STA561 Pemrograman Statistika

Nur Andi Setiabudi

2021-09-08

Contents

Welcome

```
cowsay "Data science is awesome!"
```

```
##  _________________________
## < Data science is awesome! >
##  -------------------------
##         \   ^__^
##          \  (oo)\_______
##             (__)\       )\/\
##                 ||----w |
##                 ||     ||
```

Chapter 1

Pengenalan R

1.1 Apa itu R?

Pada 1976, John Chambers dan tim di Bell Telephone Laboratories (bagian dari AT&T Corp) mengembangkan bahasa pemrograman S sebagai *tools* analisis statistika di internal perusahaan. Awalnya S diimplementasikan sebagai modul yang berjalan pada Fortran. Lalu pada 1988, S ditulis dalam bahasa C (yang merupakan versi ke-3) dan mulai mirip dengan bahasa yang kita kenal sekarang. Versi 4 dari bahasa S yang dirilis tahun 1998 merupakan versi yang kita gunakan sekarang. Meskipun banyak pengembangan, secara fundamental bahasa S tidak mengalami perubahan berarti sejak saat itu.

Salah satu batasan utama bahasa S adalah hanya tersedia dalam paket komersial, S-PLUS. Pada tahun 1991, dengan mengimplementasikan bahas S, R diciptakan oleh Ross Ihaka dan Robert Gentleman di Departemen Statistika di Universitas Auckland. Pada tahun 1993 diumumkan bahwa R dibuat untuk publik. Pada tahun 1995, atas saran dari Martin Mächler, Ross dan Robert mengubah lisensi R menjadi GNU General Public License sehingga menjadikan R perangkat lunak bebas. Ini sangat penting karena memungkinkan kode sumber untuk seluruh sistem R dapat diakses oleh siapa saja.

1.2 Fitur Dasar R

Pada fase awal, fitur utama R adalah sintaksnya sangat mirip dengan S, sehingga memudahkan pengguna S-PLUS untuk beralih menggunakan R. Saat ini, R dapat dijalankan di hampir semua platform komputasi dan sistem operasi. Sifatnya yang terbuka (opensource) membuat siapa pun bebas untuk mengadaptasi perangkat lunak ke platform apa pun yang mereka pilih. Salah satu hal menarik R sebagai perangkat lunak terbuka adalah perilisan fitur baru secara reguler, yang biasanya dilakukan di bulan Oktober.

Fitur utama lain yang dimiliki R adalah kemampuan grafisnya yang canggih. Kemampuan R untuk membuat grafik "kualitas publikasi" telah ada sejak awal dan secara umum lebih baik dibandingkan banyak paket statistik lainnya.

R mempertahankan filosofi bahasa S, yaitu menyediakan bahasa yang berguna untuk pekerjaan secara interaktif, dan juga memungkinkan pengguna untuk mengembangkan alat baru. Artinya pengguna dapat menggunakan R dan menerapkannya ke data, lalu secara perlahan menjadi pengembang yang menciptakan alat baru.

Terakhir, salah satu keunggulan R adalah adanya komunitas aktif dan *supportive* di mana ribuan orang di seluruh dunia telah berkontribusi kepada R baik untuk mengembangkan paket maupun saling membantu menggunakan R untuk berbagai keperluan.

1.3 Sistem R

R terbagi menjadi dua bagian utama, yaitu

- *Base R* yang merupakan perangkat lunak dasar yang berisi bahasa pemrograman R
- Paket/*package*

Paket R dapat dibagi menjadi beberapa bagian, antara lain.

- *Base R* berisi paket `base` yang diperlukan untuk menjalankan R dan berisi fungsi-fungsi paling mendasar,

- Selain itu saat instalasi, disertakan juga paket pendukung lainnya seperti `utils`, `stats`, `datasets` dan lain-lain.
- Paket-paket lainnya dapat ditambahkan setelah instalasi, yang berasal dari
  - Lebih dari 4000 paket di *The Comprehensive R Archive Network* atau CRAN
  - Sejumlah paket termasuk paket dalam pengembangan di repositori GitHub
  - Sumber-sumber lainnya

Chapter 2

Pemrograman R Dasar

2.1 Input

R merupakan bahasa interpreter. Ketika kita memasukkan suatu input pada *console* R (atau menjalankan sebuah *script* R), sebuah program dalam sistem R, dinamakan interpreter, akan mengeksekusi perintah yang kita tulis. R juga bersifat interaktif, artinya setiap perintah yang kita tulis dapat langsung dievaluasi oleh R dan hasilnya dapat ditampilkan pada layar.

Misalnya, dengan memasukkan perintah perkalian berikut pada *console* R:

```
10*2
```

Ketika kita menekan tombol enter, R akan mengeksekusi dan menampilkan hasilnya

```
## [1] 20
```

Console R diawali tanda `>`, yang menunjukkan bahwa R siap menerima perinta baru. Jika kita memasukan perintah yang tidak lengkap, maka tanda tersebut akan berubah menjadi tanda `+`.

Semua perintah atau teks yang ditulis setelah tanda `#` tidak akan dieksekusi oleh R. Biasanya ini berguna untuk memberikan komentar atau catatan

```
# perkalian 10 x 2
10*2
```

```
## [1] 20
```

2.2 Assigment

Dalam R, sangat disarankan untuk menggunakan tanda `<-` sebagai operator *assigment*. `obj <- expr` berarti masukkan nilai hasil dari operasi di sisi kanan (`expr`) ke dalam objek di sisi kiri (`obj`). Misalnya:

```
x <- 20
```

Artinya kita memasukkan nilai `20` ke dalam objek `x`. Contoh lain

```
y <- 100 + 50
```

Artinya kita memasukkan hasi dari operasi `100 + 50` ke dalam objek `y`. Selain dengan operator `<-`, kita juga dapat menggunakan operator `=` atau `->`.

Untuk menampilkan objek dalam layar, cukup tuliskan nama objek lalu enter.

```
x
```

```
## [1] 20
```

```
y
```

```
## [1] 150
```

Atau bisa juga dengan perintah `print()`

```
print(x)
```

```
## [1] 20
```

```
print(y)
```

```
## [1] 150
```

2.3 Penamaan Objek

Segala hal dalam R dipandang sebagai objek, misalnya data, fungsi, dan lain-lain. Objek-objek tersebut dapat "diberi nama" dengan apapun yang kita mau. Pada contoh sebelumnya, kita mempunyai objek dengan nama `x` dan `y`. Meskipun demikian, ada beberapa aturan penamaan objek dalam R yang harus dipenuhi, yaitu:

- Menggunakan kombinasi alfabet (a-z, A-Z), angka (0-9), titik (.) atau underscore (_),
- Hanya dapat diawali oleh alfabet, titik atau underscore dan tidak boleh diawali dengan angka,
- Tidak mengandung spasi, tab atau karakter khusus seperti !, @, #,
- Sebaiknya tidak menggunakan penamaan atau nilai yang sudah digunakan oleh R, seperti `c`, `df`, `rnorm` dan lainnya.

Ketika membuat sebuah program dalam R (atau bahasa pemrograman apapun), disarankan untuk menggunakan penamaan yang lazin dan konsisten, seperti:

- alllowercase: misal `adjustcolor`
- period.separated: misal `plot.new`
- underscore_separated: misal `numeric_version`
- lowerCamelCase: misal `addTaskCallback`
- UpperCamelCase: misal `SignatureMethod`

Note: meskipun diizinkan, penggunaan *undescore* sebaiknya dihindari karena tidak diimplementasikan disemua *engine* S.

R bersifat `case-sensitive` baik dalam penamaan objek maupun isi dari objek tersebut. Artinya huruf kecil dan huruf besar menunjukkan hal berbeda. Dengan demikian, "ABC" berbeda dengan "abc," berbeda dengan "Abc" dan berbeda dengan "AbC" dan seterusnya.

2.4 Working Directory

Sesuai namanya, *working directory* adalah folder atau *directory* di mana kita bekerja. Untuk mengetahui *working directory* kita saat ini, bisa menggunakan perintah

```
getwd()
```

```
## [1] "D:/SSD21/Bookdown/sta561"
```

Untuk mengganti *working directory*, dapat menggunakan perintah

```
setwd("D:/Learning/R")
```

Perhatikan *path* dipisahkan oleh tanda `/`, atau bisa juga dengan tanda `\\`.

```
setwd("D:\\Learning\\R")
```

Untuk mengakses file yang berada dalam *working directory*, kita cukup menuliskan nama filenya saja, misalnya

```
read.csv("dataku.csv")
```

2.5 Objek Data

R mempunyai beberapa jenis mode objek dasar, atau disebut sebagai "atomic" class dari objek, yaitu:

- *character*, misalnya `"ipb"`, `"mahasiswa"`, `"stastika"`
- *numeric*, misalnya `12, 2.3`, `1.2e-2`
- *complex*, misalnya `1.2e6+2i`

- *logical*, misalnya `T`, `F`, `TRUE`, `FALSE`

Objek Angka:

Angka dalam R umumnya diperlakukan sebagai objek numerik (atau angka riil). Artinya, sebuah angka yang terlihat sebagai "1" atau "2," sebetulnya direpresentasikan oleh R sebagai objek numerik, seperti "1.00" atau "2.00." Apabila kita menginginkan objek integer, kita harus menambahkan akhiran `L`. Misal untuk mendapatkan integer `1` harus ditulis `1L`.

2.6 Tipe Objek Data

Terdapat beberapa tipe objek data standar dalam R, yaitu:

- *Vector*: tipe sederhana dari objek data dalam R di mana setiap elemennya mempunyai mode yang sama
- *Factor*: vektor dengan anggota/elemennya berupa kategori
- *Matrix*: vektor yang berdimensi dua yaitu baris dan kolom
- *Array*: tipe objek yang dapat menyimpan data lebih dari dua dimensi
- *Dataframe*: objek yang menyimpan data dalam bentuk tabular (baris dan kolom)
- *List*: vektor dengan anggota/elemennya berupa objek. Mode dari elemen list boleh berbeda-beda

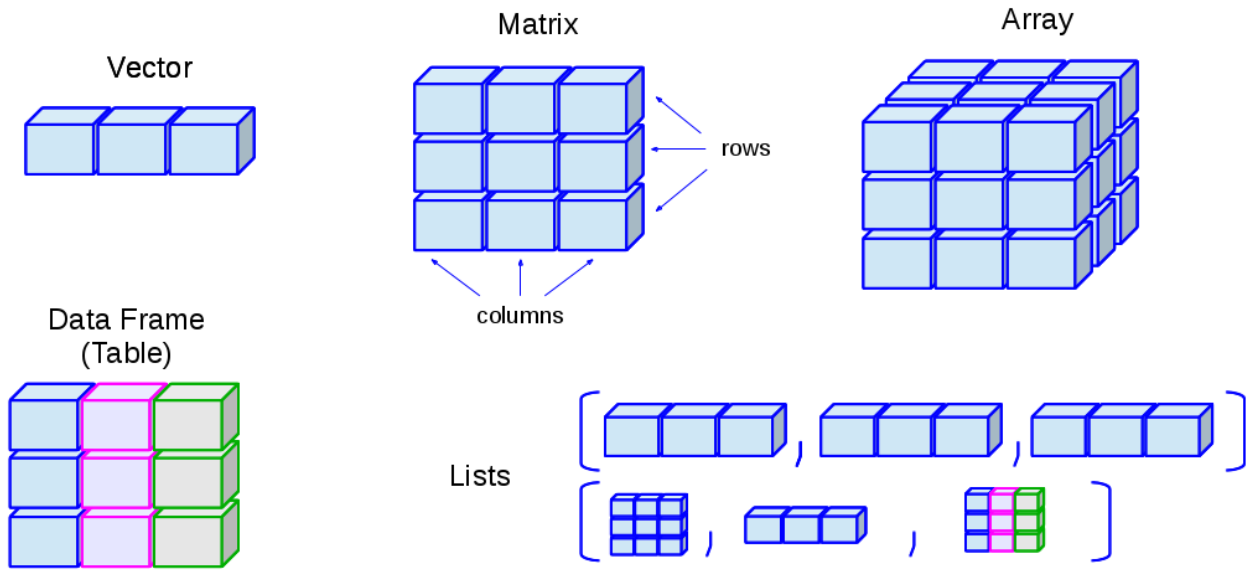

Figure 2.1: Tipe Object R

2.6.1 Vector

Vector merupakan objek data paling sederhana dalam R dan digunakan oleh hampir semua fungsi aritmetik. Dalam vector, mode anggota/elemen adalah sama. Ada beberapa cara membuat vector, di antaranya:

2.6.1.1 Membuat vector

Banyak cara membuat vector. Beberapa di antaranya adalah menggunakan perintah `c()`, `seq()` dan `rep()`.

2.6.1.1.1 Fungsi `c()` Sebuah vektor dapat dibuat dengan fungsi `c()` di mana setiap elemen dipisahkan oleh tanda koma. Misalnya.

```
a <- c(0.5, 0.6)
a
```

```
## [1] 0.5 0.6
```

Contoh lain

```
b <- c(TRUE, FALSE)    ## logical
c <- c(T, F)           ## logical
d <- c("a", "b", "c")  ## character
e <- 9:29              ## integer
f <- c(1+0i, 2+4i)     ## complex
```

Kadang kita memasukkan objek dengan mode berbeda kedalam suatu vektor, baik karena disengaja maupun tidak. Apa yang akan terjadi?

```
a <- c(1.7, "a")  # character
a
```

```
## [1] "1.7" "a"
```

```
b <- c(TRUE, 2)   # numeric
b
```

```
## [1] 1 2
```

```
c <- c("a", TRUE) # character
c
```

```
## [1] "a"    "TRUE"
```

Untuk kasus seperti itu, R akan mengkonversi data kedalam mode yang paling sesuai. Pada contoh pertama, ada dua kemungkinan mode yaitu numeric da character. Karena mengkonversi yang memungkinkan adalah konversi numeric ke character (bukan sebaliknya), maka akan mengkonversi `1.7` menjadi character `"1.7"`.

