Bab 4 Manipulasi data dengan R

Dasapta Erwin Irawan dan Prana Ugi

August 23, 2015

Contents

1 Manipulasi data dengan R base

1.1 Mengetahui struktur data

Mari kita mulai mencoba memanipulasi data dengan R. Bila anda tidak memiliki contoh data sendiri, kita coba dengan contoh data berikut ini, `Data1-WaterTemp.csv`. Data ini merupakan kumpulan data suhu air dari 20 lokasi. 1. Unduh data dari http://dasaptaerwin.net/wp/wp-content/uploads/2016/01/Data1-WaterTemp.csv. 2. Pindahkan file ke direktori kerja 3. Kemudian ketik atau *copy paste* beberapa perintah berikut ini.

```
data <- read.csv("Data1-WaterTemp.csv") # import data ke dalam R
data # melihat isi data atau anda bisa membuka file dari jendela _Global Environment_.
str(data) # untuk melihat struktur data (terlihat sebagai `int` atau `integer`)
dim(data) # untuk melihat dimensi data (jumlah kolom x jumlah baris) ada 20 baris atau sampel dan 2 kol
summary(data) # untuk melihat ringkasan data secara statistik
```

Anda dapat melihat bahwa `data` merupakan data frame yang terdiri dari 20 baris (atau sampel) dan 2 kolom. Seluruh data berjenis `int` atau integer. Kita akan mencoba memanipulasi data frame tersebut dengan R. Sebenarnya ada beberapa fungsi dasar untuk melakukan sortasi, transpose, dll, tapi syntax dan penggunaannya memerlukan baris perintah yang cukup panjang. Karena R sangat mudah untuk dikembangkan, maka kali ini kita akan menggunakan package `dplyr` yang dibuat oleh tim R Studio, Hadley Wickham. Dengan package ini mengolah data dapat dilakukan dengan syntax yang lebih mudah dipahami dan diingat.

Baris-baris perintah di bawah ini menggunakan dataset internal yang tersedia dalam package `dplyr`. Awali dengan menginstalasi `dplyr`. Sudah tahu bagaimana caranya bukan.

1.2 Menambah baris dengan `rbind()`

Untuk menggabungkan dua data frame (atau dataset) secara vertikal, anda dapat menggunakan fungsi `rbind()`. Kedua data frame harus memiliki jumlah dan nama variabel yang sama. Walaupun tidak perlu memiliki urutan variabel yang sama, tapi akan lebih baik bila urutannya pun sama. Variabel dalam hal ini adalah kolom.

Walaupun R dapat mengedit tabel atau dataset, tapi kami sangat menyarankan agar dataset dapat disunting dengan program *spreadsheet* (pengolah tabel) terpisah, seperti Ms Excel atau LibreOffice.

1.3 Menambah kolom dengan `cbind()`

Kita mungkin perlu menyatukan dua data frame, berdasarkan kolom (*column*) atau baris (*row*). Dalam R hal ini dilakukan dengan fungsi dasar `cbind()` (*column bind*) dan `rbind()` (*row bind*). Kita akan mengulang beberapa baris perintah yang pernah anda buat pada Bab 2.

