



# Modul 3

Data Preparation In Data Science Using R

# Pendahuluan

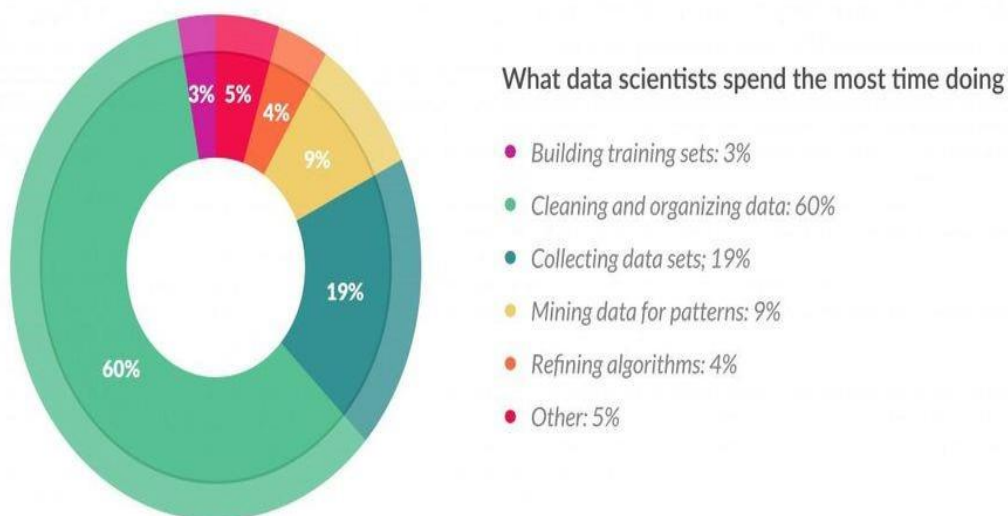
## "Data preparation accounts for about 80% of the work of data scientists"

Kutipan di atas diambil dari salah satu artikel blog online Forbes berdasarkan hasil survey terhadap 80 orang data scientist yang dilakukan oleh salah satu provider data enrichment.

Selengkapnya dapat Anda lihat pada url berikut:

<https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says>

Masih dari link yang sama, berikut adalah pie chart yang merepresentasikan porsi dari pekerjaan data scientist. Proses data preparation di chart ini adalah Collecting data sets (19%) dan Cleaning and organizing data (60%).



Dengan demikian, keterampilan untuk melakukan data preparation atau saat ini sering disebut sebagai data wrangling adalah mutlak untuk seluruh data scientist dan data engineer.

## Lalu apa saja yang dilakukan di proses ini?

Data wrangling adalah proses membaca data dari berbagai sumber dan merubah isi dan struktur sehingga dapat digunakan untuk analisa.

Data cleansing biasanya melibatkan isi yang perlu diubah karena kadang data dimasukkan dari sistem yang berbeda. Bertolak belakang dengan anggapan bahwa dengan sistem, data akan bersih dan standar. Pada kenyataannya, bisnis berkembang lebih cepat dibandingkan dengan pembuatan sistem sehingga banyak design dirubah di tengah jalan untuk mengakomodir hal ini, dan isi menjadi "berantakan". Penyebab utama lainnya tentunya adalah sistem entri data yang manual.

Contoh yang paling sering adalah informasi pelanggan (customer). Di satu sistem namanya boleh memasukkan gelar, dan di satu sistem tidak boleh... jadinya ada dua nama yang serupa tapi tak sama sebagai berikut: **"Agus Cahyono S.Kom."** dan **"Cahyono, Agus"**.

Struktur perlu dirubah karena berbagai alasan, salah satu yang paling penting adalah tiap algoritma mensyaratkan struktur data yang berbeda dan optimal.

Sebagai contoh: transaksi penjualan untuk satu algoritma bisa menggunakan row-based transaction. Sedangkan algoritma lain harus menggunakan sparse-matrix.

Data wrangling adalah topik yang sangat besar, DQLab coba merancang materi ini dengan pembagian ke beberapa course, artikel dan project – semuanya bisa diakses di satu platform yang sama.

Namun hampir sebagian besar proses data wrangling sehari-hari dengan R akan dicakup oleh course ini. Jika proses harus melibatkan tahapan yang lebih sulit dan tidak dicakup di dalam course ini juga akan diinformasikan lebih lanjut.

Klik tombol Next untuk melanjutkan ke bagian berikutnya.

# Apa saja aktivitas-aktivitas yang termasuk di dalam data wrangling?

*Apa saja aktivitas-aktivitas yang termasuk di dalam data wrangling?*

- ☒ Membersihkan data
- ☒ Menambah kolom data
- ☐ Menganalisa 10 nilai penjualan tertinggi
- ☒ Menyatukan dua dataset
- ☒ Memisahkan kolom data
- ☐ Menjaga keamanan data

# Mana aktivitas berikut yang tidak termasuk di dalam data wrangling?

*Mana aktivitas berikut yang tidak termasuk di dalam data wrangling?*

- ☒ Menjaga keamanan data
- ☐ Memisahkan kolom data
- ☐ Menyatukan dua dataset
- ☒ Menganalisa 10 nilai penjualan tertinggi
- ☐ Menambah kolom data
- ☐ Membersihkan Data

# Kenapa data wrangling sangat penting dilakukan?

*Kenapa data wrangling sangat penting dilakukan?*

- ☐ Data akan menjadi lebih aman dari para hacker
- ☒ Data dapat dimanfaatkan lebih jauh oleh bisnis
- ☐ Database tidak diperlukan lagi
- ☒ Mengurangi kesempatan bisnis yang hilang (opportunity lost)
- ☒ Antar sistem lebih mudah "berbicara" karena dapat diintegrasikan dengan lebih baik

# Dataset-dataset Yang Akan Digunakan

Untuk kepentingan penguasaan melalui praktek, sepanjang course ini kita akan bekerja dengan beberapa dataset berikut:

- ☐ Dataset kependudukan.dki dalam bentuk file teks (csv, tsv) dan dalam bentuk Excel. Semuanya dengan isi data yang sama, namun ketika menggunakan R untuk membaca, akan ditunjukkan karakteristik masing-masing yang perlu diperhatikan.
- ☐ Dataset pelanggan dalam bentuk file Excel, dimana kita akan melakukan beberapa proses pengolahan text dan tanggal di dataset ini.
- ☐ Dataset transaksi dalam bentuk database MySQL, dimana penguasaan skillset SQL yang diperlukan dapat dicapai.

Dataset ini akan dijelaskan lebih lanjut pada saat kita mulai mempraktekkan dataset ini.

Klik tombol Next untuk melanjutkan.

# Penutup

Data preparation atau data wrangling adalah proses yang memakan waktu dan tenaga paling besar bagi seorang data scientist, yaitu sampai 80 persen. Detilnya, pembacaan sumber data memakan waktu sampai 20 persen, dan mengorganisasikan dan membersihkan data memakan waktu sampai 60 persen.

Dengan demikian, skillset untuk data wrangling ini merupakan suatu yang mutlak dikuasai oleh para data scientist.

R menyediakan banyak function yang bisa digunakan untuk memudahkan Anda dalam menguasai skillset data wrangling ini.

Course ini akan mencakup cukup banyak praktek data wrangling sebagai bekal untuk mengolah data sehari-hari. Dimulai dari pemahaman mengenai konsep missing data (data yang kosong) dan factor sebagai tipe data yang penting, Anda akan diajak tahap demi tahap menguasai pengolahan teks, bekerja dengan tanggal, menggabungkan data, dan mentransformasi struktur data.

Selamat memulai pelajaran data wrangling dengan R ini, happy learning!

Klik tombol Next untuk melanjutkan ke tahap berikutnya.



# Pendahuluan

Dalam perjalanan Anda sebagai data scientist ataupun data engineer, pasti akan menghadapi kolom dengan sebagian data tidak terisi atau kosong. Ini disebut dengan *missing value* atau nilai yang hilang.

Menyadari hal ini adalah satu tahap penting karena kita akan sadar bahwa hal tersebut adalah lumrah terjadi.

Tahap selanjutnya, kita perlu dua hal:

- Bagaimana missing value direpresentasikan, ini penting sehingga kita bisa identifikasi apakah nilai hilang karena dari awal memang ga ada, apakah karena operasi matematika, dan lain-lain.
- Dengan mengetahui representasi ini, kita dapat melakukan sesuatu terhadapnya. Atau singkat dapat dikelola (manage) dengan baik.

Kedua hal tersebut sangat penting terutama bagi Anda yang berangkat dari SQL ataupun bahasa programming lain, dimana ada satu representasi missing value saja, yaitu NULL.

Sepanjang bab ini kita akan membahas representasi yang bisa dianggap missing value yaitu NA, NULL dan NaN. Dan bagaimana kita melakukan banyak hal dengan missing value ini.

Klik tombol Next untuk melanjutkan.

# Kenapa kita memerlukan representasi Missing Value?

*Kenapa kita memerlukan representasi Missing Value?*

- ☒ Agar kita bisa membuang data yang tidak relevan
- ☒ Agar data tersebut bisa diisi dengan nilai lain
- ☒ Agar kita dapat membedakan antara data yang hilang dan hasil operasi matematika yang tidak mungkin dilakukan
- ☐ Agar kita bisa membaca ulang sumber data
- ☐ Tidak ada jawaban yang benar

# Apa saja yang bisa menyebabkan Missing Value?

Apa saja yang bisa menyebabkan Missing Value?

- ☐ Dari hasil proses konversi data yang tidak berhasil / error
- ☐ Karena mengakses posisi data pada indeks yang memang tidak memiliki data
- ☐ Karena perhitungan matematika yang tidak dapat diproses sistem
- ☐ Dari sumber data memang sudah tidak ada nilainya
- ☒ Seluruh jawaban di atas benar

# NA (Not Available)

NA adalah representasi utama dari *missing value* di R, dan merupakan suatu nilai atomik.

Artinya posisi NA sama dengan contoh nilai-nilai atomik bilangan seperti 1, 2 atau 100. Juga sama dengan contoh nilai karakter seperti "a", "b", atau "g".

Karena angka dan karakter seperti contoh di atas sering disebut sebagai konstanta, maka NA adalah konstanta untuk *missing value*.

Dengan demikian, secara singkat NA adalah:

- ☐ Representasi *missing value*
- ☐ Merupakan konstanta atau nilai atomik

NA tidak bisa digunakan sebagai variable karena merupakan keyword, dan perhatikan penulisan NA dimana kedua karakter adalah huruf besar.

Klik tombol Next untuk melanjutkan ke bagian praktek dan latihan.

# NA dan default type

Untuk menggunakan NA adalah sama dengan nilai konstanta lain, cukup mengetikkan NA.

Sebagai contoh, jika kita ketikkan NA di console maka akan muncul hasil berikut:

```
> NA  
[1] NA
```

Angka [1] adalah posisi index dari konstanta (yang selalu bernilai 1) dan NA adalah nilainya sendiri. Dan NA ini secara default adalah representasi missing value untuk tipe data logical.