2.6.1.1.2 Fungsi `seq()`

Fungsi `seq()` digunakan untuk membuat vector yang berisi angka berurutan. Misalnya

Vector 1 sampai dengan 10, dengan *incremental* 1

```
x <- seq(from = 1, to = 10)
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

Atau bisa ditulis dengan perintah berikut

```
x <- 1:10
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

Vector 1 sampai dengan 10, dengan *incremental* 2

```
y <- seq(from = 1, to = 10, by = 2)
y
```

```
## [1] 1 3 5 7 9
```

2.6.1.1.3 Fungsi `rep()`

Fungsi `rep()` digunakan untuk membuat vector dengan mengulang nilai yang diinginak, misalnya

```
x <- rep(1, 10)
x
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```

2.6.1.2 Mengakses element dari vector

Element pada vector dapat diakses melalui indeksnya dengan menggunakan operator `[ ]`. Dua contoh berikut mengambil elemen pertama serta elemen ke-2 dan ke-3 dari vector

```
x <- c(10, 20, 30, 40, 50)
x[1]
```

```
## [1] 10
```

```
x[c(2,3)]
```

```
## [1] 20 30
```

2.6.1.3 Fungsi lain

Fungsi lain sering digunakan dalam vector adalah `length()` dan `class()`. Fungsi `length()` berguna untuk mengetahui pajang atau banyaknye elemen dari suatu vector sedangkan `class()` untuk mengetahi *class* atau mode dari suatu vector.

2.6.2 Factor

Faktor digunakan untuk merepresentasikan data kategorik, baik terurut/*ordered* maupun tidak diurutkan/*unordered*. Faktor dapat dianggap sebagai vektor di mana setiap elemennya memiliki label. Objek faktor dapat dibuat dengan fungsi `factor()`.

```
f <- factor(c("SD", "SMA", "SMP", "SD", "SMA", "SMP", "SD", "SMP"))
f
```

```
## [1] SD  SMA SMP SD  SMA SMP SD  SMP
## Levels: SD SMA SMP
```

```
factor(f, levels = c("SD", "SMP", "SMA"))
```

```
## [1] SD  SMA SMP SD  SMA SMP SD  SMP
## Levels: SD SMP SMA
```

```
factor(f, levels = c("SD", "SMP", "SMA"), ordered = TRUE)
```

```
## [1] SD  SMA SMP SD  SMA SMP SD  SMP
## Levels: SD < SMP < SMA
```

```
length(y)
```

```
## [1] 5
```

```
class(y)
```

```
## [1] "numeric"
```

2.6.3 Matriks

Matriks/*matrix* merupakan vector yang berdimensi dua yaitu baris dan kolom. Matriks dapat dibuat dengan mengubah dimensi dari suatu vector.

Matriks dapat dibentuk dengan perintah `matrix()`. Secara *default*, matriks dibentuk dengan cara *column-wise* (`byrow = FALSE`), yaitu dengan mengisi kolom pertama terlebih dahulu, dari atas ke bawah, dilanjutkan kolom berikutnya.

Misalnya untuk membuat matriks berukuran 2 x 3 :

```
m <- matrix(1:6, nrow = 2, ncol = 3)
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

Atau bisa dengan menambahkan argumen `byrow = TRUE` sehingga akan mengisi baris pertama terlebih dahulu, mulai dari kiri ke kanan, dilanjutknya ke baris berikutnya.

```
m <- matrix(1:6, nrow = 2, ncol = 3, byrow = TRUE)
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

Matriks dapat dibentuk secara langsung dari vector dengan cara menambahkan atribut dimensi.

```
m <- 1:10
dim(m) <- c(2, 5)
m
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    3    5    7    9
## [2,]    2    4    6    8   10
```

Cara lain membentuk matriks adalah dengan penggabungan kolom dengan fungsi `cbind()` dan penggabungan baris dengan fungsi `rbind()`.

```
x <- 1:3
y <- 10:12
cbind(x, y)
```

```
##      x  y
## [1,] 1 10
## [2,] 2 11
## [3,] 3 12
```

```
rbind(x, y)
```

```
##   [,1] [,2] [,3]
## x    1    2    3
## y   10   11   12
```

2.6.4 Array

Array adalah struktur data yang dapat menampung data multidimensi. Dalam R, jika matriks hanya mempunyai 2 dimensi, maka array dapat memiliki lebih dari 2 dimensi.

```
v1 <-  c(5, 10, 15, 20)
v2 <-  c(25, 30, 35, 40, 45, 50, 55, 60)

arr <- array(c(v1, v2), dim =c(4,4,3))
```

Untuk mengetahui dimensi dari suatu array, dapat menggunakan fungsi `dim()`

```
dim(arr)
```

```
## [1] 4 4 3
```

2.6.5 Dataframe

Baris dalam dataframe merepresentasikan pengamatan/observasi, sedangkan kolom mereprentasikan peubah/*variable*. Setiap elemen dalam kolom yang sama mempunyai mode yang sama, namun antar kolom bisa mempunyai mode yang berbeda.

Dataframe dapat dibuat menggunakan fungsi `data.frame()`:

```
df <- data.frame(foo = 1:4, bar = c(T, T, F, F))
df
```

```
##   foo   bar
## 1   1  TRUE
## 2   2  TRUE
## 3   3 FALSE
## 4   4 FALSE
```

```
df2 <- data.frame(numbers = c(10, 20, 30, 40),
                  text = c("a", "b", "c", "a"))
df2
```

```
##   numbers text
## 1      10    a
## 2      20    b
```

```
## 3      30    c
## 4      40    a
```

2.6.6 List

List merupakan bentuk khusus dari vector yang memungkinkan elemennya bisa berupa objek dengan mode yang berbeda-beda. Elemen-elemen dari list dapat berupa vector, matriks, array, list atau gabungan beberapa struktur data.

List dapat dibuat dengan menggunakan fungsi `list()`

```
s <- "A"
v <- c(1:20)
m <- matrix(1:6, nrow = 2, ncol = 3, byrow = TRUE)
df <- data.frame(numbers = c(10, 20, 30, 40),
                 text = c("a", "b", "c", "a"))

l <- list(s, v, m, df)
l
```

```
## [[1]]
## [1] "A"
##
## [[2]]
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##
## [[3]]
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
##
## [[4]]
##   numbers text
## 1      10    a
## 2      20    b
## 3      30    c
## 4      40    a
```

2.7 Missing value

Ada beberapa *missing value* dalam R, yaitu:

- NULL

  Sebuah objek yang diperoleh ketika suatu ekspresi atau fungsi menghasilkan nilai yang tidak terdefinisi *(undefined value)*

- NA

  Singkatan dari "Not Available." Merupakan sebuah logical untuk mengindikasikan *missing value*.

- NaN

  Singkatan dari "Not a Number." Merupakan sebuah logical untuk angka dan merupakan gambaran imajiner dari nilai nilai yang sangat kompleks.

- Inf / -Inf

  Singkatand ari *infinity* atau tidak hingga. Merupakan angka yang sangat besar atau sangat kecil.

```
x <- c(1, 2, NA, 10, 3)
is.na(x)
```

```
## [1] FALSE FALSE  TRUE FALSE FALSE
```

```
is.nan(x)
```

```
## [1] FALSE FALSE FALSE FALSE FALSE
```

```
x <- c(1, 2, NaN, NA, 4)
is.nan(x)
```

```
## [1] FALSE FALSE  TRUE FALSE FALSE
```

2.8 Penamaan Elemen

Objek R dapat mempunyai nama. Demikian juga dengan setiap elemen dalam sebuah objek data. Hal ini sangat berguna ketika menuliskan kode dan menjelaskan objek. Untuk memberikan nama bagi elemen-elemen dari vector, dapat menggunakan fungsi `names()`

```
x <- 1:3
names(x)
```

```
## NULL
```

```
names(x) <- c("New York", "Seattle", "Los Angeles")
x
```

```
##    New York     Seattle Los Angeles
##           1           2           3
```

```
names(x)
```

```
## [1] "New York"    "Seattle"     "Los Angeles"
```

Cara yang sama untuk list

```
names(l)
```

```
## NULL
```

```
names(l) <- c("teks", "vektor", "matriks", "tabel")
l
```

```
## $teks
## [1] "A"
##
## $vektor
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##
## $matriks
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
##
## $tabel
##   numbers text
## 1      10    a
## 2      20    b
## 3      30    c
## 4      40    a
```

```
names(l)
```

```
## [1] "teks"    "vektor"  "matriks" "tabel"
```

Matriks dapat mempunyai nama kolom dan barisnya dengan menggunakan fungsi `dimnames()`

```
m <- matrix(1:4, nrow = 2, ncol = 2)
dimnames(m) <- list(c("a", "b"), c("c", "d"))
m
```

```
##   c d
## a 1 3
## b 2 4
```

Penamaan kolom dan baris pada matriks bisa dilakukan terpisah menggunakan fungsi `colnames()` dan `rownames()`

```
colnames(m) <- c("h", "f")
rownames(m) <- c("x", "z")
m
```

```
##   h f
## x 1 3
## z 2 4
```

Seperti halnya matriks, kolom dan baris pada dataframe juga dapat diberikan nama dengan menggunakan fungsi `names()` dan `rownames()`. Perhatikan ada perbedaan fungsi yang digunakan.

```
a <- c(10, 20, 30, 40)
b <- c("a", "b", "c", "a")
df <- data.frame(a, b)
df
```

```
##    a b
## 1 10 a
## 2 20 b
## 3 30 c
## 4 40 a
```

```
names(df) <- c("numbers", "chars")
row.names(df) <- c("a", "b", "c", "d")

df
```

```
##   numbers chars
## a      10     a
## b      20     b
## c      30     c
## d      40     a
```

Note: Ketika membuat dataframe, R akan memberikan nama untuk kolom-kolom yang terbentuk. Hanya saja kadang nama yang diberikan tidak sesuai dengan apa yang kita inginkan.

```
df2 <- data.frame(c(10, 20, 30, 40),
                  c("a", "b", "c", "a"))
names(df2)
```

```
## [1] "c.10..20..30..40."     "c..a....b....c....a.."
```

2.9 Struktur Kendali

2.9.1 Percabangan

Pemilihan atau percabangan merupakan bagian penting dalam programming. Dalam R, hal ini dapat dilakukan dengan menggunakan perintah `if ... else`.

if

Statement akan dieksekusi jika expression benar. Jika expression salah, maka tidak ada yang dieksekusi.

```
# https://www.datamentor.io/r-programming
if (expression) {
   statement
}
```

Contoh

```
x <- 5
if(x > 0){
   print("Positive")
}
```

```
## [1] "Positive"
```

if ... else

Jika expression benar, maka statement1 akan dieksekusi. Jaka salah maka statement2 akan dieksekusi.

```
if (expression) {
   statement1
} else {
   statement2
}
```

Contoh

```
x <- -5
if(x > 0){
   print("Non Negative")
} else {
   print("Negative")
}
```

```
## [1] "Negative"
```

if ... else if ... else

Jika expression benar, maka statement1 akan dieksekusi. Jaka salah maka akan periksa expression2. Jika bener maka statement2 akan dieksekusi. Jika salah maka statement3 akan dieksekusi

```
if ( expression) {
   statement1
} else if ( expression2) {
   statement2
} else {
   statement3
}
```

Contoh:

```
x <- -5
if(x == 0){
   prin("Zero")
} else if(x < 0){
   print("Negative")
} else {
   print("Positive")
}
```

```
## [1] "Negative"
```

2.9.2 Loop

Loop berfungsi untuk mengulang perintah atau blok perintah. Dalam R, ada beberapa fungsi perintah looping, yaitu `for`, `while`, `repeat`

for loop

For loop digunakan untuk mengiterasi vektor

```
for (val in sequence) {
   statement
}
```

Contoh mengiterasi vektor dan menjumlahkan angka genap

```
x <- c(2,5,3,9,8,11,6)
count <- 0
```

```
for (val in x) {
   if(val %% 2 == 0) {
      count = count+1
   }
}

print(count)
```

```
## [1] 3
```

while loop

While loop digunakan untuk melakukan iterasi selama kondisi/expression tertentu terpenuhi.

```
while (expression)
{
   statement
}
```

Contoh

```
i <- 1
while (i < 6) {
   print(i)
   i = i+1
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

break dan next

Perintah `break` gigunakan di dalam loop untuk menghentikan iterasi. Sedangkan perintah `next` digunakan untuk melewati sebuah iterasi.

```
# break
if (expression) {
   break
}

# next
if (expression) {
   next
}
```

Contoh

```
# break, berhenti pada iterasi ke-3
x <- 1:5
for (val in x) {
   if (val == 3){
      break
      }
   print(val)
}
```

```
## [1] 1
## [1] 2
```

```
# next, skip iterasi ke-3
x <- 1:5
for (val in x) {
```

```
   if (val == 3){
      next
      }
   print(val)
}
```

```
## [1] 1
## [1] 2
## [1] 4
## [1] 5
```

repeat loop

Repeat loop digunakan untuk iterasi blok perintah berulang kali. Dalam repeat, tidak ada kondisi untuk keluar dari loop. Untuk itu, kita harus menggunakan perintah `break` secara eksplisit atau looping akan terus berjalan (infinite loop)

```
repeat {
   statement
}
```

Contoh

```
x <- 1
repeat {
   print(x)
   x = x+1
   if (x == 6){
      break
   }
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

Chapter 3

Operasi Dasar R

3.1 Akses Elemen

Ada tiga operator yang dapat digunakan untuk mengekstrak/mengakses elemen atau bagian dari objek R.

- Operator `[ ]` selalu mengembalikan objek dari kelas yang sama dengan aslinya. Dapat digunakan untuk memilih satu atau beberapa elemen dari suatu objek.
- Operator `[[ ]]` digunakan untuk mengekstrak elemen dari list atau dataframe. Hanya dapat digunakan untukmengekstrak satu elemen dan kelas objek yang dikembalikan tidak harus sama seperti objek awalnya.
- Operator `$` digunakan untuk mengekstrak elemen list atau dataframe melalui namanya. Secara semantik, ini mirip dengan operator `[[ ]]`.

3.1.1 Akses elemen vector

Elemen vector dapat diekstrak dengan memasukkan nomor urut elemen ke dalam operator `[ ]`. Elemen dari vektor dan objek R lainnya, dimulai dari ‘.