1.4 Mengkombinasi dua data frame dengan `merge()`

Perintah ‘merge() adalah untuk mengkombinasi data frame dengan memperhatikan kolom atau baris yang sama dalam dua data frame yang berbeda.

```
# membuat data frame
datamurid <- data.frame(nama = c("Abi", "Aci", "Adi", "Afi", "Agi", "Ali"),
                        tahun.lahir = c(76, 78, 79, 80, 83, 87),
                        usia = c(38, 36, 35, 34, 31, 27)
                        )

datamurid2 <- data.frame(nama = c("Ani", "Ami", "Aki"),
                         tahun.lahir = c(76, 78, 79),
                         usia = c(38, 36, 35)
                         )

# menggabungkan dua data frame `merge()` vs `rbind()`
datamuridtotal <- merge(datamurid, datamurid2, by="nama")
datamuridrevisi <- rbind(datamurid, datamurid2)
datamuridtotal # merge gagal
datamuridrevisi # rbind berhasil

# operasi `merge()`
## membuat data frame baru
df1 <- data.frame(ref = c('Ref1', 'Ref2'),
                  label = c('Label01', 'Label02')
                  )

df2 <- data.frame(id = c('A1', 'C2', 'B3', 'D4'),
                    ref = c('Ref1', 'Ref2' , 'Ref3','Ref1'),
                    val = c( 1.11, 2.22, 3.33, 4.44 )
                    )
df1
df2

## `merge()`
dftotal <- merge(df1, df2, by='ref', all.y = T, sort= T)
```

```
dftotal
```

Terlihat bedanya?

1.5 Memanipulasi data dengan `dplyr` package

Package `dplyr` adalah package yang diperlukan untuk melakukan manipulasi data frame. Kelebihan `dplyr`:

- Fungsional untuk eksplorasi dan transformasi data: khusus dibuat untuk memenuhi kebutuhan analisis data berukuran besar (*big data*)
- Fungsi yang intuitif dan mudah dimengerti: fungsi-fungsi terkait dinamai mirip dengan kata kerja dalam bahasa sehari-hari (*daily English*).
- Cepat: ini baru terasa bila dataset kita berjumlah ribuan baris dan kolom.

Ada lima fungsi dasar dalam package ini:

- `filter()`: untuk memfilter baris (row) dari data frame berdasarkan beberapa kriteria yang anda miliki.
- `select()`: untuk memilih satu atau dua kolom (column) yang anda perlukan.
- `arrange()`: untuk menyortir (sorting), yaitu untuk mengurutkan baris berdasarkan nilai dari satu atau lebih kolom secara ascending atau descending.
- `mutate()`: untuk menambah kolom baru yang merupakan hasil transformasi dari kolom yang lain.
- `summarise()`: untuk memperlihatkan informasi dasar dari suatu data frame, misal: rata-rata (mean), standar deviasi (SD), dll. Fungsi ini sering digabung dengan fungsi `group()`.

Bagi yang belum menginstalasi package ini, silahkan melakukan instalasi dengan perintah berikut ini:

```
install.packages("dplyr")
```

1.5.1 Start

Mulailah dengan memuat library package `dplyr` dan data frame `hflights`. Data `hflights` adalah contoh data yang diambil dari data penerbangan dari dua bandara di Houston US pada tahun 2011.

```
# load packages
library(dplyr)
library(hflights)

# explore data
data(hflights)
head(hflights)
```

```
Tampilan sedikit agak kacau bukan. Kita bisa mengakalinya dengan membuat "local data frame" bernama "fl
```

```
# convert to local data frame
flights <- tbl_df(hflights)

# printing only shows 10 rows and as many columns as can fit on your screen
flights
```

```
Kalau masih kurang memuaskan, anda bisa mengatur berapa jumlah baris yang mau anda lihat (misalkan 20 b
```

```
# you can specify that you want to see more rows
print(flights, n=20)

# convert to a normal data frame to see all of the columns
data.frame(head(flights))
```

1.5.2 filter()

Fungsi `filter()` adalah untuk memilih baris atau mengelompokkan baris berdasarkan kriteria yang anda miliki. Fungsi ini sebenarnya telah tersedia dalam R secara default, namun package `dplyr` membuatnya lebih mudah. Sekarang kita bandingkan kedua fungsi Filter() tersebut. Perintah berikut ini melakukan filter kolom `Month` ("AND") kolom `DayofMonth` yang nilainya sama dengan 1.

```
# melihat semua penerbangan di tanggal 1 Januari menggunakan fungsi dasar R untuk

flights[flights$Month==1 & flights$DayofMonth==1, ]

# melihat semua penerbangan di tanggal 1 Januari menggunakan dplyr
# anda dapat menggunakan "," atau "&", hasilnya akan sama
filter(flights, Month==1, DayofMonth==1)
filter(flights, Month==1 & DayofMonth==1)
```

Anda bisa lihat bahwa dengan `dplyr` kita cukup mengetik nama kolomnya saja (misal "namaPenerbangan"), bukan `data.frame$namakolom` (misal: "data$namaPenerbangan").