Ini kita bisa cek dengan perintah yang menggunakan function typeof, sebagai berikut.

```
typeof(NA)
```

Hasilnya akan muncul sebagai berikut:

```
> typeof(NA)  
[1] "logical"
```

Tentunya, NA adalah representasi missing value untuk seluruh tipe lain, tapi secara default adalah untuk logical dulu. Namun akan kita jelaskan lebih jauh di bagian lain pada bab ini.

## Tugas Praktek

Cobalah ketik nilai NA pada bagian [...1...] dan typeof(NA) pada bagian [...2...] dari code editor dan jalankan.

Jika berjalan dengan lancar, maka hasilnya akan terlihat seperti pada contoh output dari code Lesson di atas.

Code Editor

#Ketik nilai NA

NA # [...1...]

#Tampilkan type dari NA dengan function typeof

typeof(NA) # [...2...]

## Console

```
> #Ketik nilai NA
> NA #[...1...]
[1] NA

> #Tampilkan type dari NA dengan function typeof
> typeof(NA) #[...2...]
[1] "logical"
```

# Menggunakan function is.na

Pada saat kita membandingkan nilai atomik biasanya kita menggunakan operator ==, misalkan untuk memeriksa suatu variabel x yang bernilai 3 – yang sebelumnya kita juga telah memasukkan angka yang sama – kita bisa melakukan hal berikut.

```
x <- 3
x == 3
```

Maka hasil eksekusi perintah di atas akan menghasilkan nilai TRUE seperti terlihat dari potongan code lengkap berikut.

```
x <- 3
> x == 3
[1] TRUE
```

Nah, ini tidak berlaku untuk NA. Jadi jika nilai x kita isi dan cek dengan cara yang sama sebagai berikut.

```
x <- NA
x == NA
```

Hasil eksekusinya tidak akan menghasilkan TRUE, tapi tetap NA.

Pengecekan seperti ini tentunya tidak mungkin dilakukan jika kita menggunakan konstruksi percabangan keputusan seperti if ataupun konstruksi perulangan seperti while.

Nah, untuk mengatasi hal ini R menggunakan function bernama is.na dengan input berupa konstanta atau variabel. Mari kita ubah simbol == contoh di atas menggunakan function is.na seperti berikut.

```
x <- NA
is.na(x)
```

Kali ini hasil eksekusinya akan bernilai TRUE.

## Tugas Praktek

Cobalah definisikan variable x yang diisi dengan missing value NA pada bagian [...1...], dan masukkan konstruksi perbandingan antara variable x dengan nilai NA dengan simbol == pada bagian [...2...].

Dan terakhir, masukkan perbandingan dengan function `is.na` untuk menggantikan `[...3...]` dari code editor dan jalankan.

Jika berjalan dengan lancar, maka hasilnya akan terlihat seperti contoh output dari tiap code dari Lesson

#### Code Editor

#Buat variable x yang diisi dengan nilai NA

```
x <- NA #[...1...]
```

#Pengecekan variable x dengan nilai NA menggunakan operator ==

```
x == NA #[...2...]
```

#Pengecekan variable x dengan nilai NA menggunakan function `is.na`

```
is.na(x) #[...3...]
```

#### Console

```
> #Buat variable x yang diisi dengan nilai NA
> x <- NA #[...1...]

> #Pengecekan variable x dengan nilai NA menggunakan operator ==
> x == NA #[...2...]
[1] NA

> #Pengecekan variable x dengan nilai NA menggunakan function is.na
> is.na(x) #[...3...]
[1] TRUE
```



## Variasi NA dan is.na

Seperti disebutkan sebelumnya, NA adalah merupakan nilai atomik dan konstanta, dan awalnya merupakan representasi missing value dari logical.

Apa artinya?

Ini jika dihubungkan dengan tipe data lain di R seperti vector – dimana isinya harus seragam tipe datanya – maka vector tersebut hanya boleh berisi nilai TRUE dan FALSE.

Sebagai perbandingan, mari kita lihat contoh pembuatan satu vector dengan isi campuran dari integer dan logical berikut.

```
c(1, 2, FALSE, 3)
```

Hasil akan ditampilkan sebagai berikut.

```
[1] 1 2 0 3
```

Terlihat bahwa FALSE di atas dikonversi (beradaptasi) menjadi 0. Adaptasi ini juga disebut dengan *coerce* di dokumentasi R.

Lalu bagaimana dengan missing value NA?

NA juga memiliki konstanta yang digunakan untuk beradaptasi, yaitu:

- ☐ NA\_integer\_ untuk representasi tipe data "integer"
- ☐ NA\_real\_ untuk representasi tipe data "double"
- ☐ NA\_complex\_ untuk representasi tipe data "complex"
- ☐ NA\_character\_ untuk representasi tipe data "character"

Dan semuanya memiliki nilai NA. Jika kita cek tipe data dari salah satu variasi missing value NA sebagai berikut

```
typeof(NA_real_)
```

Maka kita akan dapatkan hasil "double" seperti berikut.

```
> typeof(NA_real_)
[1] "double"
```

Namun jika menggunakan is.na maka seluruhnya akan mengembalikan hasil TRUE.

Untuk lebih jelasnya mari kita lakukan praktek berikut.

## Tugas Praktek

Cobalah ketikkan empat function typeof untuk menggantikan bagian [...1...] s/d [...4...] code editor, dengan masing-masing secara berurut mengecek NA\_integer\_, NA\_real\_, NA\_complex\_ dan NA\_character\_.

Kemudian ketikkan lagi function is.na untuk menggantikan bagian [...5...] s/d [...8...] code editor secara berurut untuk NA\_integer\_, NA\_real\_, NA\_complex\_ dan NA\_character\_.

Jika berhasil dijalankan, masing-masing baris perintah akan mengeluarkan hasil berikut.

```
[1] "integer"
[1] "double"
[1] "complex"
[1] "character"
[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
```

### Code Editor

```
typeof(NA_integer_) # [...1...]
typeof(NA_real_) # [...2...]
typeof(NA_complex_) # [...3...]
typeof(NA_character_) # [...4...]
is.na(NA_integer_) # [...5...]
is.na(NA_real_) # [...6...]
is.na(NA_complex_) # [...7...]
is.na(NA_character_) # [...8...]
```

### Console

```
> typeof(NA_integer_) # [...1...]
[1] "integer"

> typeof(NA_real_) # [...2...]
[1] "double"
```

```
> typeof(NA_complex_) #[...3...]
[1] "complex"

> typeof(NA_character_) #[...4...]
[1] "character"

> is.na(NA_integer_) #[...5...]
[1] TRUE

> is.na(NA_real_) #[...6...]
[1] TRUE

> is.na(NA_complex_) #[...7...]
[1] TRUE

> is.na(NA_character_) #[...8...]
[1] TRUE
```

# Coercion pada Vector yang mengandung NA

Dengan adanya variasi NA yang dijelaskan pada content sebelumnya, R akan secara otomatis menggantikan NA yang secara default tipe data logical menjadi variasi tersebut.

Jadi jika kita mengetik hasil berikut.

```
c(1, 2, NA, 4, 5)
```

maka sebenarnya konstanta NA di atas akan diganti menjadi NA\_real\_. Namun tentunya ini ga akan terlihat, hanya terjadi sebagai proses di belakang layar oleh R.

## Tugas Praktek

Isi bagian [...1...] pada code editor dengan variable bernama **isi.vector** yang isinya `c(1,2,3,NA,3,1)`. Kemudian kita akan menggunakan konstruksi **lapply** untuk mengecek semua tipe, isi [...2...] dengan variable yang telah kita buat, dan [...3...] dengan function **typeof**.

Terakhir isi bagian [...4...] dengan function **is.na** dan isi dengan variable **isi.vector**.

Jika berjalan dengan baik maka deretan hasil yang muncul adalah sebagai berikut.

```
[[1]]  
[1] "double"  
  
[[2]]  
[1] "double"  
  
[[3]]  
[1] "double"  
  
[[4]]  
[1] "double"  
  
[[5]]  
[1] "double"  
  
[[6]]
```

```
[1] "double"
[1] FALSE FALSE FALSE TRUE FALSE FALSE
```

### Code Editor

#Membuat vector bernama isi.vector dengan isi bilangan, dimana salah satunya memiliki missing value

```
isi.vector <- c(1,2,3,NA,3,1) #[...1...]
```

#Mengecek keseluruhan tipe data dengan perulangan lapply dan typeof

```
lapply(isi.vector, typeof) #[...2...],[...3...]
```

#Menggunakan is.na untuk mengecek keberadaan missing value dari tiap elemen pada vector

```
is.na(isi.vector) #[...4...]
```

### Console

```
> #Membuat vector bernama isi.vector dengan isi bilangan, dimana salah satunya memiliki
ki missing value
> isi.vector <- c(1,2,3,NA,3,1) #[...1...]

> #Mengecek keseluruhan tipe data dengan perulangan lapply dan typeof
> lapply(isi.vector, typeof) #[...2...],[...3...]
[[1]]
[1] "double"

[[2]]
[1] "double"

[[3]]
[1] "double"

[[4]]
[1] "double"

[[5]]
[1] "double"

[[6]]
[1] "double"

> #Menggunakan is.na untuk mengecek keberadaan missing value dari tiap elemen pada ve
ctor
> is.na(isi.vector) #[...4...]
[1] FALSE FALSE FALSE TRUE FALSE FALSE
```

# NULL

NULL adalah nilai yang banyak digunakan di bahasa pemrograman lain ataupun SQL (Structured Query Language) untuk merepresentasikan objek yang tidak ada atau null object (missing object) atau nilai yang tidak ada (missing value).

Di dalam R, missing value telah direpresentasikan dengan NA. Nah, untuk missing object inilah kita menggunakan NULL di R.

Atau lebih detilnya, NULL adalah suatu object khusus di R yang digunakan untuk merepresentasikan nilai atau object yang tidak terdefiniskan.

Untuk lebih jelasnya, kita akan melakukan praktek dan latihan yang berkaitan dengan NULL pada bagian berikutnya.

Klik tombol Next untuk melanjutkan.

# NULL dan Vector

Untuk memahami NULL, dan perbedaan dengan NA kita langsung mempraktekkannya dengan mengisi suatu vector dengan NULL.

Berikut adalah contoh variabel vector yang mengandung 7 elemen termasuk NA dan NULL.

```
isi.vector  
<- c(1, 2, 3, NA, 5, NULL, 7)
```

Jika kita menampilkan isinya, maka akan terlihat hasil sebagai berikut.

```
> isi.vector  
[1] 1 2 3 NA 5 7
```

Terlihat dari tampilan, tidak ada NULL disana. Mari kita pastikan lagi dengan menghitung jumlah isi dari vector dengan function length.

```
> length(isi.vector)  
[1] 6
```

Hasilnya adalah 6, padahal kita memasukkan 7 elemen. Dengan demikian terlihat bahwa NULL memang mewakili undefined object dan tidak dianggap oleh vector. Dengan demikian tidak menjadi bagian dari vector.

## Tugas Praktek

Buat variable bernama isi.vector dengan isi c(1, 2, 3, NA, 5, NULL, 7) untuk menggantikan bagian [...1...] dari code editor.

Kemudian hitung panjang dari isi.vector dengan function length, untuk menggantikan bagian [...2...] dari code editor.