```
x <- c("a", "b", "c", "c", "d", "a")
```

Mengakses elemen pertama

```
x[1]
```

```
## [1] "a"
```

Mengakses elemen ke-2

```
x[2]
```

```
## [1] "b"
```

Mengakses semua elemen kecuali elemen ke-2

```
x[-2]
```

```
## [1] "a" "c" "c" "d" "a"
```

Jika vector sudah mempunyai nama, dapat diakses menggunakan namanya

```
y <- 1:3
names(y) <- c("New York", "Seattle", "Los Angeles")

y["Seattle"]
```

```
## Seattle
##       2
```

Operator `[ ]` dapat digunakan untuk mengakses beberapa elemen sekaligus, misalnya untuk mengekstrak element pertama sampai ke-4

```
x[1:4]
```

```
## [1] "a" "b" "c" "c"
```

Mengkases elemen ke-1, ke-2 dan ke-4

```
x[c(1,2,4)]
```

```
## [1] "a" "b" "c"
```

Selain dengan integer, memilih elemen juga bisa menggunakan logical. Misalnya untuk memilih elemen bukan "a"

```
u <- x != "a"
u
```

```
## [1] FALSE  TRUE  TRUE  TRUE  TRUE FALSE
```

```
x[u]
```

```
## [1] "b" "c" "c" "d"
```

Atau dapat diringkas

```
x[x != "a"]
```

```
## [1] "b" "c" "c" "d"
```

3.1.2 Akses elemen matriks

Sepertihalnya vector, akses terhadap elemen matriks dapat dilakukan dengan operator `[ ]` dengan memasukkan posisi baris dan kolom dengan format `[row, col]`. Sehingga apabila akan mengambil elemen di baris ke-2 kolom ke-1 dan baris ke-1 kolom ke-3 dapat kita tuliskan:

```
x <- matrix(1:6, 2, 3)
x
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
x[2,1] # baris ke-2 kolom ke-1
```

```
## [1] 2
```

```
x[1,3] # baris ke-1 kolom ke-3
```

```
## [1] 5
```

Atau untuk mengekstrak seluruh kolom atau baris tertentu

```
x[2,] # ekstrak baris ke-2
```

```
## [1] 2 4 6
```

```
x[,3] # ekstrak kolom ke-3
```

```
## [1] 5 6
```

3.1.3 Akses elemen list

Elemen dari list dapat diakses dengan menggunakan tiga operator di atas dengan tujuan yang berbeda-beda.

```
x <- list(foo = 1:4, bar = 0.6, foobar = c("a","b","c"))
x
```

```
## $foo
## [1] 1 2 3 4
##
## $bar
## [1] 0.6
```

```
##
## $foobar
## [1] "a" "b" "c"
```

Akses list dengan `[ ]` sama seperti vektor

```
x[1]   # elemen pertama
```

```
## $foo
## [1] 1 2 3 4
```

```
x[1:2] # elemen pertama dan kedua
```

```
## $foo
## [1] 1 2 3 4
##
## $bar
## [1] 0.6
```

Untuk akses elemen tunggal, dapat menggunakan operator `[[ ]]`

```
x[[2]]  # akses elemen ke-2
```

```
## [1] 0.6
```

```
x[["bar"]]  # akses elemen yang bernama "bar"
```

```
## [1] 0.6
```

Untuk mengakses elemen dalam elemen:

```
x[[3]][[1]]
```

```
## [1] "a"
```

Atau menggunakan operator `$`

```
x$bar
```

```
## [1] 0.6
```

```
x$data
```

```
## NULL
```

Perhatikan tidak ada elemen bernama "data," sehingga R mengembalikan "NULL," bukan *error*.

3.1.4 Akses elemen dataframe

Akses elemen data frame mirip seperti matriks dengan menggunakan operator `[ ]`

```
df <- data.frame(numbers = c(10, 20, 30, 40),
                 text = c("a", "b", "c", "a"),
                 logic = c(T, F, T, F))
df
```

```
##   numbers text logic
## 1      10    a  TRUE
## 2      20    b FALSE
## 3      30    c  TRUE
## 4      40    a FALSE
```

```
df[1,2] # baris pertama kolom ke-2
```

```
## [1] "a"
```

```
df[1,]  # baris pertama
```

```
##   numbers text logic
## 1      10    a  TRUE
```

```
df[,2]  # kolom ke-2
```

```
## [1] "a" "b" "c" "a"
```

```
df[df[1] < 30, ] # semua kolom dan semua baris yang lebih kecil dari 20
```

```
##   numbers text logic
## 1      10    a  TRUE
## 2      20    b FALSE
```

Atau dengan operator [[ ]]

```
df[[2]]  # kolom ke-2
```

```
## [1] "a" "b" "c" "a"
```

```
df[["text"]]  # kolom "text"
```

```
## [1] "a" "b" "c" "a"
```

Atau dengan operator $

```
df$text
```

```
## [1] "a" "b" "c" "a"
```

3.2 Operasi aritmatika dasar

3.2.1 Menampilkan atribut

Objek R biasanya mempunyai atribut, seperti

- names, dimnames
- dimensions
- class (e.g. integer, numeric)
- length
- dan lain-lain

Misalnya kita mempunyai data frame

```
df <- data.frame(numbers = c(10, 20, 30, 40),
                 text = c("a", "b", "c", "a"),
                 logic = c(T, F, T, F))
```

```
names(df)  # nama dari kolom
```

```
## [1] "numbers" "text"    "logic"
```

```
dim(df)    # dimensi dari df
```

```
## [1] 4 3
```

```
nrow(df)   # jumlah kolom
```

```
## [1] 4
```

```
ncol(df)   # jumlah kolom
```

```
## [1] 3
```

```
class(df)  # class objek
```

```
## [1] "data.frame"
```

```
x <- df[[1]]
length(x)  # jumlah elemen
```

```
## [1] 4
```

Untuk mengetahui atribut apasaja yang ada data objek kita, dapat menggunakan perintah `attributes()`.

```
attributes(df)
```

```
## $names
## [1] "numbers" "text"    "logic"
##
## $class
## [1] "data.frame"
##
## $row.names
## [1] 1 2 3 4
```

3.2.2 Operasi pada vector

Operasi-operasi pada vector dilakukan secara element by element (elementwise). Misalnya

```
x <- c(1:10)
y <- c(11:20)

x + y
```

```
##  [1] 12 14 16 18 20 22 24 26 28 30
```

Maka elemen pertama dari `x` akan dijumlahkan dengan elemen pertama dari `y`, elemen ke-2 dari `x` akan dijumlahkan dengan elemen ke-2 dari `y`, dan seterusnya.

Jika vector-vector yang dioperasikan memiliki panjang berbeda, maka berlalu aturan *recycling*, yaitu vektor dengan elemen sedikit akan diulang mengikuti vektor yang memiliki elemen paling banyak. Contoh

```
x <- c(1:10)
y <- c(11:18)

x + y
```

```
## Warning in x + y: longer object length is not a multiple of shorter object
## length
```

```
##  [1] 12 14 16 18 20 22 24 26 20 22
```

Objek `x` mempunyai 10 elemen sedangkan `y` hanya ada 8. Untuk penjumlahan elemen 1 sd. 8, berlaku normal seperti contoh sebelumnya, sedangkan untuk elemen 9 dan 10 menggunakan aturan *recycling*. Dalam hal ini, R akan me-*recycle* elemen pertama dan ke-2 dari `y` sebagai objek "pengganti" bagi elemen ke-9 dan 10.

3.2.2.1 Operasi sederhana vector numerik

R mengenal banyak sekali operasi numerik, seperti

- `+ - * /` : Penjumlahan, pengurangan, perkalian, pembagian
- `%%` : Modulus
- `%/%` : Pembagian integer
- `%*%` : Perkalian matriks setara `x'x`
- `%o%` : Perkalian matriks setara `xx'`
- `< <= > >= == !=` : Operasi logika/perbandingan

Contoh

```
x <- c(1:10)
y <- c(11:20)

x + y  # penjumlah
```

```
##  [1] 12 14 16 18 20 22 24 26 28 30
```

```
x < y  # logical
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
y %% 3   # modulus
```

```
##  [1] 2 0 1 2 0 1 2 0 1 2
```

```
y %/% 3  # pembagian integral
```

```
##  [1] 3 4 4 4 5 5 5 6 6 6
```

```
x %*% y  # Perkalian matriks setara `x'x`
```

```
##      [,1]
## [1,]  935
```

```
x %o% y  # Perkalian matriks setara `xx'`
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]   11   12   13   14   15   16   17   18   19    20
##  [2,]   22   24   26   28   30   32   34   36   38    40
##  [3,]   33   36   39   42   45   48   51   54   57    60
##  [4,]   44   48   52   56   60   64   68   72   76    80
##  [5,]   55   60   65   70   75   80   85   90   95   100
##  [6,]   66   72   78   84   90   96  102  108  114   120
##  [7,]   77   84   91   98  105  112  119  126  133   140
##  [8,]   88   96  104  112  120  128  136  144  152   160
##  [9,]   99  108  117  126  135  144  153  162  171   180
## [10,]  110  120  130  140  150  160  170  180  190   200
```

3.2.2.2 Operasi sederhana vector karakter

R juga mempunyai banyak fungsi untuk operasi terhadap vektor karakter, beberapa diantaranya

- `nchar()` : Menghitung panjang karakter
- `paste()` : Menggabungkan elemen
- `substr()` : Mengambil bagian dari teks berdasarkan posisi tertentu

Contoh:

```
y <- c("Institut", "Pertanian", "Bogor", "IPB")

nchar(y)  # menghitung panjang karakter
```

```
## [1] 8 9 5 3
```

```
paste(y, collapse = " ")  # menggabungkan elemen
```

```
## [1] "Institut Pertanian Bogor IPB"
```

```
paste(y, "ku", sep = "")  # menggabungkan dengan vektor lain
```

```
## [1] "Institutku"  "Pertanianku" "Bogorku"     "IPBku"
```

```
substr(y, 1, 3) # mengambil huruf pertama sampai huruf ke-3
```

```
## [1] "Ins" "Per" "Bog" "IPB"
```

3.3 Operasi pada matriks

R dilengkapi banyak fungsi untuk matriks. Beberapa diataranya: * `*` : Perkalain element by element * `t()` : Transpose * `%*%` : Perkalian matriks setara `x'x` * `%o%` : Perkalian matriks setara `xx'` * `solve()` : Menghitung matriks inverse * `eigen()` : Menghitung eigen value dan eigen vector

Contoh

```
Z1 <- matrix(1:6,2,3)
Z2 <- matrix(1:6,3,2,byrow=T)
Z3 <- matrix(6:9,2,2)
```

```
Z4 <- Z1 %*% Z2
Z4
```

```
##      [,1] [,2]
## [1,]   35   44
## [2,]   44   56
```

```
Z1 %o% Z2
```

```
## , , 1, 1
##
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
##
## , , 2, 1
##
##      [,1] [,2] [,3]
## [1,]    3    9   15
## [2,]    6   12   18
##
## , , 3, 1
##
##      [,1] [,2] [,3]
## [1,]    5   15   25
## [2,]   10   20   30
##
## , , 1, 2
##
##      [,1] [,2] [,3]
## [1,]    2    6   10
## [2,]    4    8   12
##
## , , 2, 2
##
##      [,1] [,2] [,3]
## [1,]    4   12   20
## [2,]    8   16   24
##
## , , 3, 2
##
##      [,1] [,2] [,3]
## [1,]    6   18   30
## [2,]   12   24   36
```

```
Z3 * Z4
```

```
##      [,1] [,2]
## [1,]  210  352
## [2,]  308  504
```

```
invZ <- solve(Z4) # invers
invZ
```

```
##           [,1]      [,2]
## [1,]  2.333333 -1.833333
## [2,] -1.833333  1.458333
```

```
invZ %*% Z4  # matriks identitas
```

```
##      [,1]         [,2]
## [1,]    1 2.842171e-14
## [2,]    0 1.000000e+00
```

```
h <- c(5,11)
p <- solve(Z4,h) #solusi persamaan linear Zp=h


e <- eigen(Z4) #eigen value & eigen vector dr Z4
e$values #akses eigen values
```

```
## [1] 90.7354949  0.2645051
```

```
e[[2]] #akses eigen vectors
```

```
##           [,1]       [,2]
## [1,] 0.6196295 -0.7848945
## [2,] 0.7848945  0.6196295
```

3.4 Latihan

3.4.1 Latihan 1

Tentukan output syntax program berikut:

```
c("la","ye")[rep(c(1,2,2,1),times=4)]
c("la","ye")[rep(rep(1:2,each=3),2)]
```

Jawab:

```
c("la","ye")[rep(c(1,2,2,1),times=4)]
```

```
##  [1] "la" "ye" "ye" "la" "la" "ye" "ye" "la" "la" "ye" "ye" "la" "la" "ye" "ye"
## [16] "la"
```

```
c("la","ye")[rep(rep(1:2,each=3),2)]
```

```
##  [1] "la" "la" "la" "ye" "ye" "ye" "la" "la" "la" "ye" "ye" "ye"
```

3.4.2 Latihan 2

Buatlah syntax agar dihasilkan output vektor sebagai berikut

```
X1 Y2 X3 Y4 X5 Y6 X7 Y8 X9 Y10
1  4  7  10 13 16 19 22 25 28
```

3.4.3 Latihan 3

Seorang peneliti merancang sebuah perancangan percobaan RAKL dengan 4 perlakuan dan 3 kelompok (anggaplah respon percobaan berupa baris bilangan). Bantulah peneliti tersebut untuk membuat raw data seperti output sebagai berikut!