Anda bisa menggunakan operator "ATAU" ("OR"), misalkan anda hanya ingin melihat data penerbangan dari perusahaan (kolom UniqueCarrier) "AA" atau "UA".

```
# use pipe for OR condition
filter(flights, UniqueCarrier=="AA" | UniqueCarrier=="UA")

#atau menggunakan operator `%in%`
filter(flights, UniqueCarrier %in% c("AA", "UA"))
```

1.5.3 select()

Fungsi ini juga ada dalam daftar fungsi dasar R. Coba kita bandingkan dengan perintah dalam dply. Perintah dibawah ini untuk memilih kolom "DepTime", "ArrTime", dan "FlightNum". Ketiga kolom tersebut tidak menerus.

```
# memilih (select) kolom DepTime, ArrTime, and FlightNum columns dengan fungsi dasar R
flights[, c("DepTime", "ArrTime", "FlightNum")]

# bila menggunakan dplyr
select(flights, DepTime, ArrTime, FlightNum)
```

Sekarang bagaimana bila kolom yang akan anda pilih, menerus. Gunakan ":". Dalam perintah berikut ini, kita ingin melihat kolom "Year" hingga "DayofMonth" dan kolom-kolom lain yang mengandung kata-kata "Taxi" dan "Delay".

```
# gunakan simbol colon ":" untuk memilih beberapa kolom yang menerus (bersebelahan) dan gunakan `contain
# anda dapat menggunakan `starts_with`, `ends_with`, dan `matches` untuk menyeleksi nama kolom
select(flights, Year:DayofMonth, contains("Taxi"), contains("Delay"))
```

1.5.4 arrange()

Fungsi ini membantu anda untuk memilih beberapa kolom dan mengurutkan (sorting) set kolom tersebut berdasarkan kolom tertentu. Kali ini kita akan memilih kolom "UniquesCarrier" dan "DepDelay" dan menyortirnya berdasarkan kolom "DepDelay" secara ascending atau dari nilai kecil ke besar.

Urutan kode dengan fungsi dasar R adalah sebagai berikut, yang akan kita bandingkan bila dengan fungsi package `dplyr`:

- pilih data.frame = "flights"
- pilih kolom kriteria = "flights$DepDelay"
- pilih kolom set yang akan anda urutkan (sortir) secara ascending = "UniqueCarrier" dan "DepDelay"

```
# base R approach to select UniqueCarrier and DepDelay columns and sort by DepDelay
flights[order(flights$DepDelay), c("UniqueCarrier", "DepDelay")]

# Dengan urutan yang sama kita pakai teknik dplyr
flights %>%
    select(UniqueCarrier, DepDelay) %>%
    arrange(DepDelay)

# Kalau anda ingin menyortir mengecil ke bawah (secara _descending_), gunakan operator "desc".
flights %>%
    select(UniqueCarrier, DepDelay) %>%
    arrange(desc(DepDelay))
```

1.5.5 mutate()

Dengan fungsi ini anda bisa membuat kolom baru (new column or new variable) berdasarkan suatu fungsi. Bila anda menggunakan fungsi default R, maka urutan kodenya adalah:

- buat data.frame baru = "flights$Speed"
- buat fungsi persamaannya = "flightsDistance/flightsDistance/flightsAirTime*60"
- lihat hasilnya

```
# Bila menggunakan fungsi dasar R, membuat variabel bar "Speed" (in mph)
flights$Speed <- flights$Distance / flights$AirTime*60
flights[, c("Distance", "AirTime", "Speed")]
```