### Code Editor

```
#Membuat vector dengan 7 elemen termasuk NA dan NULL
```

```
isi.vector <- c(1,2,3,NA,5,NULL,7) #[...1...]
```

```
#Menghitung jumlah elemen dari isi.vector
```

```
length(isi.vector) #[...2...]
```

## Console

```
> #Membuat vector dengan 7 elemen termasuk NA dan NULL
> isi.vector <- c(1,2,3,NA,5,NULL,7) #[...1...]

> #Menghitung jumlah elemen dari isi.vector
> length(isi.vector) #[...2...]
[1] 6
```



# Ringkasan Perbandingan NA dan NULL

Kita telah membahas mengenai NA dan NULL pada teks dan praktek sebelumnya. Untuk memudahkan pemahaman, berikut adalah rangkuman perbedaan terhadap keduanya.

	NA	NULL
Apa itu?	Adalah sebuah konstanta (variable yang tidak bisa diubah lagi nilainya) sebagai indikator dari missing value	Sebuah object untuk merepresentasikan sesuatu yang "tidak terdefinisi"
Kata kunci	Indikator	tidak terdefinisi
Panjang (length) dari object	1	0
Tipe	Logical	object
Variasi	NA dapat berubah menjadi representasi variasi missing value dengan daftar berikut: <ul style="list-style-type: none"> <li><input type="checkbox"/> NA_integer_ untuk tipe data "integer"</li> <li><input type="checkbox"/> NA_real_ untuk tipe data "double"</li> <li><input type="checkbox"/> NA_complex_ untuk tipe data "complex"</li> <li><input type="checkbox"/> NA_character_ untuk tipe data "character"</li> </ul>	Tidak ada variasi
Contoh kasus ketika kita akan menemukan object ini?	Ketika kita membaca data yang tidak memiliki nilai	Ketika mengakses posisi index yang tidak terdapat pada suatu list

Keterangan:

- Artinya NA adalah representasi missing value yang "masih" memiliki nilai logika – yang berfungsi sebagai indikator.
- NULL sendiri adalah objek tidak berbentuk, maka itu tidak terdefinisi dan panjangnya 0.

Salah satu operasi yang akan menghasilkan NULL adalah pada saat kita mengakses nama element yang tidak terdapat di dalam suatu **list**.

Klik tombol Next untuk melanjutkan.

# NULL dan List

Berbeda dengan vector, list akan menyimpan NULL apa adanya. Sebagai contoh, perintah berikut membuat list dengan 5 elemen dimana terdapat NULL dan NA di dalam list ini.

```
isi.list <- list(1, NULL, 3, NA, 5)
```

Jika kita tampilkan isinya, maka akan terlihat sebagai berikut.

```
> isi.list
[[1]]
[1] 1

[[2]]
NULL

[[3]]
[1] 3

[[4]]
[1] NA

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

Terlihat dari tampilan, NULL diisi pada list di indeks kedua. Dan dengan demikian, jika dihitung jumlah elemennya akan mendapatkan hasil 5.

```
> length(isi.list)
[1] 5
```

Kita akan coba ini pada tugas praktek berikut.

## Tugas Praktek

Buat variable bernama isi.list dengan isi list(1, NULL, 3, NA, 5) untuk menggantikan bagian [...1...] dari code editor.

Gantikan bagian [...2...] pada code editor dengan perintah menghitung jumlah elemen dengan function length dari isi.list.

### Code Editor

#Membuat list dengan 3 elemen termasuk NA dan NULL

```
isi.list <- list(1,NULL,3,NA,5) #[...1...]
```

#Menghitung jumlah elemen dari isi.list

```
length(isi.list) #[...2...]
```

### Console

```
> #Membuat list dengan 3 elemen termasuk NA dan NULL
> isi.list <- list(1,NULL,3,NA,5) #[...1...]

> #Menghitung jumlah elemen dari isi.list
> length(isi.list) #[...2...]
[1] 5
```

# Inf untuk mewakili Infinite Number

Setelah mengenalkan NA dan NULL sebagai representasi missing value dengan cukup detail pada bagian sebelumnya, sekarang kita masuk ke dua representasi angka yaitu **Inf** dan **NaN**.

**Inf** adalah singkatan dari infinite number atau representasi angka tidak terhingga. Semua pembagian angka bukan nol dengan nol adalah angka tidak terbatas, misalkan  $1/0$  atau  $-20/0$ .

Berikut adalah contoh operasi pembagian dengan 0 dan hasilnya di R.

```
> 1/0
[1] Inf

> -20/0
[1] -Inf
```

Mari kita praktekkan hal ini agar lebih jelas.

## Tugas Praktek

Ketiklah kalkulasi  $5/0$  untuk menggantikan [...1...] dan kalkulasi  $-120/0$  untuk menggantikan [...2...] pada code editor.

### Code Editor

#Hitung kalkulasi 5 dibagi dengan 0

5/0 [...1...]

#Hitung kalkulasi -120 dibagi dengan 0

-120/0 [...2...]

### Console

```
> #Hitung kalkulasi 5 dibagi dengan 0
> 5/0 [...1...]
[1] Inf

> #Hitung kalkulasi -120 dibagi dengan 0
> -120/0 [...2...]
[1] -Inf
```

# NaN (Not a Number)

NaN adalah representasi khusus untuk angka. Singkatan dari Not a Number, ini adalah representasi operasi matematika yang tidak bisa diwakili oleh angka apapun. Sering sekali bisa dianggap missing value tapi untuk hasil perhitungan.

Contoh perhitungan yang menghasilkan NaN adalah ketika angka 0 dibagi dengan 0. Ini tidak bisa dirasionalkan, sehingga NaN adalah representasi yang paling tepat.

Berikut adalah contoh perhitungan 0/0 di R dan hasilnya.

```
> 0/0  
[1] NaN
```

Mari kita praktekkan hal ini agar lebih jelas.

## Tugas Praktek

Ketiklah kalkulasi 0/0 untuk menggantikan [...1...] pada code editor.

### Code Editor

#Hitung kalkulasi 0 dibagi dengan 0

0/0 #[...1...]

### Console

```
> #Hitung kalkulasi 0 dibagi dengan 0  
> 0/0 #[...1...]  
[1] NaN
```

# NaN dari hasil function log()

Perhitungan matematika lain yang juga menghasilkan NaN adalah nilai logaritma (log) dari bilangan negatif.

Berikut adalah contoh perhitungan logaritma dari dua bilangan, 100 dan -100 di R dan hasilnya.

```
> log(100)
[1] 4.60517

> log(-100)
[1] NaN
```

Terlihat log(-100) menghasilkan nilai NaN

## Tugas Praktek

Ketiklah fungsi logaritma dari angka -1000 untuk menggantikan bagian [...1...] dari code editor.

Code Editor

```
#Hitung logaritma dari angka -1000
log(-1000) #[...1...]
```

Console

```
> #Hitung logaritma dari angka -1000
> log(-1000) #[...1...]
[1] NaN
```

# Fungsi is.nan

Seperti halnya NA, jika kita membandingkan nilai NaN dengan menggunakan == maka hasilnya bukan TRUE/FALSE tapi menjadi NA, seperti pada contoh berikut.

```
> NaN == NaN  
[1] NA
```

Untuk membandingkan NaN kita bisa gunakan function is.nan, seperti pada contoh berikut.

```
> is.nan(NaN)  
[1] TRUE
```

## Tugas Praktek

Buat variable bernama **contoh.nan** dengan perhitungan 0/0 untuk menggantikan bagian [...1...] dari code editor. Kemudian cek variabel dengan fungsi **contoh.nan** dengan mengganti code bagian [...2...].

### Code Editor

```
#Buat variable contoh.nan  
contoh.nan <- 0/0 [...1...]  
#Periksa dengan function is.nan  
is.nan(contoh.nan) [...2...]
```

### Console

```
> #Buat variable contoh.nan  
> contoh.nan <- 0/0 [...1...]  
  
> #Periksa dengan function is.nan  
> is.nan(contoh.nan) [...2...]  
[1] TRUE
```



# NaN dan is.na versus NA dan is.nan

Seluruh nilai NaN dapat diperiksa dengan fungsi `is.na`, sehingga hasilnya mengembalikan TRUE. Berikut adalah contohnya.

```
> is.na(NaN)
[1] TRUE
```

Dengan demikian, secara umum pada R bisa dikatakan NaN juga adalah missing value.

Namun tidak sebaliknya, semua nilai NA tidak akan mengembalikan nilai TRUE jika diperiksa dengan fungsi `is.nan`.

```
> is.nan(NA)
[1] FALSE
```

Dengan level yang jelas seperti ini, kita bisa memisahkan mana missing values dan kemudian mana saja bagian yang merupakan perhitungan yang menghasilkan angka yang tidak bisa didefinisikan (not a number).

## Tugas Praktek

Isi code editor berikut dengan dua contoh persis dari lesson di atas, masing-masing untuk fungsi **is.na** dan **is.nan**.

### Code Editor

#Masukkan code di bawah ini sesuai permintaan soal

```
is.na(NaN)
```

```
is.nan(NA)
```

### Console

```
> #Masukkan code di bawah ini sesuai permintaan soal
> is.na(NaN)
[1] TRUE

> is.nan(NA)
[1] FALSE
```

# Menghitung Jumlah Missing Values dari satu Vector

Setelah mengenal semua representasi missing value – dalam hal ini NA dan NaN – dan fungsi untuk mengidentifikasinya, kita dapat melakukan banyak hal.

Salah satunya adalah mengetahui jumlah missing value yang terdapat pada data kita. Akan banyak contoh pada bagian-bagian berikutnya, tapi untuk memulai kita gunakan contoh vector.

Untuk melakukan perhitungan banyaknya nilai missing value di vector, kita gunakan gabungan function `sum` dan `is.na` sebagai berikut:

```
sum(is.na( variable_vector ) == TRUE)
```

Penjelasan konstruksi tersebut adalah sebagai berikut.

Komponen	Keterangan
<code>sum(...)</code>	Function untuk menjumlahkan banyaknya elemen
<code>is.na(...) == TRUE</code>	Konstruksi untuk mengidentifikasi elemen mana yang merupakan missing value
<code>variable_vector</code>	Adalah variable yang isinya adalah vector dengan deklarasi <code>c(...)</code>

## Tugas Praktek

Pada code editor berikut terdapat satu variable bernama `isi.vector`. Hitunglah jumlah elemen bernilai missing value dengan perintah yang akan menggantikan bagian [...]

### Code Editor

#Masukkan code di bawah ini sesuai permintaan soal

```
isi.vector <- c(1,2,NA,4,5,NaN,6)
```

```
sum(is.na(isi.vector)==TRUE) #[...]
```

### Console

```
> #Masukkan code di bawah ini sesuai permintaan soal
> isi.vector <- c(1,2,NA,4,5,NaN,6)