```
> data1

   Perl Kel Resp
1    P1   1  1
2    P1   2  3
3    P1   3  5
4    P2   1  7
5    P2   2  9
6    P2   3  11
7    P3   1  13
8    P3   2  15
9    P3   3  17
10   P4   1  19
11   P4   2  21
12   P4   3  23
```

Jawab

```
jPerl <- 4
jKel <-  3
Perl <- factor(rep(paste0("P", c(1:jPerl)), each = jKel))
Kel <- factor(rep(1:jKel, jPerl))
Resp <- 2*seq(jPerl*jKel) - 1
data1 <- data.frame(Perl, Kel, Resp)
data1
```

```
##    Perl Kel Resp
## 1    P1   1    1
## 2    P1   2    3
## 3    P1   3    5
## 4    P2   1    7
## 5    P2   2    9
## 6    P2   3   11
## 7    P3   1   13
## 8    P3   2   15
## 9    P3   3   17
## 10   P4   1   19
## 11   P4   2   21
## 12   P4   3   23
```

Atau bisa dibuat fungsi sebagai berikut

```
genRancob <- function(jPerl = 4, jKel = 3){
  Perl <-factor(rep(paste0("P", c(1:jPerl)), each = jKel))
  Kel <- factor(rep(1:jKel, jPerl))
  Resp <- 2*seq(jPerl*jKel) - 1
  data1 <- data.frame(Perl, Kel, Resp)
  return(data1)
}

data1 <- genRancob(jPerl = 4, jKel = 3)
data1
```

Chapter 4

Data Wrangling

Data wrangling, disebut juga *data munging* atau *data manipulation* (dalam konotasi positif), merupakam proses transformasi atau menyiapkan data menjadi format siap dianalisis. Banyak tantangan yang dihadapi dalam tahapan ini, mulai dari ukuran data yang besar, format yang beragam, sumber yang tidak terintegrasi dan lain-lain. Sehingga tidak heran jika *data wrangling* menghabiskan hingga 80% dari waktu keseluruhan analsis yang kita lakukan.

Aktivitas utama dalam *data wrangling* di antaranya adalah:

- Membuat kolom baru, biasanya diturunkan dari kolom yang sudah ada
- Subsetting data atau memilih baris dan/kolom tertentu dari data
- Sorting atau mengurutkan data
- Recoding atau mengkodekan ulang nilai-nilai dari data
- Merging data atau menggabungkan data, baik penggabungan baris maupun kolom
- Reshaping data atau mengubah format menjadi bentuk *wide* ataupun *long*

Pada bagian ini akan dipraktikan bagaimana melakukan *data wrangling* menggunakan paket `base` atau paket-paket bawaan R lainnya. Adapun data yang akan digunakan adalah sebagai berikut:

```
employees <- data.frame(ID = c(1,2,3,5,6,7),
                        Name = c("Alex", "Joni", "Banu", "Ani", "Riska", "John"),
                        Age = c(21,27,18,25,22,27),
                        Sex = c("M","M","M", "F", "F","M"))
employees
```

```
##   ID  Name Age Sex
## 1  1  Alex  21   M
## 2  2  Joni  27   M
## 3  3  Banu  18   M
## 4  5   Ani  25   F
## 5  6 Riska  22   F
## 6  7  John  27   M
```

```
more.employees <- data.frame(ID = c(11,12,13),
                             Name = c("Bunga", "Kembang", "Puspa"),
                             Age = c(25,27,21),
                             Sex = c("M","M","M"))
more.employees
```

```
##   ID    Name Age Sex
## 1 11   Bunga  25   M
## 2 12 Kembang  27   M
## 3 13   Puspa  21   M
```

```
address <- data.frame(ID = c(1,2,3,5,6,7),
                      City = c("Bandung","Jakarta","Bogor", "Jakarta", "Bandung", "Jakarta"))
address
```

```
##   ID    City
```

```
## 1  1 Bandung
## 2  2 Jakarta
## 3  3   Bogor
## 4  5 Jakarta
## 5  6 Bandung
## 6  7 Jakarta
```

4.1 Inspeksi dataframe

Selelum melakukan *data wranglingk* lebih lanjut, hal utama yang dikerjakan ada inspeksi terhadap dataframe, diantaranya menampilkan jumlah baris dan kolom, melihat statistik ringkasan, melihat struktur dataframe serta melihat beberapa baris data (baik baris teratas maupun terbawah)

```
# menghitung jumlah baris
nrow(employees)
```

```
## [1] 6
```

```
# menghitung jumlah baris
ncol(employees)
```

```
## [1] 4
```

```
# menghitung dimensi (baris dan kolom)
dim(employees)
```

```
## [1] 6 4
```

```
# menghitung statistik ringkasan
summary(employees)
```

```
##        ID           Name                Age            Sex
##  Min.   :1.00   Length:6           Min.   :18.00   Length:6
##  1st Qu.:2.25   Class :character   1st Qu.:21.25   Class :character
##  Median :4.00   Mode  :character   Median :23.50   Mode  :character
##  Mean   :4.00                      Mean   :23.33
##  3rd Qu.:5.75                      3rd Qu.:26.50
##  Max.   :7.00                      Max.   :27.00
```

```
# melihat struktur dataframe
str(employees)
```

```
## 'data.frame':    6 obs. of  4 variables:
##  $ ID  : num  1 2 3 5 6 7
##  $ Name: chr  "Alex" "Joni" "Banu" "Ani" ...
##  $ Age : num  21 27 18 25 22 27
##  $ Sex : chr  "M" "M" "M" "F" ...
```

```
# menampilkan beberapa baris teratas
head(employees)
```

```
##   ID  Name Age Sex
## 1  1  Alex  21   M
## 2  2  Joni  27   M
## 3  3  Banu  18   M
## 4  5   Ani  25   F
## 5  6 Riska  22   F
## 6  7  John  27   M
```

```
# menampilkan beberapa baris terbawah
tail(employees)
```

```
##   ID  Name Age Sex
## 1  1  Alex  21   M
## 2  2  Joni  27   M
## 3  3  Banu  18   M
```

```
## 4  5   Ani  25   F
## 5  6 Riska  22   F
## 6  7  John  27   M
```

4.2 Mengakses elemen

Mengakses atau memilih sebagian elemen dari dataframe dapat menggunakan operator `[ ]`, `[[ ]]` dan `$`

```
# baris 1, kolom 2
employees[1,2]
```

```
## [1] "Alex"
```

```
# baris 1, kolom "Name"
employees[1,"Name"]
```

```
## [1] "Alex"
```

```
# baris 1, kolom "Name"
employees[1,]$Name
```

```
## [1] "Alex"
```

```
# baris ke-1, semua kolom
employees[1,]
```

```
##   ID Name Age Sex
## 1  1 Alex  21   M
```

```
# kolom ke-2, semua baris
employees[,2]
```

```
## [1] "Alex"  "Joni"  "Banu"  "Ani"   "Riska" "John"
```

```
# kolom "Name", semua baris
employees[,"Name"]
```

```
## [1] "Alex"  "Joni"  "Banu"  "Ani"   "Riska" "John"
```

```
# kolom "Name", semua baris
employees$Name
```

```
## [1] "Alex"  "Joni"  "Banu"  "Ani"   "Riska" "John"
```

```
# baris 1-2, semua kolom
employees[1:2,]
```

```
##   ID Name Age Sex
## 1  1 Alex  21   M
## 2  2 Joni  27   M
```

```
# kolom 1-2, semua baris
employees[,1:2]
```

```
##   ID  Name
## 1  1  Alex
## 2  2  Joni
## 3  3  Banu
## 4  5   Ani
## 5  6 Riska
## 6  7  John
```

```
# kolom 1-2, semua baris
employees[,c(1, 2)]
```

```
##   ID  Name
## 1  1  Alex
## 2  2  Joni
```

```
## 3  3  Banu
## 4  5   Ani
## 5  6 Riska
## 6  7  John
```

```
# kolom "ID" dan "Name", semua baris
employees[,c("ID", "Name")]
```

```
##   ID  Name
## 1  1  Alex
## 2  2  Joni
## 3  3  Banu
## 4  5   Ani
## 5  6 Riska
## 6  7  John
```

```
# semua kolom, hanya baris yang memenuhi kriteria
employees[employees$Age > 20, ]
```

```
##   ID  Name Age Sex
## 1  1  Alex  21   M
## 2  2  Joni  27   M
## 4  5   Ani  25   F
## 5  6 Riska  22   F
## 6  7  John  27   M
```

4.3 Mengubah data

Dimungkinkan untuk mengubah/*mengupdate* nilai dari dataframe (misalnya jika diketahui ada kesalahan pencatatan)

```
# Mengubah data di baris ke-3 kolom "Age"
employees[3,"Age"] <- 29
employees
```

```
##   ID  Name Age Sex
## 1  1  Alex  21   M
## 2  2  Joni  27   M
## 3  3  Banu  29   M
## 4  5   Ani  25   F
## 5  6 Riska  22   F
## 6  7  John  27   M
```

4.4 Mengurutkan baris

Mengurutkan baris dapat dilakukan secara *ascending* (dari kecil ke besar, atau A-Z) atau *descending* (dari kecil ke besar, atau Z-A).

```
# mengurutkan ascending
employees[order(employees$Age),]
```

```
##   ID  Name Age Sex
## 1  1  Alex  21   M
## 5  6 Riska  22   F
## 4  5   Ani  25   F
## 2  2  Joni  27   M
## 6  7  John  27   M
## 3  3  Banu  29   M
```

```
# mengurutkan descending
employees[order(employees$Age, decreasing=T),]
```

```
##   ID  Name Age Sex
## 3  3  Banu  29   M
```

```
## 2  2  Joni  27   M
## 6  7  John  27   M
## 4  5   Ani  25   F
## 5  6 Riska  22   F
## 1  1  Alex  21   M
```

4.5 Menggabungkan dataframe

Penggabungan dataframe dapat dilakukan secara column-wise (merge) ataupun row-wise (union). Untuk merge, bisa dilakukan dengan perintah `merge()` (berdasarkan ID tertentu), atau `cbind()` (berdasarkan urutan baris). Untuk penggabungan union, bisa menggunakan `rbind()`

```
# menggabungkan kolom berdasarkan ID
merge(employees, address, by="ID")
```

```
##   ID  Name Age Sex    City
## 1  1  Alex  21   M Bandung
## 2  2  Joni  27   M Jakarta
## 3  3  Banu  29   M   Bogor
## 4  5   Ani  25   F Jakarta
## 5  6 Riska  22   F Bandung
## 6  7  John  27   M Jakarta
```

```
# menggabungkan kolom berdardasarkan urutan baris
Salary <- c(100, 120, 110, 90, 130, 120)
cbind(employees, Salary)
```

```
##   ID  Name Age Sex Salary
## 1  1  Alex  21   M    100
## 2  2  Joni  27   M    120
## 3  3  Banu  29   M    110
## 4  5   Ani  25   F     90
## 5  6 Riska  22   F    130
## 6  7  John  27   M    120
```

```
# menggabungkan baris berdasarkan urutan kolom
all.employees <- rbind(employees, more.employees)
all.employees
```

```
##   ID    Name Age Sex
## 1  1    Alex  21   M
## 2  2    Joni  27   M
## 3  3    Banu  29   M
## 4  5     Ani  25   F
## 5  6   Riska  22   F
## 6  7    John  27   M
## 7 11   Bunga  25   M
## 8 12 Kembang  27   M
## 9 13   Puspa  21   M
```

4.6 Agregasi data

Menghitung agregasi data

```
# menghitung rata-rata age
mean(all.employees$Age)
```

```
## [1] 24.88889
```

```
# menghitung jumlah employees berdasarkan jenis kelamin
aggregate(all.employees$Sex, list(City=all.employees$Sex), FUN=length)
```

```
##   City x
## 1    F 2
```

```
## 2    M 7
```

4.7 Reshaping dataframe

Reshaping adalah mengubah format dataframe dari "long" ke "wide" atau sebaliknya. Sebagai contoh

```
# https://riptutorial.com/r/example/12036/the-reshape-function
set.seed(1234)
df <- data.frame(identifier=rep(1:5, each=3),
                 location=rep(c("up", "down", "left", "up", "center"), each=3),
                 period=rep(1:3, 5), counts=sample(35, 15, replace=TRUE),
                 values=runif(15, 5, 10))[-c(4,8,11),]
df
```

```
##    identifier location period counts   values
## 1           1       up      1     28 9.052993
## 2           1       up      2     16 7.628488
## 3           1       up      3     22 9.573291
## 5           2     down      2      5 5.228851
## 6           2     down      3     16 7.280457
## 7           3     left      1      4 6.325933
## 9           3     left      3     22 7.536534
## 10          4       up      1     26 5.905481
## 12          4       up      3     15 6.006240
## 13          5   center      1     14 6.294049
## 14          5   center      2     14 9.960752
## 15          5   center      3      4 9.036762
```

4.7.1 Long to wide

Reshape ke format *wide* untuk pada kolom "period"

```
df.wide <- reshape(df, idvar="identifier", timevar="period",
                   v.names=c("values", "counts"), direction="wide")
df.wide
```