Bila anda menggunakan dplyr:

- pilih data.frame = "flights"
- pilih kolom variabel yang diperlukan = "Distance dan AirTime"
- buat fungsi persamaannya = "Speed = Distance / AirTime*60" yang hasilnya disimpan sebagai variabel yang baru

```
# menggunakan dplyr
flights %>%
    select(Distance, AirTime) %>%
    mutate(Speed = Distance/AirTime*60)

# atau anda bisa mempersingkatnya menjadi:
flights <- flights %>% mutate(Speed = Distance/AirTime*60)
```

1.5.6 summarise()

Fungsi ini untuk melihat ringkasan (summary) dari suatu kolom (variabel) atau grup kolom (group of variables). Sebagai contoh, jika anda ingin menghitung rata-rata delay kedatangan (arrival delay) untuk tiap tujuan (destination), maka alurnya sbb jika menggunakan fungsi default R:

- pilih data.frame = "flights"
- `tapply` kolom yang diperlukan = "ArrDelay" dan "Dest"
- fungsi yang diperlukan = "mean" dan abaikan baris NA = "na.rm=TRUE" atau gunakan "aggregate".

```
# menggunakan fyngsi dasar
head(with(flights, tapply(ArrDelay, Dest, mean, na.rm=TRUE)))

# atau gunakan or you can use "aggregate"
head(aggregate(ArrDelay ~ Dest, flights, mean))
```

Alur dengan dplyr: - pilih data.frame = "flights" - group_by kolom = "Dest" + summarise dengan menghitung "mean" dari kolom "ArrDelay", abaikan baris NA = "na.rm=TRUE" - simpan hasil dalam kolom "avg_delay"

```
# menggunakan dplyr: buat table digrup berdasarkan "Dest" kemudian then `summarise` each group dengan me
flights %>%
    group_by(Dest) %>%
    summarise(avg_delay = mean(ArrDelay, na.rm=TRUE))
```

Anda juga bisa menggunakan fungsi `summarise_each()` untuk lebih dari satu kolom. Hal yang sama untuk fungsi mutate_each().

Alurnya sebagai berikut:

- pilih data.frame = "flights"
- group_by = "UniqueCarrier"
- lakukan summarise_each untuk menghitung "mean" terhadap persentase penerbangan yang "Cancelled" dan "Diverted"

```
# untuk setiap "carrier", hitung persentase penerbangan yang dibatalkan (cancelled) atau diubah rutenya
flights %>%
    group_by(UniqueCarrier) %>%
    summarise_each(funs(mean), Cancelled, Diverted)
```

Contoh lain anda ingin mengetahui nilai min dan max untuk kolom-kolom dengan kata-kata "Delay" untuk tiap perusahaan penerbangan dalam kolom "UniqueCarrier". Alurnya sbb:

- pilih data.frame = "flights"
- group_by = "UniqueCarrier"
- lakukan summarise_each() untuk melihat "min" dan "max" pada kolom yang mengandung kata "Delay"

```
# untuk tiap penerbangan, hitung minimum dan maksimum delay untuk kedatangan dan keberangkatan
flights %>%
    group_by(UniqueCarrier) %>%
    summarise_each(funs(min(., na.rm=TRUE), max(., na.rm=TRUE)), matches("Delay"))
```

1.6 Mengubah format data dengan package "reshape" atau "reshape2"

Format yang dimaksud di sini adalah mengubah format data lebar (*wide*) data menjadi format data panjang (*long*) atau sebaliknya. Dalam bahasa sehari-hari, *reshape* sama dengan *transpose*. Kali ini kita akan menggunakan package `reshape2` sebagai pembaruan dari package `reshape` oleh Hadley Wickham.