> sum(is.na(isi.vector)==TRUE) #[...]
[1] 2
```

# Kesimpulan

Selamat, dengan telah melewati bab ini maka Anda telah menguasai dasar untuk mengenal dan menangani missing value.

Ini penting karena tanpa pengetahuan yang cukup mengenai missing value ini, Anda akan mengalami kesulitan ketika berhadapan dengan beragam jenis data dan proses perhitungan yang pada kenyataannya memang tidak rapi.

Mari kita rangkum apa yang telah Anda tempuh sejauh ini:

- ☐ Mengetahui representasi missing value di R, yaitu NA (Not Available)
- ☐ NA dan variasinya (`NA_integer_`, `NA_real_`, `NA_complex_`, dan `NA_character_`)
- ☐ NULL sebagai representasi missing object
- ☐ Perilaku NA dan NULL untuk vector dan list
- ☐ NaN sebagai representasi missing value angka (Not a Number)
- ☐ Penggunaan function `is.na` dan `is.nan`

Dengan bekal materi tersebut, kita telah siap untuk mengolah data pada saat membaca berbagai variasi data yang memiliki missing value.

Namun sebelum itu kita perlu menguasai lagi satu struktur data penting di R, yaitu `factor`.

Klik tombol Next untuk melanjutkan ke bab mengenai `Factor`.

# Pendahuluan

Factor adalah suatu tipe variabel yang intensif digunakan di R – dengan demikian menjadi sangat penting untuk mempelajari Factor ini.

Walaupun awalnya kelihatan sederhana, namun Factor adalah salah satu bagian dari R yang paling sering ditanyakan. Penyebabnya karena fungsi analisa dan grafik yang banyak memanfaatkan factor dan kadang hasilnya tidak sesuai – contoh kasus dan solusinya termasuk yang akan dibahas pada bab ini.

Secara keseluruhan, berikut adalah apa yang akan Anda pelajari mengenai factor pada bab ini:

- ☐ Apa itu factor dan kenapa diperlukan?
- ☐ Kapan factor digunakan
- ☐ Cara membuat factor di R
- ☐ Bagaimana sebenarnya nilai disimpan di dalam factor
- ☐ Menggunakan function untuk merubah nilai factor
- ☐ Merubah pengurutan di factor

Dengan menguasai keterampilan menggunakan factor ini, Anda akan lebih siap untuk melakukan banyak hal di R.

Klik tombol Next untuk melanjutkan.

# Apa itu Factor?

**Factor** adalah suatu variable di R yang menyimpan daftar nilai-nilai kategori atau nominal.

Berikut adalah beberapa contoh daftar nilai kategori yang bisa disimpan dalam factor:

- ☐ Nama bulan: Januari, Februari, Maret.
- ☐ Jenis pakaian: Atasan, Jeans, Rok, Kaos.
- ☐ Satuan berat: kg, mg, ton, kwintal.
- ☐ dan lain-lain.

Dengan demikian, factor menyimpan nilai-nilai yang terbatas (diskrit). Nama bulan terdiri dari 12 nilai yaitu dari Januari s/d Desember.

Jika tidak terbatas atau kontinu, misalkan angka berat seperti 64.5 kg, 11.2 kg, 80.39 kg, dan seterusnya – maka ini tidak bisa digolongkan sebagai *factor*.

Karena sifat data dengan nilai terbatas ini, factor sering disebut juga sebagai *categorical variable* atau variabel kategorik.

Klik tombol Next untuk melanjutkan.

# Mana pernyataan berikut yang benar mengenai factor?

*Mana pernyataan berikut yang benar mengenai factor?*

- ☒ Factor adalah variable dengan nilai terbatas
- ☒ Factor adalah variabel kategorik atau categorical variable
- ☐ Factor adalah tipe data vector
- ☐ Factor adalah variable dengan nilai kontinu
- ☐ Factor adalah tipe data yang mirip dengan data.frame

# Mana yang tidak termasuk daftar nilai yang bisa disimpan factor?

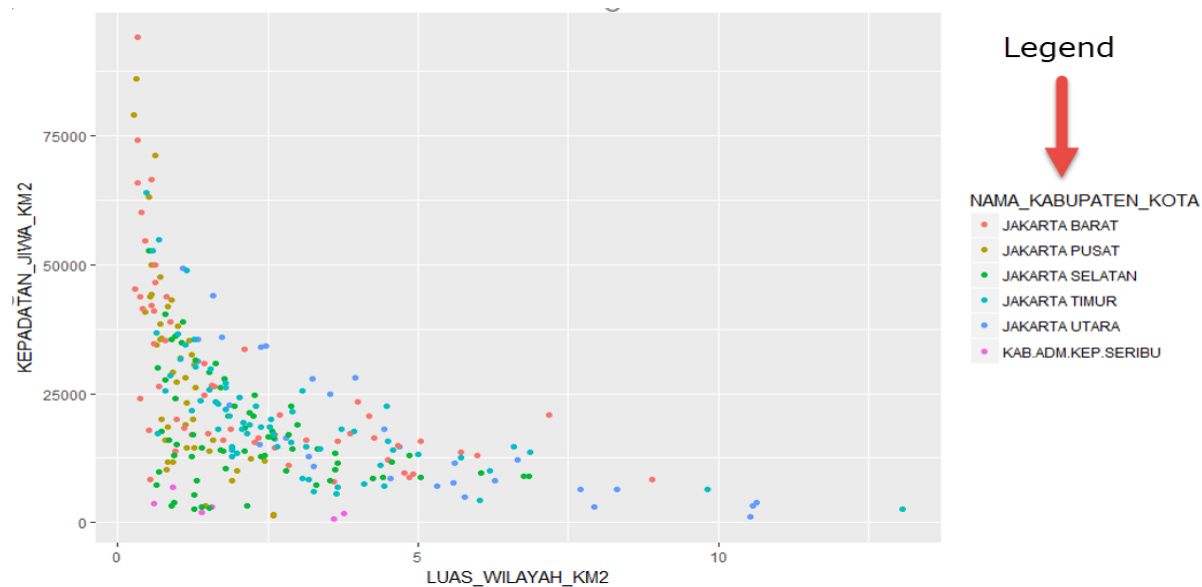
*Mana yang tidak termasuk daftar nilai yang bisa disimpan factor?*

- ☒ 3.14 s/d 1001.00
- ☐ Manchester United, Liverpool, Arsenal, Chelsea
- ☐ Jeans, Kaos, Atasan
- ☐ Januari, Februari, Maret
- ☐ Universitas Multimedia Nusantara, DQLab, Skystar Ventures

# Kenapa Factor, dan Kapan Sebaiknya Digunakan?

**Factor** adalah variabel yang sangat penting untuk digunakan pada kasus analisa statistik, menggambar grafik (*plotting*), pemodelan regresi ataupun *machine learning*.

Sebagai contoh kasus, maka fungsi plotting pada library ggplot2 akan mampu mengambil variabel sebagai *legend* jika direpresentasikan sebagai factor.



Banyaknya function yang menggunakan factor dikarenakan dengan adanya factor ini maka nilai kategoris lebih mudah diolah dan dianalisa.

Kita sebaiknya menggunakan factor ketika kita ingin

- ☐ memastikan bahwa variable yang kita gunakan memiliki data terbatas (diskrit)
- ☐ dan menginformasikan kepastian ini kepada function atau model analisa yang kita gunakan

Klik tombol Next untuk melanjutkan.



# Kenapa Factor menjadi penting di R?

*Kenapa Factor menjadi penting di R?*

- ☒ Banyak digunakan oleh fungsi analisa dan plotting di R
- ☒ Digunakan oleh ggplot2 untuk legend
- ☐ Bisa digunakan untuk perhitungan matematika dasar
- ☐ Memisahkan teks dan angka secara jelas
- ☐ Karena semua variabel harus dikonversi menjadi factor

# Kapan harusnya menggunakan factor?

*Kapan harusnya menggunakan factor?*

- ☐ Ketika kita butuh pengolahan nilai kontinu
- ☐ Ketika kita ingin melakukan konversi huruf kecil ke huruf besar
- ☐ Tidak ada jawaban yang benar
- ☐ Ketika kita ingin membuang spasi dari data teks
- ☒ Ketika kita butuh pengolahan nilai diskrit

# Membuat Factor di R

Factor dapat dibuat di R dengan function **factor**, dengan syntax paling sederhana sebagai berikut:

```
factor( x = vector)
```

atau

```
factor(vector)
```

Sebagai contoh, untuk membuat factor dari tiga nama bulan pertama pada kalender kita gunakan perintah berikut:

```
factor(c("Jan", "Feb", "Mar"))
```

Perintah di atas akan menampilkan hasil berikut:

```
[1] Jan Feb Mar
Levels: Feb Jan Mar
```

Terlihat ada dua hasil, yaitu nilai yang dimasukkan dan ada satu lagi output **Levels**. Berikut adalah penjelasan dari hasil di atas.

Hasil	Keterangan
[1] Jan Feb Mar	<b>[1]</b> adalah tampilan output dengan hasil tampilan dimulai pada indeks ke 1 <b>Jan Feb Mar</b> adalah tampilan dari nilai factor
Levels: Feb Jan Mar	Levels: adalah atribut yang menempel pada setiap faktor dan berisi daftar diskrit dari variasi nilai-nilai yang terdapat pada faktor.  Untuk kasus ini terdapat tiga nilai variasi yaitu Feb Jan Mar – dan <b>kebetulan</b> sama dengan jumlah nilai yang terdapat pada factor saat ini, tetapi dengan urutan yang berbeda dengan tampilan isian factor.  Pada levels, terlihat Feb dimulai duluan dibandingkan Jan karena secara urutan alfabet pertama pada Feb – yaitu F –

lebih kecil dibandingkan alfabet pertama pada Jan – yaitu J.
--

### Tugas Praktek

Pada code editor, buatlah satu factor dari suatu vector yang berisi nilai teks "Jan", "Feb" dan "Mar".

#### Code Editor

#Buatlah factor dengan isi nilai teks "Jan", "Feb", dan "Mar"

```
factor(c("jan", "Feb", "Mar"))
```

#### Console

```
> #Buatlah factor dengan isi nilai teks "Jan", "Feb", dan "Mar"
> factor(c("jan", "Feb", "Mar"))
[1] jan Feb Mar
Levels: Feb Mar jan
```

# Atribut levels dan class pada Factor

Perbandingan yang kontras antara factor dengan vector atau data.frame adalah pada factor terdapat atribut tambahan, yaitu levels.

Atribut sendiri adalah variable yang melekat dan menjadi bagian dari objek atau variable lain.

Selain atribut levels, pada factor terdapat juga atribut class.

Untuk melihat seluruh atribut pada kita menggunakan function attributes, berikut adalah contoh penggunaannya.

```
attributes(variable)
```

Dan nilai atribut tergantung penerapannya oleh R. Dan untuk levels, atributnya selalu bertipe karakter atau teks. Ini merupakan catatan penting yang sangat berguna dalam pemanfaatan factor lebih jauh.

Mari kita langsung lakukan praktek agar lebih jelas.

## Tugas Praktek

Pada code editor, terdapat satu variable bernama faktor.bulan yang telah diisi dengan factor. Tampilkan atribut dari variable ini memasukkan perintah pada bagian [...].

Jika semua berjalan lancar, maka hasilnya akan tampil sebagai berikut.