```
##    identifier location values.1 counts.1 values.2 counts.2 values.3 counts.3
## 1           1       up 9.052993       28 7.628488       16 9.573291       22
## 5           2     down       NA       NA 5.228851        5 7.280457       16
## 7           3     left 6.325933        4       NA       NA 7.536534       22
## 10          4       up 5.905481       26       NA       NA 6.006240       15
## 13          5   center 6.294049       14 9.960752       14 9.036762        4
```

4.7.2 Wide to long

Reshape ke format *long*

```
reshape(df.wide, idvar="identifier",
        varying=list(c(3,5,7), c(4,6,8)), direction="long")
```

```
##     identifier location time values.1 counts.1
## 1.1          1       up    1 9.052993       28
## 2.1          2     down    1       NA       NA
## 3.1          3     left    1 6.325933        4
## 4.1          4       up    1 5.905481       26
## 5.1          5   center    1 6.294049       14
## 1.2          1       up    2 7.628488       16
## 2.2          2     down    2 5.228851        5
## 3.2          3     left    2       NA       NA
## 4.2          4       up    2       NA       NA
## 5.2          5   center    2 9.960752       14
## 1.3          1       up    3 9.573291       22
## 2.3          2     down    3 7.280457       16
```

```
## 3.3          3     left    3 7.536534       22
## 4.3          4       up    3 6.006240       15
## 5.3          5   center    3 9.036762        4
```

Chapter 5

Data Wrangling dengan `tidyverse`

Tiga sub-bab pertama dalam artikel ini merupakan bagian dari tugas STA581 dan pernah dipublikasikan RPubs.com/nurandi.

5.1 Data Wrangling

Aktivitas apa yang biasa dilakukan oleh *data scientist* terhadap data tabular? Barangkali menghapus kolom atau baris, melakukan kalkulasi, menambahkan kolom baru atau melakukan agregasi. Aktivitas-aktivitas tersebut sering disebut sebagai *data wrangling* (The OHI Team 2019) atau manipulasi data (dalam konitasi positif) yang bertujuan untuk mengubah data menjadi format yang lebih mudah digunakan atau mudah dipahami. Manipulasi data menjadi bagian tidak diterpisahkan dalam persiapan data yang umumnya membutuhkan waktu paling lama dari keseluruhan rangkaian analisis data. Skenario dalam proses ini berbeda-beda tergantung pada data yang digunakan dan tujuan yang ingin dicapai (Stobierski 2021).

5.2 R, `tidyverse` dan `dplyr`

Sebagai bahasa pemrograman populer dalam sains data, R menyediakan berbagai paket/*library* untuk tujuan-tujuan spesifik. Sebagai contoh, kita dapat memanfaatkan paket `tidyverse`; sekumpulan beberapa paket untuk ekplorasi, manipulasi dan visualisasi data (Wickham et al. 2019), yang terdiri dari paket-paket antara lain:

- `ggplot2` : membuat grafik dan visualisasi data
- `dplyr` : manipulasi data
- `tidyr` : membentuk *"tidy data"*, yaitu data dalam format yang konsisten
- `readr` : membaca berbagai data tabular
- `purrr` : bekerja dengan fungsi dan vektor
- `tibble` : bentuk lain dari *data frame* yang lebih modern
- `stringr` : bekerja dengan *string*
- `forcats` : bekerja dengan *factor*
- `lubridate` : bekerja dengan data berformat tanggal dan waktu

Artikel ini akan fokus pada pemanfaatan paket `dplyr` (Wickham et al. 2021b). Paket yang dikembangkan oleh Hadley Wickham dan tim ini dipandang sebagai *"grammar"* yang di dalamnya tersedia sejumlah *"verb"* untuk menyesaikan berbagai pekerjaan terkait manipulasi data (Wickham et al. 2021a), di antaranya untuk:

- memilih kolom,
- menyeleksi baris berdasarkan kriteria tertentu,
- agregasi data,
- menghitung kolom/variabel baru,
- mengatur urutan baris, dan lain-lain

5.3 Studi Kasus: Data MovieLens

Untuk mengeksplorasi dasar-dasar manipulasi data dengan `dplyr`, kita akan menggunakan data **MoviLens** (Harper and Konstan 2015), yang bisa diperoleh dari paket `dslabs`. Data set ini berisi rating dari *movie*/film

Figure 5.1: tidyverse. Sumber gambar: Che Smith (chsmith1@davidson.edu)

dari website MovieLens yang dikumpulkan dan dikelola oleh GroupLens, kelompok riset di Universitas Minnesota

5.3.1 Persiapan

Instalasi paket-paket yang diperlukan, yaitu `tidyverse` (atau cukup `dplyr`) dan `dslabs`. Instalasi paket ini sifatnya opsional. Maksudnya apabila paket tersebut sudah terinstal maka tidak perlu melakukan instalasi lagi.

```
install.packages(c("tidyverse","dslabs"))
```

Lalu *load* paket-paket tersebut.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
```

```
## Warning: package 'readr' was built under R version 4.1.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(dslabs)
```

Selanjutnya *load* data `movilens` dari paket `dslabs`

```
data(movielens)
```

Sebelum memulai proses manipulasi data, sangat direkomendasikan untuk melihat bentuk dan struktur data. Kita sudah mempunyai data `movilens`, yang merupakan sebuah *data frame*, yang dapat kita ubah menjadi `tibble` agar lebih mudah dalam menginspeksi data, terutama data yang berukuran besar. Sebuah `tibble` apabila ditampilkan dalam layar, hanya muncul maksimal 10 baris pertama, dilengkapi dengan informasi mengenai dimensi tabel, nama dan tipe kolom serta tampilan akan menyesuaikan lebar layar.

```
movielens <- as_tibble(movielens)
movielens
```

```
## # A tibble: 100,004 x 7
##    movieId title                 year genres           userId rating timestamp
##      <int> <chr>                <int> <fct>             <int>  <dbl>     <int>
##  1      31 Dangerous Minds       1995 Drama                 1    2.5    1.26e9
##  2    1029 Dumbo                 1941 Animation|Childr~     1    3      1.26e9
##  3    1061 Sleepers              1996 Thriller              1    3      1.26e9
##  4    1129 Escape from New York  1981 Action|Adventure~     1    2      1.26e9
##  5    1172 Cinema Paradiso (Nuo~ 1989 Drama                 1    4      1.26e9
##  6    1263 Deer Hunter, The      1978 Drama|War             1    2      1.26e9
##  7    1287 Ben-Hur               1959 Action|Adventure~     1    2      1.26e9
##  8    1293 Gandhi                1982 Drama                 1    2      1.26e9
##  9    1339 Dracula (Bram Stoker~ 1992 Fantasy|Horror|R~     1    3.5    1.26e9
## 10    1343 Cape Fear             1991 Thriller              1    2      1.26e9
## # ... with 99,994 more rows
```

Alternatif lain untuk menampilkan struktur data adalah fungsi `glimpse`. Fungsi ini sebenernya mirip dengan fungsi `str` dari paket `utils`.

```
glimpse(movielens)
```

```
## Rows: 100,004
## Columns: 7
## $ movieId   <int> 31, 1029, 1061, 1129, 1172, 1263, 1287, 1293, 1339, 1343, 13~
## $ title     <chr> "Dangerous Minds", "Dumbo", "Sleepers", "Escape from New Yor~
## $ year      <int> 1995, 1941, 1996, 1981, 1989, 1978, 1959, 1982, 1992, 1991, ~
## $ genres    <fct> Drama, Animation|Children|Drama|Musical, Thriller, Action|Ad~
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ rating    <dbl> 2.5, 3.0, 3.0, 2.0, 4.0, 2.0, 2.0, 2.0, 3.5, 2.0, 2.5, 1.0, ~
## $ timestamp <int> 1260759144, 1260759179, 1260759182, 1260759185, 1260759205, ~
```

Dari output di atas kita tahun bahwa data `movielens` terdiri dari 100004 baris dan 7 kolom, yaitu

Nama	Tipe	Contoh	Keterangan
movieId	int	31	ID film
title	chr	Dangerous Minds	Judul film
year	int	1995	Tahun rilis
genres	fct	Drama	*Genre*/aliran (bisa terdapat beberapa genre)
userId	int	1	ID pengguna
rating	dbl	2.5	Rating
timestamp	int	1260759144	Waktu dalam format *unix timestamp*

5.3.2 Operator *pipe* %>%

Sebelum membahas `dplyr` lebih lanjut, mari berkenalan dengan operator *pipe* `%>%`. *Pipe* merupakan operator yang berasal dari paket `magrittr` (Bache and Wickham 2020), yang dalam `tidyverse` dimuat secara otomatis.

Perhatikan perintah berikut ini.

```
nama_fungsi(nama_object)
```

apabila ditulis dengan *pipe*, akan menjadi

```
nama_object %>% nama_fungsi
```

Operator *pipe* sangat bermafaat untuk menuliskan banyak operasi secara sekuensial atau berurutan. Sebagai contoh, kita ingin membulatkan vektor numerik hingga dua tempat desimal, mengurutkannya dari besar ke kecil, lalu tampilkan enam elemen pertama.

```
set.seed(123)
number_data <- runif(n = 15, min = 0, max = 100)
```

Dengan *base R* dapat kita tulis

```
head(sort(round(number_data, digit = 2), decreasing = TRUE))
```

```
## [1] 95.68 94.05 89.24 88.30 78.83 67.76
```

Dengan operator *pipe* menjadi:

```
number_data %>%
  round(digits = 2) %>%
  sort(decreasing = TRUE) %>%
  head()
```

```
## [1] 95.68 94.05 89.24 88.30 78.83 67.76
```

5.3.3 `dplyr`*'s verbs*

Sebagai *"grammar"* untuk manipulasi data, paket `dplyr` mempunyai setidaknya lima *"verbs"* utama, masing-masing mempunya fungsi yang spesifik, yaitu:

- `select()` : memilih kolom
- `filter()` : menyeleksi baris berdasarkan kriteria tertentu
- `summarise()` : meringkas atau agregasi data
- `mutate()` : menghitung kolom/variabel baru
- `arrange()` : mengatur urutan baris

Selain fungsi-fungsi di atas, masih banyak fungsi lain yang dapat digunakan, misalnya `group_by()` untuk pengelompokan data. Mari kita eksplorasi lebih lanjut.

5.3.4 Memilih kolom: `select()`

Ketika bekerja dengan data yang mempunyai banyak kolom, biasanya kita ingin memilih kolom-kolom tertentu saja. Hal ini bisa kita lakukan dengan memanfaatkan fungsi `select()`berdasarkan nama atau posisi kolom. Misalnya dua perintah berikut akan memilih kolom `title`, `year` dan `genres` dari `movielens`.

```
movielens %>%
  select(title, year, genres)
```

```
## # A tibble: 100,004 x 3
##    title                                  year genres
##    <chr>                                 <int> <fct>
##  1 Dangerous Minds                        1995 Drama
##  2 Dumbo                                  1941 Animation|Children|Drama|Music~
##  3 Sleepers                               1996 Thriller
##  4 Escape from New York                   1981 Action|Adventure|Sci-Fi|Thrill~
##  5 Cinema Paradiso (Nuovo cinema Paradiso) 1989 Drama
##  6 Deer Hunter, The                       1978 Drama|War
##  7 Ben-Hur                                1959 Action|Adventure|Drama
##  8 Gandhi                                 1982 Drama
##  9 Dracula (Bram Stoker's Dracula)        1992 Fantasy|Horror|Romance|Thriller
## 10 Cape Fear                              1991 Thriller
## # ... with 99,994 more rows
```

```
movielens %>%
  select(2, 3, 4)
```

```
## # A tibble: 100,004 x 3
##    title                                  year genres
##    <chr>                                 <int> <fct>
##  1 Dangerous Minds                        1995 Drama
##  2 Dumbo                                  1941 Animation|Children|Drama|Music~
##  3 Sleepers                               1996 Thriller
##  4 Escape from New York                   1981 Action|Adventure|Sci-Fi|Thrill~
##  5 Cinema Paradiso (Nuovo cinema Paradiso) 1989 Drama
##  6 Deer Hunter, The                       1978 Drama|War
##  7 Ben-Hur                                1959 Action|Adventure|Drama
##  8 Gandhi                                 1982 Drama
```

```
##  9 Dracula (Bram Stoker's Dracula)          1992 Fantasy|Horror|Romance|Thriller
## 10 Cape Fear                                1991 Thriller
## # ... with 99,994 more rows
```

Kita dapat menambahkan tanda minus - untuk tidak memilih kolom tersebut.

```
movielens %>%
  select(-title, -year, -genres)
```

```
## # A tibble: 100,004 x 4
##    movieId userId rating  timestamp
##      <int>  <int>  <dbl>      <int>
##  1      31      1    2.5 1260759144
##  2    1029      1    3   1260759179
##  3    1061      1    3   1260759182
##  4    1129      1    2   1260759185
##  5    1172      1    4   1260759205
##  6    1263      1    2   1260759151
##  7    1287      1    2   1260759187
##  8    1293      1    2   1260759148
##  9    1339      1    3.5 1260759125
## 10    1343      1    2   1260759131
## # ... with 99,994 more rows
```

Ada sejumlah fungsi pembantu *(helper function)* yang bisa digunakan dalam `select()`, di antaranya:

- `starts_with("abc")` : nama kolom diawali "abc."
- `ends_with("xyz")` : nama kolom diakhiri "xyz."
- `contains("ijk")` : nama kolom mengandung "ijk."
- `num_range("x", 1:3)` : memilih kolom `x1`, `x2` dan `x3`.