Si Pembuat, Hadley Wickham menggunakan prinsip melelehkan ("melt") data frame untuk kemudian dicetak ("cast") dengan bentuk data frame sesuai keinginan kita. Data "dilelehkan" untuk membuat kombinasi id-variabel yang unik untuk setiap barisnya. Ketik beberapa perintah berikut ini.

Bilamana anda mengalami masalah dalam instalasi package `reshape2`, coba gunakan package `reshape`. Tinggal ganti nama `reshape2` menjadi `reshape` dalam kode di bawah ini.

```
# Kita akan menggunakan data air quality yang sudah terinstalasi dalam sistem R
# example of melt function
install.packages("reshape2")
library(reshape2)
```

Berikut ini adalah contoh `wide data format`. Jumlah kolom lebih banyak dibandinkan jumlah baris.

```
#   ozone   wind  temp
# 1 23.62 11.623 65.55
# 2 29.44 10.267 79.10
# 3 59.12  8.942 83.90
# 4 59.96  8.794 83.97
```

Sebaliknya, ini adalah `long data format`. Jumlah kolom lebih sedikit dibadingkan jumlah baris.

```
#    variable  value
# 1     ozone 23.615
# 2     ozone 29.444
# 3     ozone 59.115
# 4     ozone 59.962
# 5      wind 11.623
# 6      wind 10.267
# 7      wind  8.942
# 8      wind  8.794
# 9      temp 65.548
# 10     temp 79.100
# 11     temp 83.903
# 12     temp 83.968
```

Dalam praktek, `long data format` biasanya digunakan untuk data *time series*, sedangkan `wide data format` umum digunakan untuk data *spatial*.

Untuk mengubah format, kita dapat menggunakan `reshape2` package. Fungsi dasarnya ada dua:

- `melt`: mereformat `wide-format` `long-format`;
- `cast`: mengubah `long-format` menjadi `wide-format`.

Kita coba package ini dengan memanfaatkan data `airquality` yang sudah terinstalasi dalam R.

Pertama kita mengubah nama-nama kolom menggunakan `lower case`. Dalam analisis data biasanya memang kita tidak terlalu mengutamakan nama kolom yang panjang dan deskriptif cukup nama pendek terdiri dari beberapa huruf dan (sebaiknya) `lower case`.

```
# mengubah nama kolom dari `sentence case` menjadi `lower case`
names(airquality) <- tolower(names(airquality))
head(airquality)
```

Outputnya...

```
#   ozone solar.r wind temp month day
# 1    41     190  7.4   67     5   1
# 2    36     118  8.0   72     5   2
# 3    12     149 12.6   74     5   3
# 4    18     313 11.5   62     5   4
# 5    NA      NA 14.3   56     5   5
# 6    28      NA 14.9   66     5   6
```

Kita beri nama data frame menjadi `aql`.

```
# renaming data frame to aql
aql <- melt(airquality) # [a]ir [q]uality [l]ong format
# checking data frame
head(aql)
tail(aql)
```

Outputnya `head()` ...

```
#   variable value
# 1    ozone    41
# 2    ozone    36
# 3    ozone    12
# 4    ozone    18
# 5    ozone    NA
# 6    ozone    28
```

Output `tail()`...

```
#     variable value
# 913      day    25
# 914      day    26
# 915      day    27
# 916      day    28
# 917      day    29
# 918      day    30
```

Mengatur `ID variables` atau kolom variable yang digunakan sebagai identitas data.

```
# mengatur ID variables
aql <- melt(airquality, id.vars = c("month", "day"))
head(aql)
```

Outputnya ...

```
#   month day variable value
# 1     5   1    ozone    41
# 2     5   2    ozone    36
# 3     5   3    ozone    12
# 4     5   4    ozone    18
# 5     5   5    ozone    NA
# 6     5   6    ozone    28
```