```
$levels
[1] "Feb" "Jan" "Mar"
$class
[1] "factor"
```

Dimana hasilnya terdapat dua atribut, yaitu **levels** dan **class** (ditandai dengan tanda \$ pada nama atribut di depannya) dengan nilai-nilai atributnya (perhatikan semua memiliki tanda kutip pada nilainya – yang menandakan nilainya berupa karakter atau teks).

Dan disini terlihat atribut class berisi "factor" – menandakan objek ini memang adalah sebuah factor.

## Code Editor

```
#Variable factor bernama faktor.bulan dengan nilai teks "Jan", "Feb", dan "Mar"
```

```
faktor.bulan <- factor(c("Jan","Feb","Mar"))
```

```
attributes(faktor.bulan) #[...]
```

## Console

```
> #Variable factor bernama faktor.bulan dengan nilai teks "Jan", "Feb", dan "Mar"
> faktor.bulan <- factor(c("Jan","Feb","Mar"))

> attributes(faktor.bulan) #[...]
$levels
[1] "Feb" "Jan" "Mar"

$class
[1] "factor"
```

# Function levels dan class pada Factor

Atribut levels dan class dapat juga dilihat dengan function levels dan class dengan input berupa factor, dengan konstruksi sebagai berikut:

```
levels(variable)
class(variable)
```

Mari kita langsung lakukan praktek agar lebih jelas.

## Tugas Praktek

Pada code editor, terdapat satu variable bernama faktor.bulan yang telah diisi dengan factor. Tampilkan levels dari variable dengan perintah yang menggantikan bagian [...1...]. Kemudian menampilkan class dari variable dengan perintah yang menggantikan bagian [...2...].

Jika semua berjalan lancar, maka akan muncul dua hasil dari masing-masing perintah yang dimasukkan sebagai berikut.

```
[1] "Feb" "Jan" "Mar"
[1] "factor"
```

## Code Editor

#Variable factor bernama faktor.bulan dengan nilai teks "Jan", "Feb", dan "Mar"

```
faktor.bulan <- factor(c("Jan","Feb","Mar"))
```

```
levels(faktor.bulan) #[...1...]
```

```
class(faktor.bulan) #[...2...]
```

## Console

```
> #Variable factor bernama faktor.bulan dengan nilai teks "Jan", "Feb", dan "Mar"
> faktor.bulan <- factor(c("Jan","Feb","Mar"))

> levels(faktor.bulan) #[...1...]
[1] "Feb" "Jan" "Mar"

> class(faktor.bulan) #[...2...]
[1] "factor"
```

# Perulangan Nilai pada Factor

Jika pada praktek sebelumnya, factor diisi dengan tiga contoh nilai nama bulan pertama yang tidak berulang. Pada praktek kali ini kita coba akan memasukkan nilai yang berulang, dimana "Jan" dan "Mar" akan dimasukkan berulang.

Berikut adalah contohnya:

```
factor(c("Jan", "Feb", "Mar", "Jan", "Mar", "Jan"))
```

Jika dieksekusi maka hasilnya adalah sebagai berikut:

```
[1] Jan Feb Mar Jan Mar Jan  
Levels: Feb Jan Mar
```

Berikut adalah penjelasan hasilnya.

Hasil	Keterangan
[1] Jan Feb Mar Jan Mar Jan	<b>[1]</b> adalah tampilan output dengan hasil tampilan dimulai pada indeks ke 1 <b>Jan Feb Mar Jan Mar Jan</b> adalah tampilan dari nilai-nilai factor
Levels: Feb Jan Mar	Levels: adalah atribut yang menempel pada setiap faktor dan berisi daftar diskrit dari variasi nilai-nilai yang terdapat pada faktor.  Variasi nilainya masih sama, yaitu tiga (Feb Jan Mar).

## Tugas Praktek

Pada code editor, masukkan code seperti contoh pada Lesson dan jalankan. Jika sudah selesai, tekan tombol Submit untuk melanjutkan ke lesson selanjutnya.

Code Editor

#Buatlah factor dengan teks "Jan", "Feb", "Mar", "Jan", "Mar", dan "Jan"

```
factor(c("Jan", "Feb", "Mar", "Jan", "Mar", "Jan"))
```



## Console

```
> #Buatlah factor dengan teks "Jan", "Feb", "Mar","Jan","Mar", dan "Jan"  
> factor(c("Jan","Feb","Mar","Jan","Mar","Jan"))  
[1] Jan Feb Mar Jan Mar Jan  
Levels: Feb Jan Mar
```

# Bagaimana sebenarnya Factor disimpan?

Di “belakang layar”, factor sebenarnya memiliki 2 bagian:

- Vector yang berisi nilai angka bulat (integer).
- Vector yang berisi nilai-nilai kategori, ini disimpan dalam atribut levels – isinya selalu bertipe character / teks.

Untuk penjelasannya, mari kita perhatikan ilustrasi dari perintah factor berikut:

```
factor(c("Jan","Feb","Mar"))
```

Proses ini diilustrasikan pada gambar berikut. Untuk penjelasan, lihat paragraph setelah gambar ini.

## Bagaimana Factor disimpan di R? (bagian 1)

<https://www.dqlab.id>

1 factor(c("Jan","Feb","Mar"))

real value | attribute

3 Factor

1	2
2	1
3	3

2 Levels

1	"Feb"
2	"Jan"
3	"Mar"

Proses dari terjemahan perintah di atas:

1. R menerima perintah dengan function factor(c("Jan","Feb","Mar"))
2. R akan mencari variasi nilai (levels) dan diurutkan – dalam hal ini pengurutan alfabet – dan dipetakan berdasarkan index yang bernilai integer. Disini nilai index 1 mewakili “Feb”, 2 mewakili “Jan”, dan 3 mewakili “Mar”
3. Dari levels, nilai-nilai “Jan”, “Feb”, “Mar” dicari nilai index-nya dan dimasukkan sebagai nilai-nilai pada factor ( 2, 1, 3 ).

Dan mari kita perhatikan satu lagi contoh ilustrasi dari pengembangan factor di atas:

## Bagaimana Factor disimpan di R? (bagian 2)

<https://www.dqlab.id>

1 `factor(c("Jan","Feb","Mar","Jan","Mar"))`



3 Factor

1	2
2	1
3	3
4	2
5	3

2 Levels

1	"Feb"
2	"Jan"
3	"Mar"

```
factor(c("Jan","Feb","Mar","Jan","Mar"))
```

Penjelasan dari proses di atas:

1. R menerima perintah dengan function `factor(c("Jan","Feb","Mar","Jan","Mar"))`
2. R akan mencari variasi nilai (levels) dan diurutkan – dalam hal ini pengurutan alfabet – dan dipetakan berdasarkan index yang bernilai integer. Disini nilai index 1 mewakili "Feb", 2 mewakili "Jan", dan 3 mewakili "Mar"
3. Dari levels, nilai-nilai "Jan", "Feb", "Mar", "Jan", "Mar" dicari nilai index-nya dan dimasukkan sebagai nilai-nilai pada factor ( 2, 1, 3, 2, 3).

Dengan demikian, kita simpulkan kembali dari ilustrasi di atas bahwa nilai dari factor sebenarnya adalah nilai bilangan bulat (integer) dengan nilai-nilai kategoris disimpan pada atribut **levels**.

Praktek-praktek selanjutnya akan memperkuat pemahaman Anda tentang hal ini.

Klik tombol Next untuk melanjutkan.

## Berapa nilai pada factor?

Berikut adalah nilai-nilai yang terdapat pada factor dengan perintah berikut  
`factor(c("Charlie", "Arief", "Budi", "Arief"))`?

- ☐ 2, 1, 3, 1
- ☒ 3, 1, 2, 1
- ☐ 1, 2, 3, 4
- ☐ 1, 2, 1, 2
- ☐ Tidak ada yang benar

# Penggunaan as.integer pada Factor

Untuk mengambil nilai-nilai index integer pada factor, kita bisa menggunakan function `as.integer` dengan syntax berikut.

```
as.integer(variable factor)
```

Mari kita langsung praktekkan dengan tugas berikut.

## Tugas Praktek

Pada code editor, ganti bagian [...] dengan function `as.integer` dan dengan input `variable factor.bulan`.

### Code Editor

```
#Buatlah factor dengan teks "Jan", "Feb", "Mar", "Jan", "Mar", dan "Jan"
factor.bulan <- factor(c("Jan", "Feb", "Mar", "Jan", "Mar", "Jan"))
as.integer(factor.bulan) #[...]
```

### Console

```
> #Buatlah factor dengan teks "Jan", "Feb", "Mar", "Jan", "Mar", dan "Jan"
> factor.bulan <- factor(c("Jan", "Feb", "Mar", "Jan", "Mar", "Jan"))
> as.integer(factor.bulan) #[...]
[1] 2 1 3 2 3 2
```

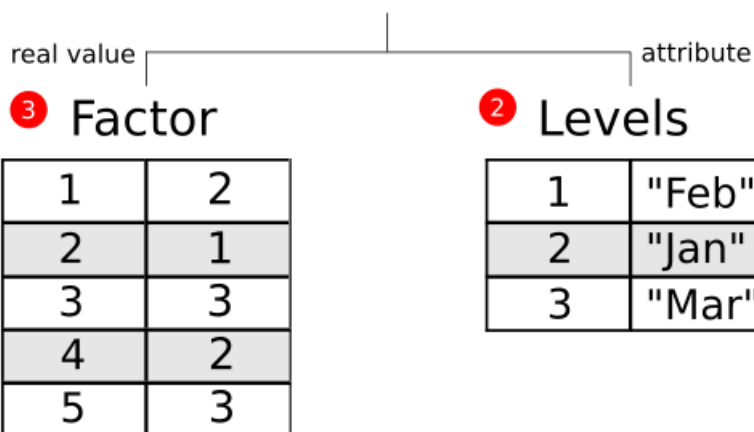
# Mengganti "Jan" menjadi "Januari"

Sering sekali nilai kategori tidak sesuai dengan yang kita inginkan, sebagai contoh kita ingin nilai "Jan" ditampilkan sebagai "Januari". Di factor, kita lakukan ini dengan mengganti nilai levels pada index yang kita inginkan.

Mari kita perhatikan contoh faktor berikut. Terdapat satu variabel factor.bulan dengan levels "Feb", "Jan", dan "Mar".

1

```
factor.bulan <- factor(c("Jan","Feb","Mar","Jan","Mar"))
```



Jika kita ingin ganti "Jan" (posisi ke 2 pada levels) menjadi "Januari", maka perintah yang kita gunakan adalah:

```
levels(factor.bulan)
[2] <- "Januari"
```

Berikut adalah penjelasan konstruksi perintah di atas.

Konstruksi	Keterangan
levels(...)	adalah fungsi untuk mengambil levels dari sebuah factor
factor.bulan	Variable factor
[2]	Posisi kedua – dalam hal ini posisi dari levels
<-	Operator assignment atau perintah untuk memasukkan data
"Januari"	Nilai yang akan dimasukkan ke levels posisi kedua

### Tugas Praktek

Pada code editor, ganti bagian [...1...] dengan perintah mengganti posisi ketiga dari levels dari "Mar" menjadi "Maret".

Kemudian ganti bagian [...2...] dengan perintah untuk menampilkan variable faktor.bulan.

Jika semua berjalan dengan baik, maka hasilnya akan tampak sebagai berikut.

