, dan 3) Impor data dari file.

1. Secara bertahap

Listing 7: membuat dataframe secara bertahap

```
kolom.1 <- c(15,20,18,25)   #membuat kolom 1 bertipe numeric
kolom.2 <- c("sepeda", "bus", "kereta", NA) #kolom 2 bertipe
karakter
kolom.3 <- c(T,T,T,F)              #kolom 3 berkolom logical
df.1 <- data.frame(kolom.1,kolom.2, kolom.3) #menyisipkan ko-
lom-kolom
names(df.1) <- c("umur", "kendaraan", "pelajar") #memberi nama
kolom
df.1          #menampilkan dataframe
```

```
  umur kendaraan pelajar
1   15    sepeda    TRUE
2   20       bus    TRUE
3   18    kereta    TRUE
4   25      <NA>   FALSE
```

2. Menggunakan dataframe yang sudah ada dalam library/package

   R sudah menyediakan beberapa contoh dataframe yang ada dalam library bawaannya. Library "datasets" sebagai misal, memiliki banyak sekali contoh-contoh dataframe baik nyata maupun fiktif seperti "*cars*" berisi tentang observasi kecepatan dan jarak yang diperlukan untuk berhentinya mobil, "*mtcars*" (motor trend car road tests), "*AirPassengers*" berisi data time series bulanan tentang jumlah penumpang pesawat dalam kurun waktu tahun 1949-1960 dan masih banyak lagi. Anda dapat mencari dataframe dalam library-library bawaan R dengan perintah sebagai berikut :

Listing 8: melihat dataframe di semua package

```
data(package = .packages(all.available = TRUE))
```

   ketikkan saja nama *dataframe* tersebut, contoh "cars":

Listing 9: dataframe cars

```
cars
```

```
   speed dist
1      4    2
2      4   10
3      7    4
4      7   22
5      8   16
...dst
```

3. Hasil impor data dari file merupakan tipe data dataframe/list.

   Silakan lihat kembali subbab tentang impor data dari file.

FACTOR/ORDERED

R menghandle data kategorikal dengan menggunakan *factor* atau *ordered. Ordered* memiliki sedikit perbedaan dibandingkan dengan *factor* yaitu, *ordered* digunakan untuk data kategori

yang bersifat urutan *(ordinal),* sedangkan faktor hanya memetakan kategori ke dalam index angka. Jadi dalam pemrograman obyek oriented, tipe data factor diturunkan (inherited) dari tipe data factor.

Listing 10: Factor/Ordered

```
> #contoh variabel kualitatif
> label.tipe <- c("rendah", "sedang", "tinggi")
#contoh hasil observasi aktivitas blogging 10 blogger
#1=rendah, 2=sedang, 3=tinggi
> hasil.observasi <- c(1,1,2,1,1,3,2,2,1,2)
> f.hasil.observasi <- factor(hasil.observasi) #mengubah ke tipe factor
#output sebelum pemetaan kategori
> f.hasil.observasi
 [1] 1 1 2 1 1 3 2 2 1 2 2 3 3 3 2
Levels: 1 2 3
> #memetakan kategori
> levels(f.hasil.observasi)<-label.tipe
> #output sesudah pemetaan kategori
> f.hasil.observasi
[1] rendah rendah sedang rendah rendah tinggi sedang sedang
rendah sedang
Levels: rendah sedang tinggi
> #mengubah factor ke dalam ordered
> o.hasil.observasi <- as.ordered(f.hasil.observasi)
> #output setelah konversi ke ordered
> o.hasil.observasi
 [1] rendah rendah sedang rendah rendah tinggi sedang sedang
rendah sedang
Levels: rendah < sedang < tinggi
```

Pengoperasian Data

Fungsi-fungsi yang cukup penting dalam pengoperasian variabel dan data antara lain adalah:

1. Melisting dengan **`ls()`**, mengedit dengan **`edit(nama_object)`**, dan menghapus dengan **`rm(object1,object2,...)`** object-object yang aktif.