Selain memilih kolom, `select()` juga dapat digunakan untuk mengubah nama kolom, misalnya

```
movielens %>%
  select(movie_title = title, year, genres)
```

```
## # A tibble: 100,004 x 3
##    movie_title                               year genres
##    <chr>                                    <int> <fct>
##  1 Dangerous Minds                           1995 Drama
##  2 Dumbo                                     1941 Animation|Children|Drama|Music~
##  3 Sleepers                                  1996 Thriller
##  4 Escape from New York                      1981 Action|Adventure|Sci-Fi|Thrill~
##  5 Cinema Paradiso (Nuovo cinema Paradiso)   1989 Drama
##  6 Deer Hunter, The                          1978 Drama|War
##  7 Ben-Hur                                   1959 Action|Adventure|Drama
##  8 Gandhi                                    1982 Drama
##  9 Dracula (Bram Stoker's Dracula)           1992 Fantasy|Horror|Romance|Thriller
## 10 Cape Fear                                 1991 Thriller
## # ... with 99,994 more rows
```

5.3.5 Menyeleksi baris: `filter()`

`filter()` digunakan untuk menyeleksi atau memilih baris atau observasi berdasarkan nilainya. Misalnya kita ingin menampilkan film-film yang dirilis tahun 1995.

```
movielens %>%
  filter(year == 1995)
```

```
## # A tibble: 6,635 x 7
##    movieId title                  year genres           userId rating timestamp
##      <int> <chr>                 <int> <fct>             <int>  <dbl>     <int>
##  1      31 Dangerous Minds        1995 Drama                 1    2.5    1.26e9
##  2      10 GoldenEye              1995 Action|Adventur~      2    4      8.35e8
##  3      17 Sense and Sensibility  1995 Drama|Romance         2    5      8.35e8
```

```
##  4      39 Clueless                1995 Comedy|Romance        2    5      8.35e8
##  5      47 Seven (a.k.a. Se7en)    1995 Mystery|Thriller      2    4      8.35e8
##  6      50 Usual Suspects, The     1995 Crime|Mystery|T~      2    4      8.35e8
##  7      52 Mighty Aphrodite        1995 Comedy|Drama|Ro~      2    3      8.35e8
##  8      62 Mr. Holland's Opus      1995 Drama                 2    3      8.35e8
##  9     110 Braveheart              1995 Action|Drama|War      2    4      8.35e8
## 10     144 Brothers McMullen, The  1995 Comedy                2    3      8.35e8
## # ... with 6,625 more rows
```

Dalam `filter()`, kita dapat menggunakan berbagai operator, seperti operator dasar `<`, `<=`, `>`, `>`, `==` (sama dengan) dan `%in%` (bagian dari). Argumen `filter()` yang lebih dari satu dapat digabungkan dengan *bolean* operator, yaitu `&` (*and*/dan), `|` (*or*/atau) dan `!` (*not*/tidak). Misalnya untuk menampilkan film-film yang dirilis tahun 1995 dan 1996 serta beraliran/*genre* hanya drama:

```
movielens %>%
  filter(year %in% c(1995, 1996) & genres == 'Drama')
```

```
## # A tibble: 582 x 7
##    movieId title                     year genres userId rating  timestamp
##      <int> <chr>                    <int> <fct>   <int>  <dbl>      <int>
##  1      31 Dangerous Minds           1995 Drama       1    2.5 1260759144
##  2      62 Mr. Holland's Opus        1995 Drama       2    3    835355749
##  3    1358 Sling Blade               1996 Drama       6    2   1109258181
##  4      31 Dangerous Minds           1995 Drama       7    3    851868750
##  5      40 Cry, the Beloved Country  1995 Drama       7    4    851866901
##  6    1358 Sling Blade               1996 Drama       8    0.5 1154474527
##  7      26 Othello                   1995 Drama       9    3    938628655
##  8    1358 Sling Blade               1996 Drama       9    4    938628450
##  9    1358 Sling Blade               1996 Drama      10    5    942766420
## 10    1423 Hearts and Minds          1996 Drama      10    4    942766420
## # ... with 572 more rows
```

Sekarang, kolom `genres` hanya berisi satu nilai yaitu `Drama` sehingga kita bisa harus kolom tersebut

```
movielens %>%
  filter(year %in% c(1995, 1996) & genres == 'Drama') %>%
  select(-genres)
```

```
## # A tibble: 582 x 6
##    movieId title                     year userId rating  timestamp
##      <int> <chr>                    <int>  <int>  <dbl>      <int>
##  1      31 Dangerous Minds           1995      1    2.5 1260759144
##  2      62 Mr. Holland's Opus        1995      2    3    835355749
##  3    1358 Sling Blade               1996      6    2   1109258181
##  4      31 Dangerous Minds           1995      7    3    851868750
##  5      40 Cry, the Beloved Country  1995      7    4    851866901
##  6    1358 Sling Blade               1996      8    0.5 1154474527
##  7      26 Othello                   1995      9    3    938628655
##  8    1358 Sling Blade               1996      9    4    938628450
##  9    1358 Sling Blade               1996     10    5    942766420
## 10    1423 Hearts and Minds          1996     10    4    942766420
## # ... with 572 more rows
```

5.3.6 Menambah kolom: `mutate()`

Selain menggunakan kolom yang sudah tersedia dalam data, seringkali kita ingin membuat kolom baru yang merupakan turunan dari kolom yang sudah ada. Dalam `movielens`, kolom `timestamp` ditulis dalam format `unix timestamp` (jumlah detik dihitung sejak 1 Januari 1970, jam 00:00:00 UTC). Agar lebih mudah dipahami, kita dapat membuat kolom baru dengan mengubah kolom tersebut ke format *datetime*.

```
movielens %>%
  mutate(ts = as.POSIXct(timestamp, origin = "1970-01-01")) %>%
  select(-timestamp)
```

```
## # A tibble: 100,004 x 7
##    movieId title            year genres        userId rating ts
##      <int> <chr>           <int> <fct>          <int>  <dbl> <dttm>
##  1      31 Dangerous Minds  1995 Drama              1    2.5 2009-12-14 09:52:24
##  2    1029 Dumbo            1941 Animation|Ch~      1    3   2009-12-14 09:52:59
##  3    1061 Sleepers         1996 Thriller           1    3   2009-12-14 09:53:02
##  4    1129 Escape from Ne~  1981 Action|Adven~      1    2   2009-12-14 09:53:05
##  5    1172 Cinema Paradis~  1989 Drama              1    4   2009-12-14 09:53:25
##  6    1263 Deer Hunter, T~  1978 Drama|War          1    2   2009-12-14 09:52:31
##  7    1287 Ben-Hur          1959 Action|Adven~      1    2   2009-12-14 09:53:07
##  8    1293 Gandhi           1982 Drama              1    2   2009-12-14 09:52:28
##  9    1339 Dracula (Bram ~  1992 Fantasy|Horr~      1    3.5 2009-12-14 09:52:05
## 10    1343 Cape Fear        1991 Thriller           1    2   2009-12-14 09:52:11
## # ... with 99,994 more rows
```

Contoh lain, kita ingin membuat kolom baru yang menyatakan bahwa film berjenis Drama atau bukan:

```
movielens %>%
  mutate(isDrama = grepl("Drama", genres))
```

```
## # A tibble: 100,004 x 8
##    movieId title             year genres         userId rating timestamp isDrama
##      <int> <chr>            <int> <fct>           <int>  <dbl>     <int> <lgl>
##  1      31 Dangerous Minds   1995 Drama               1    2.5    1.26e9 TRUE
##  2    1029 Dumbo             1941 Animation|Chi~      1    3      1.26e9 TRUE
##  3    1061 Sleepers          1996 Thriller           1    3      1.26e9 FALSE
##  4    1129 Escape from New~  1981 Action|Advent~      1    2      1.26e9 FALSE
##  5    1172 Cinema Paradiso~  1989 Drama              1    4      1.26e9 TRUE
##  6    1263 Deer Hunter, The  1978 Drama|War          1    2      1.26e9 TRUE
##  7    1287 Ben-Hur           1959 Action|Advent~     1    2      1.26e9 TRUE
##  8    1293 Gandhi            1982 Drama              1    2      1.26e9 TRUE
##  9    1339 Dracula (Bram S~  1992 Fantasy|Horro~     1    3.5    1.26e9 FALSE
## 10    1343 Cape Fear         1991 Thriller           1    2      1.26e9 FALSE
## # ... with 99,994 more rows
```

Kedua perintah di atas dapat digabungkan menjadi

```
movielens %>%
  mutate(ts = as.POSIXct(timestamp, origin = "1970-01-01"),
         isDrama = grepl("Drama", genres)) %>%
  select(-timestamp)
```

```
## # A tibble: 100,004 x 8
##    movieId title        year genres    userId rating ts                  isDrama
##      <int> <chr>       <int> <fct>      <int>  <dbl> <dttm>              <lgl>
##  1      31 Dangerous ~  1995 Drama          1    2.5 2009-12-14 09:52:24 TRUE
##  2    1029 Dumbo        1941 Animatio~      1    3   2009-12-14 09:52:59 TRUE
##  3    1061 Sleepers     1996 Thriller       1    3   2009-12-14 09:53:02 FALSE
##  4    1129 Escape fro~  1981 Action|A~      1    2   2009-12-14 09:53:05 FALSE
##  5    1172 Cinema Par~  1989 Drama          1    4   2009-12-14 09:53:25 TRUE
##  6    1263 Deer Hunte~  1978 Drama|War      1    2   2009-12-14 09:52:31 TRUE
##  7    1287 Ben-Hur      1959 Action|A~      1    2   2009-12-14 09:53:07 TRUE
##  8    1293 Gandhi       1982 Drama          1    2   2009-12-14 09:52:28 TRUE
##  9    1339 Dracula (B~  1992 Fantasy|~      1    3.5 2009-12-14 09:52:05 FALSE
## 10    1343 Cape Fear    1991 Thriller       1    2   2009-12-14 09:52:11 FALSE
## # ... with 99,994 more rows
```

5.3.7 Meringkas data: `summarise()`

`summarise()` berfungsi untuk meringkas atau agregasi baris data, seperti untuk menghitung banyaknya pengamatan, nilai tengah, total, nilai maksimum dan minimum, dan lain-lain.

```
movielens %>%
  summarise(uniqueTitle = n_distinct(title),
```

```
            totalReview = n(),
            avgRating = mean(rating))
```

```
## # A tibble: 1 x 3
##   uniqueTitle totalReview avgRating
##         <int>       <int>     <dbl>
## 1        8832      100004      3.54
```

Contoh di atas menghitung banyaknya baris, banyaknya judul unik, dan rata-rata dari rating dalam keseluruhan *dataframe*, dan meringkasnya menjadi satu baris. Kita dapat melakukan agregasi untuk setiap kelompok/*group*/*class* satu kolom atau lebih, dengan memanfaatkan perintah `group_by()`. Misalnya, contoh di atas dapat dimodifikasi agar perhitungan dilakukan untuk setiap tahun rilis. Dengan menambahkan `group_by(year)`, maka perintah yang dimaksud adalah sebagai berikut:

```
movielens %>%
  group_by(year) %>%
  summarise(uniqueTitle = n_distinct(title),
            totalReview = n(),
            avgRating = mean(rating))
```

```
## # A tibble: 104 x 4
##     year uniqueTitle totalReview avgRating
##    <int>       <int>       <int>     <dbl>
##  1  1902           1           6      4.33
##  2  1915           1           2      3
##  3  1916           1           1      3.5
##  4  1917           1           2      4.25
##  5  1918           1           2      4.25
##  6  1919           1           1      3
##  7  1920           3          15      3.7
##  8  1921           5          12      4.42
##  9  1922           6          28      3.80
## 10  1923           3           3      4.17
## # ... with 94 more rows
```

Terlihat bahwa kolom tahun bersifat unik, artinya satu tahun hanya menempati satu baris.

`mutate()` juga dapat dipasangkan dengan `group_by()`, sehingga kolom baru yang terbentu akan berisi nilai agregat yang dihitung per grup. Misal

```
movielens %>%
  group_by(year) %>%
  mutate(uniqueTitle = n_distinct(title),
            totalReview = n(),
            avgRating = mean(rating)) %>%
  filter(year < 1920)
```

```
## # A tibble: 14 x 10
## # Groups:   year [6]
##    movieId title    year genres  userId rating timestamp uniqueTitle totalReview
##      <int> <chr>   <int> <fct>    <int>  <dbl>     <int>       <int>       <int>
##  1    7065 Birth ~  1915 Drama|~    262    2.5    1.43e9           1           2
##  2   32898 Trip t~  1902 Action~    262    3      1.43e9           1           6
##  3   32898 Trip t~  1902 Action~    299    4.5    1.34e9           1           6
##  4   32898 Trip t~  1902 Action~    378    4      1.44e9           1           6
##  5    3309 Dog's ~  1918 Comedy     468    4.5    1.30e9           1           2
##  6    7065 Birth ~  1915 Drama|~    468    3.5    1.30e9           1           2
##  7    8511 Immigr~  1917 Comedy     468    4.5    1.30e9           1           2
##  8   32898 Trip t~  1902 Action~    468    4.5    1.30e9           1           6
##  9   62383 20,000~  1916 Action~    468    3.5    1.30e9           1           1
## 10   72626 Billy ~  1919 Comedy~    468    3      1.30e9           1           1
## 11   32898 Trip t~  1902 Action~    481    5      1.44e9           1           6
## 12   32898 Trip t~  1902 Action~    547    5      1.43e9           1           6
```

```
## 13    3309 Dog's ~  1918 Comedy     554    4      1.01e9           1           2
## 14    8511 Immigr~  1917 Comedy     648    4      1.18e9           1           2
## # ... with 1 more variable: avgRating <dbl>
```

Perhatikan output diatas, untuk kelompok tahun yang sama, maka `uniqueTitle`, `totalReview` dan `avgRating` juga sama nilainya.