```
[1] Januari Feb      Maret   Januari Maret   Januari
Levels: Feb Januari Maret
```

### Code Editor

#Buatlah factor dengan teks "Jan", "Feb", "Mar", "Jan", "Mar", dan "Jan"

```
factor.bulan <- factor(c("Jan","Feb","Mar","Jan","Mar","Jan"))
```

#Mengganti levels

```
levels(factor.bulan)[2] <- "Januari"#[...1...]
```

```
levels(factor.bulan)[3] <- "Maret" #[...2...]
```

```
factor.bulan
```

### Console

```
> #Buatlah factor dengan teks "Jan", "Feb", "Mar","Jan","Mar", dan "Jan"
> factor.bulan <- factor(c("Jan","Feb","Mar","Jan","Mar","Jan"))

> #Mengganti levels
> levels(factor.bulan)[2] <- "Januari"#[...1...]

> levels(factor.bulan)[3] <- "Maret" #[...2...]

> factor.bulan
[1] Januari Feb      Maret   Januari Maret   Januari
Levels: Feb Januari Maret
```

# Angka sebagai Kategori

Jika sebelumnya kita memasukkan teks sebagai nilai kategori pada saat mendefinisikan factor. Bagaimana dengan angka?

Jika angka dimasukkan ke dalam vector, tetap akan dikenal sebagai nilai kategoris dan dimasukkan ke dalam levels. Dan karena itu angka akan dikonversi menjadi teks.

## Tugas Praktek

Pada code editor, cobalah buat suatu variable factor bernama factor.umur dengan isi berupa vector c(12, 35, 24, 12, 35, 37) pada bagian [...1...].

Kemudian tampilkan variable factor.umur tersebut pada bagian [...2...].

Jika semua berjalan lancar, maka akan muncul hasil sebagai berikut.

```
[1] 12 35 24 12 35 37
Levels: 12 24 35 37
```

## Code Editor

```
#Buatlah factor bernama factor.umur dengan isi c(12, 35, 24, 12, 35, 37)
factor.umur <- factor(c(12,35,24,12,35,37)) #[...1...]
#Tampilkan variable factor.umur
factor.umur #[...2...]
```

## Console

```
> #Buatlah factor bernama factor.umur dengan isi c(12, 35, 24, 12, 35, 37)
> factor.umur <- factor(c(12,35,24,12,35,37)) #[...1...]

> #Tampilkan variable factor.umur
> factor.umur #[...2...]
[1] 12 35 24 12 35 37
Levels: 12 24 35 37
```



# NA, NaN, NULL pada saat pembentukan Factor

NA, NULL dan NaN adalah tiga nilai khusus untuk merepresentasikan missing values atau nilai yang hilang di R.

Jika ketiganya dimasukkan ke dalam factor melalui deklarasi vector, maka prinsip berikut tetap berlaku:

- ❑ Na dan NaN akan menjadi bagian dari isi factor, NULL akan dihilangkan
- ❑ Hanya NaN yang akan dikenali sebagai levels

Sebagai contoh, jika kita membuat factor dengan nilai berikut.

```
factor(c("Bandung", "Jakarta", NA, "Jakarta", NaN, "Medan", NULL, NULL, "Bandung"))
```

maka hasil tampilannya adalah sebagai berikut.

```
[1] Bandung Jakarta      Jakarta NaN      Medan  Bandung
Levels: Bandung Jakarta Medan NaN
```

Terlihat nilai NULL dibuang di tampilan isi factor, kemudian pada levels nilai NA juga dibuang.

## Tugas Praktek

Pada code editor, buatlah satu variabel dengan nama factor.lokasi dengan isi sama persis dengan contoh pada lesson di atas – ketikkan ini pada bagian [...1...].

Tampilkan variable factor tersebut pada bagian [...2...].

### Code Editor

```
#Buatlah variable factor.lokasi dengan isi berupa vector c("Bandung", "Jakarta", NA, "Jakarta", NaN, "Medan", NULL, NULL, "Bandung")
```

```
factor.lokasi <- factor (c("Bandung", "Jakarta", NA, "Jakarta", NaN, "Medan", NULL, NULL, "Bandung")) #[...1...]
```

```
#Tampilkan factor.lokasi
```

```
factor.lokasi #[...2...]
```

## Console

```
> #Buatlah variable factor.lokasi dengan isi berupa vector c("Bandung", "Jakarta", NA, "Jakarta", NaN, "Medan", NULL, NULL, "Bandung")
> factor.lokasi <- factor (c("Bandung", "Jakarta", NA, "Jakarta", NaN, "Medan", NULL, NULL, "Bandung")) #[...1...]

> #Tampilkan factor.lokasi
> factor.lokasi #[...2...]
[1] Bandung Jakarta <NA>      Jakarta NaN      Medan   Bandung
Levels: Bandung Jakarta Medan NaN
```

# Menghitung panjang Factor dengan length

Panjang factor dapat dihitung dengan menggunakan function length dengan syntax sederhana berikut.

```
length(variable)
```

Hanya nilai NULL yang tidak terhitung sebagai bagian dari factor.

## Tugas Praktek

Pada code editor, terdapat satu variable factor.lokasi. Hitunglah panjang dari variable ini dengan menggunakan perintah length. Ketikkan perintah ini untuk menggantikan bagian [...].

Jika semua berjalan dengan lancar, maka akan muncul hasil sebagai berikut.

```
[1] 7
```

Jadi panjang dari variable factor.lokasi adalah 7.

## Code Editor

```
#Buatlah variable factor.lokasi dengan isi berupa vector c("Bandung", "Jakarta", NA, "Jakarta", NaN, "Medan", NULL, NULL, "Bandung")
```

```
factor.lokasi <- factor(c("Bandung", "Jakarta", NA, "Jakarta", NaN, "Medan", NULL, NULL, "Bandung"))
```

```
#Tampilkan panjang dari variable factor.lokasi
```

```
length(factor.lokasi) #[...]
```

## Console

```
> #Buatlah variable factor.lokasi dengan isi berupa vector c("Bandung", "Jakarta", NA, "Jakarta", NaN, "Medan", NULL, NULL, "Bandung")
> factor.lokasi <- factor(c("Bandung", "Jakarta", NA, "Jakarta", NaN, "Medan", NULL, NULL, "Bandung"))
```

```
> #Tampilkan panjang dari variable factor.lokasi
> length(factor.lokasi) #[...]
[1] 7
```

# Menyusun levels dari awal

Sejauh ini factor yang kita buat seakan-akan hanya bisa dirubah isinya, namun bukan urutannya. Ini kita bisa kendalikan juga dengan memberikan nilai-nilai kategori sesuai urutan yang kita inginkan pada argumen levels di function factors.

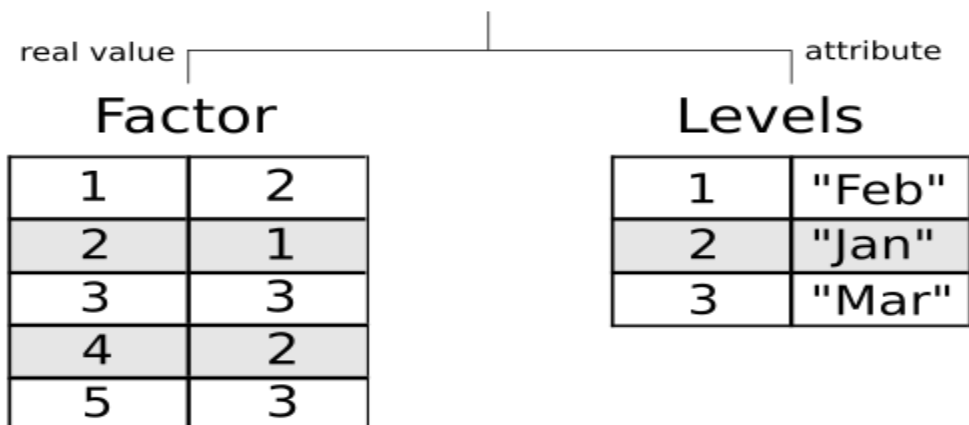
```
factor(..., levels = ...)
```

Sebagai contoh, agar levels bernilai urut dari "Jan" s/d "Mar" maka pada saat membuat factor kita harus sisipkan argumen labels sebagai berikut:

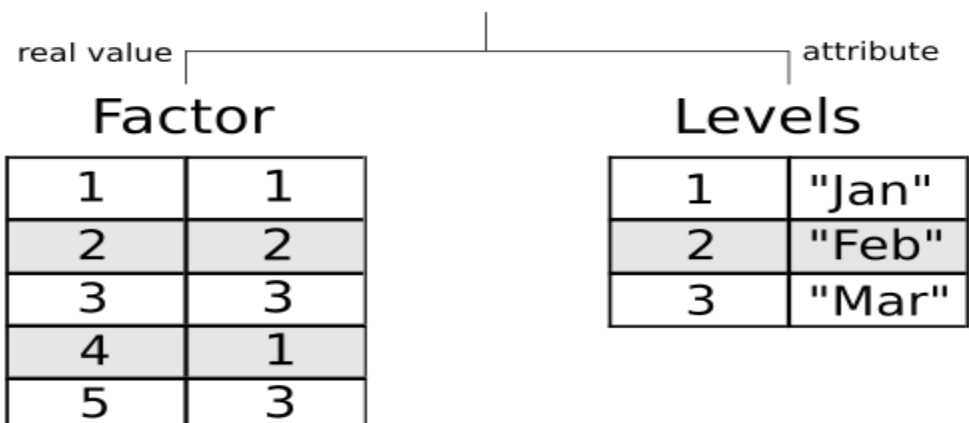
```
factor(..., levels = c("Jan", "Feb", "Mar"))
```

Cobalah perhatikan contoh ilustrasi dari dua pembuatan factor, dengan dan tanpa argumen levels.

❶ `factor.bulan <- factor(c("Jan", "Feb", "Mar", "Jan", "Mar"))`



❷ `factor.bulan <- factor(c("Jan", "Feb", "Mar", "Jan", "Mar"), levels = c("Jan", "Feb", "Mar"))`



### Tugas Praktek

Pada code editor, terdapat satu variable factor.lokasi, tambahkan argumen labels pada bagian [...] sehingga urutan labels menjadi "Jan","Feb","Mar".

Jika semua berjalan dengan lancar, maka akan muncul hasil sebagai berikut.