   Listing 11:ls(), rm(), dan edit()

   ```
   > ls()
   [1]"df.1" "angka.1" "angka.2"
   > rm(angka.1,angka.2)
   > # jangan lakukan perintah di bawah ini
   > # kecuali anda ingin menghapus semua object yang aktif !!!
   > # rm(list=ls()) #menghapus semua object
   > df.1.baru <- edit(df.1) #copy lalu edit df.1 ke df.1.baru
   ```

2. Menempelkan dataframe dalam konteks dengan attach(nama_dataframe)

   Tujuan fungsi **`attach()`** adalah menempelkan dataframe ke dalam konteks, sehingga untuk memanggil nama kolom tidak perlu menuliskan nama dataframe lagi, cukup dengan nama kolomnya.

   Listing 12: attach dataframe cars

```
> #sebelum di attach
> cars$speed #memanggil kolom speed dalam dataframe cars
 [1]  4  4  7  7  8  9 10 10 10 11 11 12 12 12 12 13 13 13
13 ...
[26] 15 16 16 17 17 17 18 18 18 18 19 19 19 20 20 20 20 20
22 ...
> attach(cars) #menempelkan cars dalam konteks
> #sesudah di attach
> speed #cukup panggil nama kolomnya
 [1]  4  4  7  7  8  9 10 10 10 11 11 12 12 12 12 13 13 13
13 ...
[26] 15 16 16 17 17 17 18 18 18 18 19 19 19 20 20 20 20 20
22 ...
```

3. Melihat struktur data menggunakan **`str(nama_object)`**, **`class(nama_object)`** dan **`unclass(nama_object)`**

Listing 13: melihat struktur data object

```
> str(df.1)           # melihat struktur object
'data.frame':   4 obs. of  3 variables:
 $ umur     : num  15 20 18 25
 $ kendaraan: Factor w/ 3 levels "bus","kereta",..: 3 1 2 NA
 $ pelajar  : logi  TRUE TRUE TRUE FALSE
> class(df.1)         # melihat class atau tipe object
[1] "data.frame"
> # melihat struktur object lebih lengkap
> # seperti nama-nama field, atribut dan fungsi
> unclass(df.1)
$umur
[1] 15 20 18 25
$kendaraan
[1] sepeda bus    kereta <NA>
Levels: bus kereta sepeda
$pelajar
[1]  TRUE  TRUE  TRUE FALSE
attr(,"row.names")
[1] 1 2 3 4
```

Ekspor Object, Data dan Output

Object, Data dan Output dalam terminal dapat diekspor dalam file. Hal ini mungkin dapat dilakukan dengan cara yang praktis yaitu *copy* dari terminal lalu paste ke dalam file text. Namun kalau jumlah barisnya melampaui lebar terminal maka sebaiknya anda gunakan saja fungsi **`capture.output(variabel/perintah,file=nama_file)`** untuk mengekspor tampilan terminal.

Listing 14: capture output

```
> #ekspor dataframe df.1 ke dalam filetext output_1.txt
> capture.output(df.1,file="output_1.txt")
> #ekspor output perintah ls() ke dalam file text output_2.txt
> capture.output(ls(),file="output_2.txt")
```

Penutup

Pembahasan kali ini merupakan dasar pemrogaman di R sebelum melangkah ke pembahasan *datamining*. Dasar-dasar pemrograman R tersebut adalah pengelolaan data yang meliputi impor-ekspor data, mengenal tipe data, dan operasi data. Setelah memahami pengelolaan data, di pembahasan berikutnya pengguna harus dapat memanfaatkan lingkungan R untuk menghasilkan informasi visual melalui fasilitas *plotting*.

Referensi

1. Ihaka, R. & Gentleman, R. (1996). ***"R: A Language for Data Analysis and Graphics". Journal of Computational and Graphical Statistics 5 (3): 299–314***. www.jstor.org
2. Rossiter, D. G. (2009). ***Introduction to the R Project for Statistical Computing for use at ITC.*** Accessed:03-03- 2010. http:// www.itc.nl/ personal/ rossiter
3. Williams, Graham J. ***Rattle: A Data Mining GUI for R***. Accessed:06-04-2012. http://journal.r-project.org/archive/2009-2/RJournal_2009-2_Williams.pdf
4. http://www.R-project.org

Biografi Penulis

Sigit Wahyu Kartiko. Menyelesaikan DIV di Sekolah Tinggi Akuntansi Negara dan meraih gelar Magister Ekonomi di Fakultas Ekonomi Universitas Indonesia. Memiliki kompetensi pada bidang *programming* (java, scala, R, C, C++), *database*, *accounting* (privat, publik), *public sector economics*.