5.3.8 Mengurutkan baris: `arrange()`

Data yang terurut umumnya lebih mudah dibaca. Di paket `dplyr` kita dapat mengurutkan *dataframe* berdasarkan kolom tertentu dengan fungsi `arrange()`. Contoh sebelumnya, misalnya, dapat kita urutkan dari tahun terlama ke tahun terbaru sebagai berikut:

```
movielens %>%
  group_by(year) %>%
  mutate(uniqueTitle = n_distinct(title),
            totalReview = n(),
            avgRating = mean(rating)) %>%
  filter(year < 1920) %>%
  arrange(year)
```

```
## # A tibble: 14 x 10
## # Groups:   year [6]
##    movieId title    year genres  userId rating timestamp uniqueTitle totalReview
##      <int> <chr>   <int> <fct>    <int>  <dbl>     <int>       <int>       <int>
##  1   32898 Trip t~  1902 Action~    262    3      1.43e9           1           6
##  2   32898 Trip t~  1902 Action~    299    4.5    1.34e9           1           6
##  3   32898 Trip t~  1902 Action~    378    4      1.44e9           1           6
##  4   32898 Trip t~  1902 Action~    468    4.5    1.30e9           1           6
##  5   32898 Trip t~  1902 Action~    481    5      1.44e9           1           6
##  6   32898 Trip t~  1902 Action~    547    5      1.43e9           1           6
##  7    7065 Birth ~  1915 Drama|~    262    2.5    1.43e9           1           2
##  8    7065 Birth ~  1915 Drama|~    468    3.5    1.30e9           1           2
##  9   62383 20,000~  1916 Action~    468    3.5    1.30e9           1           1
## 10    8511 Immigr~  1917 Comedy     468    4.5    1.30e9           1           2
## 11    8511 Immigr~  1917 Comedy     648    4      1.18e9           1           2
## 12    3309 Dog's ~  1918 Comedy     468    4.5    1.30e9           1           2
## 13    3309 Dog's ~  1918 Comedy     554    4      1.01e9           1           2
## 14   72626 Billy ~  1919 Comedy~    468    3      1.30e9           1           1
## # ... with 1 more variable: avgRating <dbl>
```

5.3.9 Gabungan beberapa fungsi sekaligus

Setelah mempraktikan bagaimana menggunakan fungsi-fungsi dasar `dplyr`, mari gabungkan beberapa fungsi dalam satu perintah.

Contoh 1: Katakan untuk setiap film drama, kita ingin menghitung berapa banyak penilaian yang diberikan pada tahun perdana dan tahun-tahun setelahnya. Hasilnya diurutkan dari yang mendapat penilaian terbanyak di tahun perdana.

```
movielens %>%
  filter(grepl("Drama", genres)) %>%
  mutate(yearRating = as.numeric(format(as.POSIXct(timestamp, origin = "1970-01-01"), "%Y"))) %>%
  mutate(firstYear = year == yearRating, nextYear = year < yearRating) %>%
  group_by(title) %>%
  summarise(firstYear = sum(firstYear), nextYear = sum(nextYear)) %>%
  arrange(desc(firstYear))
```

```
## # A tibble: 4,249 x 3
##    title                    firstYear nextYear
##    <chr>                        <int>    <int>
##  1 Fargo                           19      205
##  2 Gladiator                       19      153
```

```
##  3 American Beauty                18      202
##  4 Blair Witch Project, The       18       68
##  5 Ex Machina                     18        8
##  6 High Fidelity                  18       70
##  7 Dark Knight, The               17      104
##  8 Sixth Sense, The               17      176
##  9 Erin Brockovich                16       69
## 10 Eraser                         14       55
## # ... with 4,239 more rows
```

Contoh 2: Kita akan menampilkan satu film dengan rata-rata rating terbaik untuk setiap tahun perilisan. Jika ada beberapa film yang mempunyai rating tertinggi, maka dipilih film dengan jumlah rating terbanyak. Hasil akhir berupa *dataframe* dengan kolom tahun, judul dan rata-rata rating.

```
movielens %>%
  group_by(year, title) %>%
  summarise(avgRating = mean(rating), nRating = n()) %>%
  group_by(year) %>%
  arrange(year, desc(avgRating), desc(nRating)) %>%
  mutate(rn = row_number()) %>%
  filter(rn == 1) %>%
  select(-rn, -nRating) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'year'. You can override using the `.groups` argument.
```

```
## # A tibble: 104 x 3
##     year title                                                    avgRating
##    <int> <chr>                                                        <dbl>
##  1  1902 Trip to the Moon, A (Voyage dans la lune, Le)                 4.33
##  2  1915 Birth of a Nation, The                                        3
##  3  1916 20,000 Leagues Under the Sea                                  3.5
##  4  1917 Immigrant, The                                                4.25
##  5  1918 Dog's Life, A                                                 4.25
##  6  1919 Billy Blazes, Esq.                                            3
##  7  1920 Cabinet of Dr. Caligari, The (Cabinet des Dr. Caligari., Das)  4
##  8  1921 Goat, The                                                     5
##  9  1922 Cops                                                          5
## 10  1923 Our Hospitality                                               4.5
## # ... with 94 more rows
```

Dari hasil eksplorasi di atas, paket `dplyr` yang merupakan salah satu bagian inti dari paket `tidyverse` merupakan alat yang bisa diandalkan untuk manipulasi *dataframe* dalam R. Meskipun demikian, untuk keperluan yang lebih komplek, `dplyr` membutuhkan fungsi-fungsi yang tersedia di paket lain, baik itu paket bawaan seperti `base` dan `utils`, maupun paket lain. Misalnya untuk mengolah data *string/text* bisa menggunakan paket `stringr`, data berformat tanggal dan waktu bisa menggunakan paket `lubridate`. Sementara untuk melakukan *pivoting* atau *un-pivoting* bisa menggunakan paket `tidyr`.

Contoh-contoh lain dalam menggunakan `dplyr` dapat dipelajari di buku *R for Data Science* (Wickham and Grolemund 2017).

5.4 Join/Merge Dataframe

Join atau merge merupakan proses penggabungan dua dataframe berdasarkan kolom kunci tertentu. Dalam `dplyr`/`tidyverse` dikenal beberapa jenis join yaitu `inner_join()`, `semi_join()`, `left_join()`, `right_join()`,`full_join()`, dan `anti_join()`.

Kita akan mempraktikan penggunaan ke-enam jenis join tersebut menggunakan data berikut ini. Seluruh pembahasan dalam sub-bagian join/merge diambil dari stat545.com.

```
superheroes <- tibble::tribble(
       ~name, ~alignment,  ~gender,          ~publisher,
   "Magneto",      "bad",   "male",            "Marvel",
     "Storm",     "good", "female",            "Marvel",
```

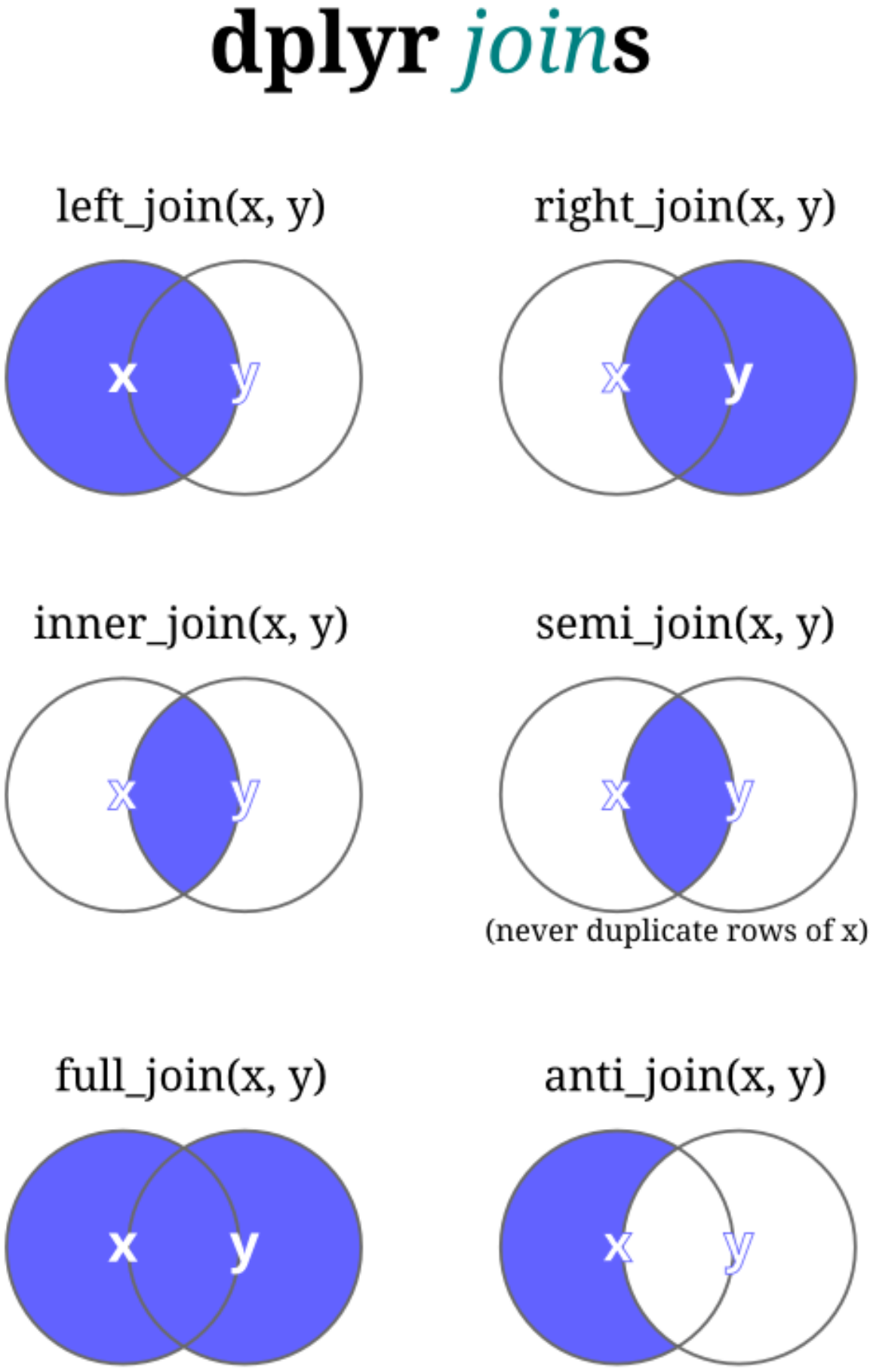

Figure 5.2: Join dalam tidyverse. Sumber gambar: Minnier & Niederhausen)

```
  "Mystique",      "bad", "female",            "Marvel",
    "Batman",     "good",   "male",                "DC",
     "Joker",      "bad",   "male",                "DC",
  "Catwoman",      "bad", "female",                "DC",
   "Hellboy",     "good",   "male", "Dark Horse Comics"
  )

publishers <- tibble::tribble(
  ~publisher, ~yr_founded,
        "DC",       1934L,
    "Marvel",       1939L,
     "Image",       1992L
  )
```

5.4.1 inner_join(x,y)

Mengembalikan semua baris dari `x` yang berpasangan dengan nilai di tabel `y` berdasarkan kolom tertentu. Semua kolom dari kedua tabel akan ditampilkan. Jika ada *multiple matches*, maka seluruh kombinasi yang *match* tersebut akan dikembalikan.

```
inner_join(superheroes, publishers)
```

```
## Joining, by = "publisher"
```

```
## # A tibble: 6 x 5
##   name     alignment gender publisher yr_founded
##   <chr>    <chr>     <chr>  <chr>          <int>
## 1 Magneto  bad       male   Marvel          1939
## 2 Storm    good      female Marvel          1939
## 3 Mystique bad       female Marvel          1939
## 4 Batman   good      male   DC              1934
## 5 Joker    bad       male   DC              1934
## 6 Catwoman bad       female DC              1934
```

5.4.2 semi_join(x,y)

Mirip seperti `inner_join()` hanya saja tidak akan mengembalikan nilai duplikat. Pada `semi_join`, hanya mengembalikan kolom-kolom dari tabel `x`.

```
semi_join(superheroes, publishers)
```

```
## Joining, by = "publisher"
```

```
## # A tibble: 6 x 4
##   name     alignment gender publisher
##   <chr>    <chr>     <chr>  <chr>
## 1 Magneto  bad       male   Marvel
## 2 Storm    good      female Marvel
## 3 Mystique bad       female Marvel
## 4 Batman   good      male   DC
## 5 Joker    bad       male   DC
## 6 Catwoman bad       female DC
```

5.4.3 left_join(x,y)

Mengembalikan semua baris dari `x`. Jika ada match dengan tabel `y`, maka data akan dikembalikan. Jika tidak match, maka akan diisi dengan `NA`. Pada `left_join()` kolom-kolom dari tabel `x` dan `y` dikembalikan.

```
left_join(superheroes, publishers)
```

```
## Joining, by = "publisher"
```

```
## # A tibble: 7 x 5
##   name     alignment gender publisher         yr_founded
```

```
##   <chr>    <chr>     <chr>  <chr>                <int>
## 1 Magneto  bad       male   Marvel                1939
## 2 Storm    good      female Marvel                1939
## 3 Mystique bad       female Marvel                1939
## 4 Batman   good      male   DC                    1934
## 5 Joker    bad       male   DC                    1934
## 6 Catwoman bad       female DC                    1934
## 7 Hellboy  good      male   Dark Horse Comics       NA
```

5.4.4 right_join(x,y)

Kebalikannya dari `left_join()`

```
right_join(superheroes, publishers)
```

```
## Joining, by = "publisher"
```

```
## # A tibble: 7 x 5
##   name     alignment gender publisher yr_founded
##   <chr>    <chr>     <chr>  <chr>          <int>
## 1 Magneto  bad       male   Marvel          1939
## 2 Storm    good      female Marvel          1939
## 3 Mystique bad       female Marvel          1939
## 4 Batman   good      male   DC              1934
## 5 Joker    bad       male   DC              1934
## 6 Catwoman bad       female DC              1934
## 7 <NA>     <NA>      <NA>   Image           1992
```