```
[1] Jan Feb Mar Jan Mar  
Levels: Jan Feb Mar
```

### Code Editor

```
#Variable factor dengan isi vector c("Jan","Feb","Mar","Jan","Mar")  
factor(c("Jan","Feb","Mar","Jan","Mar"), levels=c("Jan","Feb","Mar")) #[...]
```

### Console

```
> #Variable factor dengan isi vector c("Jan","Feb","Mar","Jan","Mar")  
> factor(c("Jan","Feb","Mar","Jan","Mar"), levels=c("Jan","Feb","Mar")) #[...]  
[1] Jan Feb Mar Jan Mar  
Levels: Jan Feb Mar
```

# Kesimpulan

Factor adalah suatu tipe variabel yang intensif digunakan di R – dengan demikian menjadi sangat penting untuk mempelajari Factor ini.

Dari praktek keseluruhan yang telah dilakukan, kita telah mencapai tujuan untuk mempelajari factor dari aspek berikut:

- ☐ Apa itu factor dan kenapa diperlukan?
- ☐ Kapan factor digunakan
- ☐ Cara membuat factor di R
- ☐ Bagaimana sebenarnya nilai disimpan di dalam factor
- ☐ Menggunakan function untuk merubah nilai factor
- ☐ Merubah pengurutan di factor

Dengan menguasai keterampilan menggunakan factor ini, Anda sudah lebih siap untuk melakukan banyak hal di R.

Klik tombol Next untuk melanjutkan ke bab selanjutnya.

# Pendahuluan

Setelah menguasai penanganan data dengan Factor dan missing values. Saatnya kita langsung membaca data asli untuk diolah.

Untuk pembacaan data pada course ini, DQLab membagi ke dalam dua bab. Bab ini dikhususkan untuk membaca data dari file teks dan Excel. Sedangkan di bab lain (pada Part 2) akan difokuskan untuk membaca dari database.

Kita akan gunakan satu dataset yang sama untuk bab ini, yaitu dataset Kependudukan Jakarta – ini dengan tujuan agar kita tidak bingung dengan banyaknya dataset.

Namun dataset ini akan datang dengan tiga variasi, yaitu:

- ☐ Teks file dengan format csv (comma separated value) dengan nama `dkikepadatankelurahan2013.csv`.
- ☐ Teks file dengan format tsv (tab separated value) dengan nama `dkikepadatankelurahan2013.tsv`.
- ☐ File Excel dengan format xlsx dengan nama `dkikepadatankelurahan2013.xlsx`.

Fokus untuk bab ini adalah membaca ketiga sumber data saja tanpa pengolahan lebih lanjut – agar materi bisa lebih terfokus.

Beberapa bagian dari bab ini hampir sama dari sisi function yang digunakan pada bab Introduction to R dan Data Visualization with ggplot2, namun lebih detil melihat struktur data hasil dari pembacaan file.

Klik tombol Next untuk melanjutkan.

Next >

# Dataset Kependudukan Jakarta

Dataset yang akan kita gunakan sepanjang praktek course ini adalah data wilayah dan kepadatan kependudukan pemerintah provinsi DKI Jakarta tahun 2013.

DQLab sudah mengambil dan konversi file ini untuk digunakan dalam praktek kita.

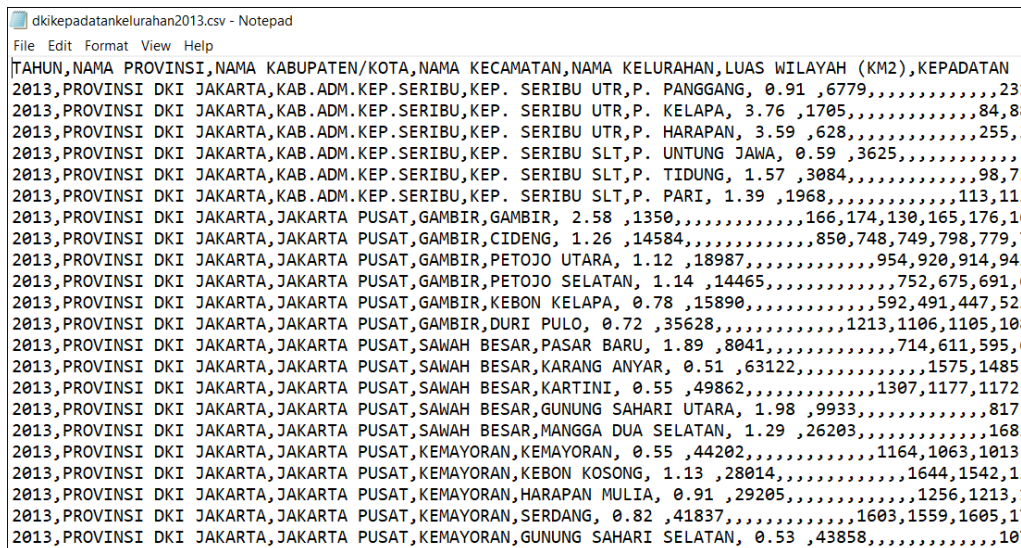
Namun jika diinginkan, sumber dataset ini dapat didownload dari data.go.id dengan url lengkap berikut:

<http://data.go.id/dataset/29fa341d-cfa4-4093-b06b-ab1c246d063a/resource/6b73cc66-7e1b-43da-9421-2770997e5242/download/dkikepadatankelurahan2013.xlsx>

Data kependudukan dipilih karena hampir seluruh organisasi, termasuk bisnis dan institusi pemerintah memerlukan ini untuk mengambil keputusan strategik. Sebagai contoh, dimana saya perlu membuka toko atau kantor cabang? Produk apa yang bakal laku di area tersebut?

Kemudian struktur dari dataset ini perlu diolah lebih lanjut agar optimal. Atas dasar kedua alasan inilah, data kependudukan dipilih untuk course data wrangling with R.

Berikut adalah tampilan data kependudukan DKI Jakarta tersebut jika dibuka di aplikasi Notepad. Perhatikan tanda pemisah antar kolom adalah menggunakan koma.



TAHUN	NAMA PROVINSI	NAMA KABUPATEN/KOTA	NAMA KECAMATAN	NAMA KELURAHAN	LUAS WILAYAH (KM2)	KEPADATAN
2013	PROVINSI DKI JAKARTA	KAB. ADM. KEP. SERIBU	KEP. SERIBU UTR	P. PANGGANG	0.91	6779
2013	PROVINSI DKI JAKARTA	KAB. ADM. KEP. SERIBU	KEP. SERIBU UTR	P. KELAPA	3.76	1705
2013	PROVINSI DKI JAKARTA	KAB. ADM. KEP. SERIBU	KEP. SERIBU UTR	P. HARAPAN	3.59	628
2013	PROVINSI DKI JAKARTA	KAB. ADM. KEP. SERIBU	KEP. SERIBU SLT	P. UNTUNG JAWA	0.59	3625
2013	PROVINSI DKI JAKARTA	KAB. ADM. KEP. SERIBU	KEP. SERIBU SLT	P. TIDUNG	1.57	3084
2013	PROVINSI DKI JAKARTA	KAB. ADM. KEP. SERIBU	KEP. SERIBU SLT	P. PARI	1.39	1968
2013	PROVINSI DKI JAKARTA	JAKARTA	PUSAT, GAMBIR	GAMBIR	2.58	1350
2013	PROVINSI DKI JAKARTA	JAKARTA	PUSAT, GAMBIR	CIDENG	1.26	14584
2013	PROVINSI DKI JAKARTA	JAKARTA	PUSAT, GAMBIR	PETOJO UTARA	1.12	18987
2013	PROVINSI DKI JAKARTA	JAKARTA	PUSAT, GAMBIR	PETOJO SELATAN	1.14	14465
2013	PROVINSI DKI JAKARTA	JAKARTA	PUSAT, GAMBIR	KEBON KELAPA	0.78	15890
2013	PROVINSI DKI JAKARTA	JAKARTA	PUSAT, GAMBIR	DURI PULO	0.72	35628
2013	PROVINSI DKI JAKARTA	JAKARTA	PUSAT, SAWAH BESAR	PASAR BARU	1.89	8041
2013	PROVINSI DKI JAKARTA	JAKARTA	PUSAT, SAWAH BESAR	KARANG ANYAR	0.51	63122
2013	PROVINSI DKI JAKARTA	JAKARTA	PUSAT, SAWAH BESAR	KARTINI	0.55	49862
2013	PROVINSI DKI JAKARTA	JAKARTA	PUSAT, SAWAH BESAR	GUNUNG SAHARI UTARA	1.98	9933
2013	PROVINSI DKI JAKARTA	JAKARTA	PUSAT, SAWAH BESAR	MANGGA DUA SELATAN	1.29	26203
2013	PROVINSI DKI JAKARTA	JAKARTA	PUSAT, KEMAYORAN	KEMAYORAN	0.55	44202
2013	PROVINSI DKI JAKARTA	JAKARTA	PUSAT, KEMAYORAN	KEBON KOSONG	1.13	28014
2013	PROVINSI DKI JAKARTA	JAKARTA	PUSAT, KEMAYORAN	HARAPAN MULIA	0.91	29205
2013	PROVINSI DKI JAKARTA	JAKARTA	PUSAT, KEMAYORAN	SERDANG	0.82	41837
2013	PROVINSI DKI JAKARTA	JAKARTA	PUSAT, KEMAYORAN	GUNUNG SAHARI SELATAN	0.53	43858

Dan ini adalah tampilan jika data tersebut dibuka di aplikasi Excel.



#	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	
1	TAHUN	NAMA PROVINSI	NAMA KABUPATEN/KOTA	NAMA KECAMATAN	NAMA KELURAHAN	LUAS WILAYAH (KM2)	KEPADATAN (JIWA/KM2)													35-39 Laki-Laki	35-
2	2013	PROVINSI DKI JAKARTA	KAB.ADM.KEP.SERIBU	KEP. SERIBU UTR	P. PANGGANG	0.91	6779													231	
3	2013	PROVINSI DKI JAKARTA	KAB.ADM.KEP.SERIBU	KEP. SERIBU UTR	P. KELAPA	3.76	1705													84	
4	2013	PROVINSI DKI JAKARTA	KAB.ADM.KEP.SERIBU	KEP. SERIBU UTR	P. HARAPAN	3.59	628													255	
5	2013	PROVINSI DKI JAKARTA	KAB.ADM.KEP.SERIBU	KEP. SERIBU SLT	P. UNTUNG JAWA	0.59	3625													199	
6	2013	PROVINSI DKI JAKARTA	KAB.ADM.KEP.SERIBU	KEP. SERIBU SLT	P. TIDUNG	1.57	3084													98	
7	2013	PROVINSI DKI JAKARTA	KAB.ADM.KEP.SERIBU	KEP. SERIBU SLT	P. PARI	1.39	1968													113	
8	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	GAMBIR	GAMBIR	2.58	1350													166	
9	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	GAMBIR	CIDENG	1.26	14584													850	
10	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	GAMBIR	PETOJO UTARA	1.12	18987													954	
11	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	GAMBIR	PETOJO SELATAN	1.14	14465													752	
12	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	GAMBIR	KEBON KELAPA	0.78	15890													592	
13	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	GAMBIR	DURI PULO	0.72	35628													1213	
14	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	SAWAH BESAR	PASAR BARU	1.89	8041													714	
15	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	SAWAH BESAR	KARANG ANYAR	0.51	63122													1575	
16	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	SAWAH BESAR	KARTINI	0.55	49862													1307	
17	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	SAWAH BESAR	GUNUNG SAHARI UTARA	1.98	9933													817	
18	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	SAWAH BESAR	MANGGA DUA SELATAN	1.29	26203													1683	
19	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	KEMAYORAN	KEMAYORAN	0.55	44202													1164	
20	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	KEMAYORAN	KEBON KOSONG	1.13	28014													1644	
21	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	KEMAYORAN	HARAPAN MULIA	0.91	29205													1256	
22	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	KEMAYORAN	SERDANG	0.82	41837													1603	
23	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	KEMAYORAN	GUNUNG SAHARI SELATAN	0.53	43858													1071	
24	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	KEMAYORAN	CEMPAKA BARU	0.99	38088													1750	
25	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	KEMAYORAN	SUMUR BATU	1.15	23271													1452	
26	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	KEMAYORAN	UTAN PANJANG	1.05	31889													1610	
27	2013	PROVINSI DKI JAKARTA	JAKARTA PUSAT	SENEN	SENEN	0.81	10158													398	

Terlihat bahwa ada 12 kolom (header dengan huruf H s/d S) yang kosong. Ini akan terbaca sebagai missing value.

Data tersebut memiliki 25 variable kolom dengan penjelasan sebagai berikut:

- ☐ **TAHUN:** Tahun
- ☐ **NAMA PROVINSI:** Nama provinsi di DKI Jakarta, dan nilainya hanya ada satu
- ☐ **NAMA KABUPATEN/KOTA:** Nama kabupaten/kota di DKI Jakarta
- ☐ **NAMA KECAMATAN:** Nama kecamatan di DKI Jakarta
- ☐ **NAMA KELURAHAN:** Nama kelurahan di DKI Jakarta
- ☐ **LUAS WILAYAH (KM2):** Luas wilayah (km persegi)
- ☐ **KEPADATAN (JIWA/KM2):** Kepadatan penduduk (jiwa/km2)
- ☐ **35-39 Laki-Laki:** Jumlah penduduk laki-laki dengan rentang umur 35-39 tahun
- ☐ **35-39 Perempuan:** Jumlah penduduk perempuan dengan rentang umur 35-39 tahun
- ☐ **40-44 Laki-Laki:** Jumlah penduduk laki-laki dengan rentang umur 40-44 tahun
- ☐ **40-44 Perempuan:** Jumlah penduduk perempuan dengan rentang umur 40-44 tahun
- ☐ **45-49 Laki-Laki:** Jumlah penduduk laki-laki dengan rentang umur 45-49 tahun
- ☐ **45-49 Perempuan:** Jumlah penduduk perempuan dengan rentang umur 45-49 tahun
- ☐ **50-54 Laki-Laki:** Jumlah penduduk laki-laki dengan rentang umur 50-54 tahun
- ☐ **50-54 Perempuan:** Jumlah penduduk perempuan dengan rentang umur 50-54 tahun
- ☐ **55-59 Laki-Laki:** Jumlah penduduk laki-laki dengan rentang umur 55-59 tahun
- ☐ **55-59 Perempuan:** Jumlah penduduk perempuan dengan rentang umur 55-59 tahun
- ☐ **60-64 Laki-Laki:** Jumlah penduduk laki-laki dengan rentang umur 60-64 tahun
- ☐ **60-64 Perempuan:** Jumlah penduduk perempuan dengan rentang umur 60-64 tahun
- ☐ **65-69 Laki-Laki:** Jumlah penduduk laki-laki dengan rentang umur 65-69 tahun
- ☐ **65-69 Perempuan:** Jumlah penduduk perempuan dengan rentang umur 65-69 tahun
- ☐ **70-74 Laki-Laki:** Jumlah penduduk laki-laki dengan rentang umur 70-74 tahun
- ☐ **70-74 Perempuan:** Jumlah penduduk perempuan dengan rentang umur 70-74 tahun
- ☐ **>75 Laki-Laki:** Jumlah penduduk laki-laki dengan rentang umur di atas 75 tahun
- ☐ **>75 Perempuan:** Jumlah penduduk perempuan dengan rentang umur di atas 75 tahun

Hampir seluruh kolom berisi nilai angka, kecuali empat kolom berikut: **NAMA PROVINSI, NAMA KABUPATEN/KOTA, NAMA KECAMATAN, dan NAMA KELURAHAN.**

Klik tombol Next untuk melanjutkan ke praktek berikutnya.

# Membaca Dataset CSV

Untuk membaca file versi csv dari dataset kependudukan tersebut kita bisa gunakan function **read.csv** – function ini akan membaca isi dari file teks tersebut menjadi data.frame di R.

File csv yang akan kita baca berlokasi di url berikut:

<https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.csv>

Untuk membaca file tersebut kita gunakan function read.csv dengan perintah lengkap sebagai berikut.