Kita coba menggunakan fungsi `melt`.

```
aql <- melt(airquality, id.vars = c("month", "day"),
  variable.name = "climate_variable",
  value.name = "climate_value")
head(aql)
```

Outputnya, data frame kita berubah menjadi seperti ini.

```
#   month day climate_variable climate_value
# 1     5   1            ozone           41
# 2     5   2            ozone           36
# 3     5   3            ozone           12
# 4     5   4            ozone           18
# 5     5   5            ozone           NA
# 6     5   6            ozone           28
```

Selanjutnya kita coba fungsi `cast`. Fungsi `cast` punya banyak variasi. Bilamana anda lebih sering menggunakan tipe data frame dalam pekerjaan, maka gunakan fungsi `dcast`. Untuk tipe vector, matrix, atau array, anda dapat menggunakan fungsi `acast`.

```
# mengubah data frame airquality menjadi aql dengan id vars month dan day
aql <- melt(airquality, id.vars = c("month", "day"))

# mengubah data frame aql menjadi aqw
aqw <- dcast(aql, month + day ~ variable)
head(aqw)
```

Outputnya menjadi ...

```
#   month day ozone solar.r wind temp
# 1     5   1    41     190  7.4   67
# 2     5   2    36     118  8.0   72
# 3     5   3    12     149 12.6   74
# 4     5   4    18     313 11.5   62
# 5     5   5    NA      NA 14.3   56
# 6     5   6    28      NA 14.9   66
```

Bandingkan dengan data awal.

```
head(airquality) # original data
```

Output data awal...

```
#   ozone solar.r wind temp month day
# 1    41     190  7.4   67     5   1
# 2    36     118  8.0   72     5   2
# 3    12     149 12.6   74     5   3
# 4    18     313 11.5   62     5   4
# 5    NA      NA 14.3   56     5   5
# 6    28      NA 14.9   66     5   6
```

Coba beberapa baris berikut ini.

One confusing "mistake" you might make is casting a dataset in which there is more than one value per data cell. For example, this time we won't include day as an ID variable:

```
dcast(aql, month ~ variable)
```

Outputnya ...

```
#   month ozone solar.r wind temp
# 1     5    31      31   31   31
# 2     6    30      30   30   30
# 3     7    31      31   31   31
# 4     8    31      31   31   31
# 5     9    30      30   30   30
```

Mestinya akan muncul peringatan: `Aggregation function missing: defaulting to length`.

Coba kode ini untuk `cast` data `aql` dengan ID var month dan agregasi mean, hilangkan data yang mengandung NA.

```
dcast(aql, month ~ variable, fun.aggregate = mean,
  na.rm = TRUE)
```

Outputnya ...

```
#   month ozone solar.r   wind  temp
# 1     5 23.62   181.3 11.623 65.55
# 2     6 29.44   190.2 10.267 79.10
# 3     7 59.12   216.5  8.942 83.90
# 4     8 59.96   171.9  8.794 83.97
# 5     9 31.45   167.4 10.180 76.90
```

Anda akan perlu melakukan beberapa kali googling dan mencoba berbagai variasi kode untuk melihat output yang sesuai dengan kebutuhan.

2 Tautan terkait

- Tutorial data manipulation dari Rutgers Uni
- Situs Cook book R
- Situs Science Nature
- http://www.sr.bham.ac.uk/~ajrs/R/r-manipulate_data.html

- [Video: Hands-on dplyr tutorial for faster data manipulation in R, by: Kevin Markham (@justmarkham)](https://www.youtube.com/watch?v=jWjqLW-u3hc)
- [R code: Hands-on dplyr tutorial for faster data manipulation in R, by: Kevin Markham (@justmarkham)](http://rpubs.com/justmarkham/dplyr-tutorial)
- [Blog: Dataschool.io, by: Kevin Markham (@justmarkham)](http://www.dataschool.io/dplyr-tutorial-for-faster-data-mar
- Blog Sean Anderson
- reshape2 website