5.4.5 full_join(x,y)

Mengembalikan semua baris dari tabel `x` dan `y` baik yang berpasangan maupun tidak. Untuk baris yang tidak berpasangan akan diisi dengan `NA`. `full_join()` mengembalikan kolom-kolom dari kedua tabel.

```
full_join(superheroes, publishers)
```

```
## Joining, by = "publisher"
```

```
## # A tibble: 8 x 5
##   name     alignment gender publisher         yr_founded
##   <chr>    <chr>     <chr>  <chr>                  <int>
## 1 Magneto  bad       male   Marvel                  1939
## 2 Storm    good      female Marvel                  1939
## 3 Mystique bad       female Marvel                  1939
## 4 Batman   good      male   DC                      1934
## 5 Joker    bad       male   DC                      1934
## 6 Catwoman bad       female DC                      1934
## 7 Hellboy  good      male   Dark Horse Comics         NA
## 8 <NA>     <NA>      <NA>   Image                   1992
```

5.4.6 anti_join(x,y)

Mengembalikan baris-baris dari `x` yang tidak berpasangan dengan `y`. Hanya kolom dari `x` yang dikembalikan.

```
anti_join(superheroes, publishers)
```

```
## Joining, by = "publisher"
```

```
## # A tibble: 1 x 4
##   name    alignment gender publisher
##   <chr>   <chr>     <chr>  <chr>
## 1 Hellboy good      male   Dark Horse Comics
```

5.5 Reshaping Dataframe

Reshaping atau pivoting merupakan proses mengubah format tabel, dari wide ke long atau sebaliknya. Namun umumnya, analis data lebih tertarik untuk mendapatkan format data yang "tidy." Suatu data disebut *tidy* apabila:

- Setiap kolom adalah variabel/peubah
- Setiap baris adalah pengamatan
- Setiap cell adalah nilai tunggal

Format *tidy data* merupakan format penyimpanan standar dalam *tidyverse*. Apabila data kita sudah berformat *tidy*, maka akan memudahkan untuk dilakukan analisis.

Selain dengan paket `base` (seperti yang telah dipelajari di bab sebelumnya), salah satu paket yang dapat digunakan untuk melakukan reshape data adalah `tidyr` yang merupakan bagian dari paket `tidyverse`.

Apabila belum menginstal paket tersebut, dipersilakan menginstalnya terlebih dahulu dan load kedalam workspace:

```
install.packages("tidyr")
```

```
library(tidyr)
```

Untuk praktik kita akan menggunakan data set `airquality` yang ada pada paket `datasets` (Sumber).

```
data(airquality)
head(airquality)
```

```
##   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
```

5.5.1 Long ke wide

Data `airquality` berformat long. Misalnya kita akan mengubahnya menjadi format wide untuk kolom `Month` sebagai nama dan `Temp` sebagai nilai. Fungsi yang dapat digunakan adalah `pivot_wider()`

```
air_wide <-
  airquality %>%
  select(Temp, Month, Day) %>%
  pivot_wider(names_from = Month,
              values_from = Temp,
              names_prefix = "Month")
air_wide
```

```
## # A tibble: 31 x 6
##      Day Month5 Month6 Month7 Month8 Month9
##    <int>  <int>  <int>  <int>  <int>  <int>
##  1     1     67     78     84     81     91
##  2     2     72     74     85     81     92
##  3     3     74     67     81     82     93
##  4     4     62     84     84     86     93
##  5     5     56     85     83     85     87
##  6     6     66     79     83     87     84
##  7     7     65     82     88     89     80
##  8     8     59     87     92     90     78
##  9     9     61     90     92     90     75
## 10    10     69     87     89     92     73
## # ... with 21 more rows
```

Sekarang tabel berformat wide, yang disusun berdasarkan baris `Day` dan kolom `Month5`, `Month6`, dan seterusnya. Nilai dalam setiap cell adalah `Temp`.

5.5.2 Wide ke long

Jika tabel berupa format wide, kita bisa mengubahnya menjadi long dengan perintah `pivot_longer()`.

```
air_long <-
  airquality %>%
  pivot_longer(cols = Ozone:Temp,
               names_to = "Variable",
               values_to = "Value")
air_long
```

```
## # A tibble: 612 x 4
##    Month   Day Variable Value
##    <int> <int> <chr>    <dbl>
##  1     5     1 Ozone     41
##  2     5     1 Solar.R  190
##  3     5     1 Wind       7.4
##  4     5     1 Temp      67
##  5     5     2 Ozone     36
##  6     5     2 Solar.R  118
##  7     5     2 Wind       8
##  8     5     2 Temp      72
##  9     5     3 Ozone     12
## 10     5     3 Solar.R  149
## # ... with 602 more rows
```

Contoh diatas mengubah tabel menjadi long, kolom Ozone hinggal Temp menjadi kolom variabel yang nilai-nilainya ditempatkan pada kolom value.

Chapter 6

Linux Shell

6.1 Apa itu Shell?

Shell adalah istilah untuk *commnd-line interface* antara user dan sistem operasi. Tampilan ini biasa dikenal dengan TUI (Text User Interfaces). Shell biasa digunakan mode interaktif, yaitu langsung kita ketikan perintah pada "terminal" dan enter, maka shell akan memberikan response berupa hasil atau response lainnya. Shell juga dapat digunakan dalam mode non interaktif *(batch)*. Kita simpan perintah-perintah ke dalam file, dan kemudian shell membaca dan mengeksekusi setiap baris command/perintah yang ada dalam file tersebut.

Shell dapat berfungsi sebagai interpreter sekaligus bahasa pemrograman. Sebagai interpreter, artinya shell sebagai penyedia *utility* yang dilengkapi banyak sekali *command* bawaan, seperti `cd`, `rm`, `mkdir` dan lain-lain. Meskipun tujuan utamanya bukan sebagai bahasa pemrograman, shell juga dilengkapi dengan *variable*, *control flow* dan lain-lain.

Setiap sistem operasi mempunyai shell masing-masing, seperti:

- Mac OS X = bash
- Windows = cmd.exe
- Linux = bash

Untuk kesempatan ini, kita akan praktik dasar-dasar shell yang bisa digunakan di sistem operasi berbasis UNIX (seperti Linux, Mac OS dan lain-lain). Sebetulnya, UNIX tidak cuma mempunyai **bash** atau *Bourne Again Shell*, tetapi ada shell lain yang bisa digunakan seperti *Bourne shell/sh*, *Korn shell/ksh*, *POSIX shell/sh*.

Bagi pengguna Windows, dapat menggunakan Windows Subsystem for Linux (WSL).

6.2 Perintah Dasar Shell

6.2.1 Direktori

- Melihat working directory: `pwd`

```
pwd
```

```
## /mnt/d/SSD21/Bookdown/sta561
```

- Membuat directory baru: `mkdir`

```
mkdir test
```

- Pindah ke directory lain: `cd`

```
cd test
```

6.2.2 File

- Membuat file baru: `touch`

```
touch myfile.txt
```

- Menulis pada file

```
echo "hello!" >> myfile.txt
```

- Menampilkan list file dan folder: `ls`

```
ls
```

```
## 01-what-r.Rmd
## 02-r-dasar.Rmd
## 03-operasi-dasar-r.Rmd
## 04-data-wrangling-1.Rmd
## 05-data-wrangling-2.Rmd
## 91-shell.Rmd
## 99-referensi.Rmd
## README.md
## _bookdown.yml
## _bookdown_files
## _output.yml
## docs
## img
## index.Rmd
## myfile.txt
## nur_andi_sta561.Rmd
## nur_andi_sta561.log
## nur_andi_sta561.pdf
## nur_andi_sta561_cache
## reference.bib
## series_cache
## sta561.Rproj
## sta561_cache
## test
## test-figure
```

```
# List file dalam format lebih lengkap
ls -l
```

```
## total 932
## -rwxrwxrwx 1 nurandi nurandi   3243 Sep  6 03:16 01-what-r.Rmd
## -rwxrwxrwx 1 nurandi nurandi  15860 Sep  7 18:15 02-r-dasar.Rmd
## -rwxrwxrwx 1 nurandi nurandi   8636 Sep  6 16:13 03-operasi-dasar-r.Rmd
## -rwxrwxrwx 1 nurandi nurandi   5602 Sep  7 15:52 04-data-wrangling-1.Rmd
## -rwxrwxrwx 1 nurandi nurandi  18459 Sep  8 20:30 05-data-wrangling-2.Rmd
## -rwxrwxrwx 1 nurandi nurandi   2585 Sep  8 20:45 91-shell.Rmd
## -rwxrwxrwx 1 nurandi nurandi     21 Sep  8 20:47 99-referensi.Rmd
## -rwxrwxrwx 1 nurandi nurandi      8 Sep  6 11:52 README.md
## -rwxrwxrwx 1 nurandi nurandi    330 Sep  8 20:30 _bookdown.yml
## drwxrwxrwx 1 nurandi nurandi    512 Sep  8 21:08 _bookdown_files
## -rwxrwxrwx 1 nurandi nurandi    466 Sep  8 20:30 _output.yml
## drwxrwxrwx 1 nurandi nurandi    512 Sep  8 21:08 docs
## drwxrwxrwx 1 nurandi nurandi    512 Sep  7 17:44 img
## -rwxrwxrwx 1 nurandi nurandi    505 Sep  8 20:09 index.Rmd
## -rwxrwxrwx 1 nurandi nurandi      7 Sep  8 21:08 myfile.txt
## -rwxrwxrwx 1 nurandi nurandi  55245 Sep  8 21:08 nur_andi_sta561.Rmd
## -rwxrwxrwx 1 nurandi nurandi  29875 Sep  8 21:07 nur_andi_sta561.log
## -rwxrwxrwx 1 nurandi nurandi 783372 Sep  8 21:07 nur_andi_sta561.pdf
## drwxrwxrwx 1 nurandi nurandi    512 Sep  6 16:33 nur_andi_sta561_cache
## -rwxrwxrwx 1 nurandi nurandi   5429 Sep  7 15:41 reference.bib
## drwxrwxrwx 1 nurandi nurandi    512 Sep  6 12:35 series_cache
## -rwxrwxrwx 1 nurandi nurandi    240 Sep  8 19:59 sta561.Rproj
## drwxrwxrwx 1 nurandi nurandi    512 Sep  6 16:32 sta561_cache
## drwxrwxrwx 1 nurandi nurandi    512 Sep  8 21:08 test
## drwxrwxrwx 1 nurandi nurandi    512 Sep  4 07:28 test-figure
```

```
# List file termasuk hidden file
ls -a
```

```
## .
## ..
## .Rhistory
## .Rproj.user
## .git
## .gitignore
## 01-what-r.Rmd
## 02-r-dasar.Rmd
## 03-operasi-dasar-r.Rmd
## 04-data-wrangling-1.Rmd
## 05-data-wrangling-2.Rmd
## 91-shell.Rmd
## 99-referensi.Rmd
## README.md
## _bookdown.yml
## _bookdown_files
## _output.yml
## docs
## img
## index.Rmd
## myfile.txt
## nur_andi_sta561.Rmd
## nur_andi_sta561.log
## nur_andi_sta561.pdf
## nur_andi_sta561_cache
## reference.bib
## series_cache
## sta561.Rproj
## sta561_cache
## test
## test-figure
```

- Memindahkan dan mengubah nama file: `mv`

```
mv myfile.txt fileku.txt
```

- Menyalin file: `cp`

```
cp fileku.txt myfile.txt
```

- Tampilkan list file: `ls`

```
ls
```

```
## 01-what-r.Rmd
## 02-r-dasar.Rmd
## 03-operasi-dasar-r.Rmd
## 04-data-wrangling-1.Rmd
## 05-data-wrangling-2.Rmd
## 91-shell.Rmd
## 99-referensi.Rmd
## README.md
## _bookdown.yml
## _bookdown_files
## _output.yml
## docs
## fileku.txt
## img
## index.Rmd
## myfile.txt
## nur_andi_sta561.Rmd
```

```
## nur_andi_sta561.log
## nur_andi_sta561.pdf
## nur_andi_sta561_cache
## reference.bib
## series_cache
## sta561.Rproj
## sta561_cache
## test
## test-figure
```

- Menghitung jumlah baris: `wc`

```
wc myfile.txt
```

```
## 1 1 7 myfile.txt
```

- Menampilkan isi file: `cat`

```
cat myfile.txt
```

```
## hello!
```

- Menghapus file: `rm`

```
rm myfile.txt fileku.txt
```

- Menghapus directory: `rm -r`

```
rm -r test
```

Referensi

Bache, Stefan Milton, and Hadley Wickham. 2020. *Magrittr: A Forward-Pipe Operator for r.* https://CRAN.R-project.org/package=magrittr.

Harper, F. Maxwell, and Joseph A. Konstan. 2015. "The MovieLens Datasets: History and Context." *ACM Trans. Interact. Intell. Syst.* 5 (4). https://doi.org/10.1145/2827872.

Stobierski, Tim. 2021. "Data Wrangling: What It Is & Why It's Important." Harvard Business School Online. https://online.hbs.edu/blog/post/data-wrangling.

The OHI Team. 2019. "Introduction to Open Data Science." Ocean Health Index. https://ohi-science.org/data-science-training/.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2021a. "A Grammar of Data Manipulation: Dplyr." RStudio. https://dplyr.tidyverse.org/.

———. 2021b. *Dplyr: A Grammar of Data Manipulation.* https://CRAN.R-project.org/package=dplyr.

Wickham, Hadley, and Garrett Grolemund. 2017. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data.* 1st ed. Paperback; O'Reilly Media. http://r4ds.had.co.nz/.