```
penduduk.dki <-  
read.csv("https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.csv",  
sep=",")
```

Komponen	Deskripsi
penduduk.dki	nama variable yang digunakan untuk menampung data dari contoh dataset
read.csv	function yang digunakan untuk membaca contoh dataset yang berupa file
<a href="https://academy.dqlab.id/dataset/customer_segments.txt">https://academy.dqlab.id/dataset/customer_segments.txt</a>	lokasi dataset yang terdapat di web DQLab. Jika lokasi file dan aplikasi R terdapat di komputer lokal Anda, maka gantilah dengan lokasi file di lokal. Misalkan c:\data\customer_segments.txt
sep=", "	Parameter pemisah (separator) antar kolom data. Kita gunakan tanda koma untuk dataset penduduk DKI Jakarta.

Jika terjadi error berikut, cobalah periksa kembali penulisan code – huruf besar, huruf kecil dan juga penulisan lokasi file – dengan teliti.

**Error in file(file, "rt") : cannot open the connection**

Jika tidak terjadi error maka langkah selanjutnya adalah menampilkan isi data dengan mengetikkan nama variable pelanggan pada code editor sebagai berikut.

```
penduduk.dki
```

Hasil eksekusi perintah ini sebagian akan tampak sebagai berikut.

	TAHUN	NAMA. PROVINSI	NAMA. KABUPATEN. KOTA	NAMA. KECAMATAN	NAMA. KELURAHAN	LUAS. WILAYAH. . KM2.	KEPADATAN. . JIWA. KM2.
1	2013	PROVINSI DKI	JAKARTA	KAB. ADM. KEP. SERIBU	KEP. SERIBU UTR	P. PANGGANG	0.91 6779
2	2013	PROVINSI DKI	JAKARTA	KAB. ADM. KEP. SERIBU	KEP. SERIBU UTR	P. KELAPA	3.76 1705
3	2013	PROVINSI DKI	JAKARTA	KAB. ADM. KEP. SERIBU	KEP. SERIBU UTR	P. HARAPAN	3.59 628
4	2013	PROVINSI DKI	JAKARTA	KAB. ADM. KEP. SERIBU	KEP. SERIBU SLT	P. UNTUNG JAWA	0.59 3625
5	2013	PROVINSI DKI	JAKARTA	KAB. ADM. KEP. SERIBU	KEP. SERIBU SLT	P. TIDUNG	1.57 3084
6	2013	PROVINSI DKI	JAKARTA	KAB. ADM. KEP. SERIBU	KEP. SERIBU SLT	P. PARI	1.39 1968
7	2013	PROVINSI DKI	JAKARTA	JAKARTA PUSAT	GAMBIR	GAMBIR	2.58 1350
8	2013	PROVINSI DKI	JAKARTA	JAKARTA PUSAT	GAMBIR	CIDENG	1.26 14584
9	2013	PROVINSI DKI	JAKARTA	JAKARTA PUSAT	GAMBIR	PETOJO UTARA	1.12 18987
10	2013	PROVINSI DKI	JAKARTA	JAKARTA PUSAT	GAMBIR	PETOJO SELATAN	1.14 14465
11	2013	PROVINSI DKI	JAKARTA	JAKARTA PUSAT	GAMBIR	KEBON KELAPA	0.78 15890
12	2013	PROVINSI DKI	JAKARTA	JAKARTA PUSAT	GAMBIR	DURI PULO	0.72 35628
13	2013	PROVINSI DKI	JAKARTA	JAKARTA PUSAT	SAWAH BESAR	PASAR BARU	1.89 8041
14	2013	PROVINSI DKI	JAKARTA	JAKARTA PUSAT	SAWAH BESAR	KARANG ANYAR	0.51 63122
15	2013	PROVINSI DKI	JAKARTA	JAKARTA PUSAT	SAWAH BESAR	KARTINI	0.55 49862
16	2013	PROVINSI DKI	JAKARTA	JAKARTA PUSAT	SAWAH BESAR	GUNUNG SAHARI UTARA	1.98 9933
17	2013	PROVINSI DKI	JAKARTA	JAKARTA PUSAT	SAWAH BESAR	MANGGA DUA SELATAN	1.29 26203

Terlihat isi data dari tujuh kolom pertama dan terdapat nomor baris pada tiap data yang ditampilkan.

### Tugas Praktek

Lengkapi bagian [...1...] pada code editor untuk membaca file seperti yang ditunjukkan pada bagian Lesson.

Code Editor

#Membaca dataset dengan read.csv dan dimasukkan ke variable penduduk.dki

```
penduduk.dki <- read.csv("https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.csv", sep=",")
#[...1...]
```

penduduk.dki

Console

(LANGSUNG PRAKTEK)

# Profile Dataset dengan Function str

Adalah praktek yang sangat baik untuk mengenal atau melakukan *profile* tiap dataset yang sudah dibaca ke dalam R – dan secara sederhana di R dapat kita lakukan dengan perintah str.

Str akan menyajikan informasi tiap kolom dataset dalam format yang *compact* – satu baris informasi saja per row. Pendekatan singkat dan jelas ini membuat str menjadi function favorit dan efektif untuk mengenal data di tahap awal.

Syntaxnya juga cukup sederhana, cukup masukkan dataset ke dalam function ini seperti pada contoh berikut.

```
str(penduduk.dki)
```

## Tugas Praktek

Gantilah bagian [...1...] pada code editor dengan perintah str yang menggunakan input variable penduduk.dki.

Jika berjalan dengan lancar, maka outputnya sebagian akan terlihat sebagai berikut.

```
> str(penduduk.dki)
'data.frame': 267 obs. of 37 variables:
 $ TAHUN : int 2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
 $ NAMA.PROVINSI : Factor w/ 1 level "PROVINSI DKI JAKARTA": 1 1 1 1 1 1 1 1 1 1 ...
 $ NAMA.KABUPATEN.KOTA : Factor w/ 6 levels "JAKARTA BARAT",...: 6 6 6 6 6 6 2 2 2 2 ..
 .
 $ NAMA.KECAMATAN : Factor w/ 44 levels "CAKUNG","CEMPAKA PUTIH",...: 22 22 22 21 21 21
 9 9 9 9 ...
 $ NAMA.KELURAHAN : Factor w/ 267 levels "ANCOL","ANGKE",...: 165 164 163 168 167 166 5
 5 24 195 194 ...
 $ LUAS.WILAYAH..KM2. : num 0.91 3.76 3.59 0.59 1.57 1.39 2.58 1.26 1.12 1.14 ...
 $ KEPADATAN..JIWA.KM2.: int 6779 1705 628 3625 3084 1968 1350 14584 18987 14465 ...
 $ X : logi NA NA NA NA NA NA ...
 $ X.1 : logi NA NA NA NA NA NA ...
 $ X.2 : logi NA NA NA NA NA NA ...
 ...
```

Berikut adalah penjelasan dari beberapa hasil tersebut:

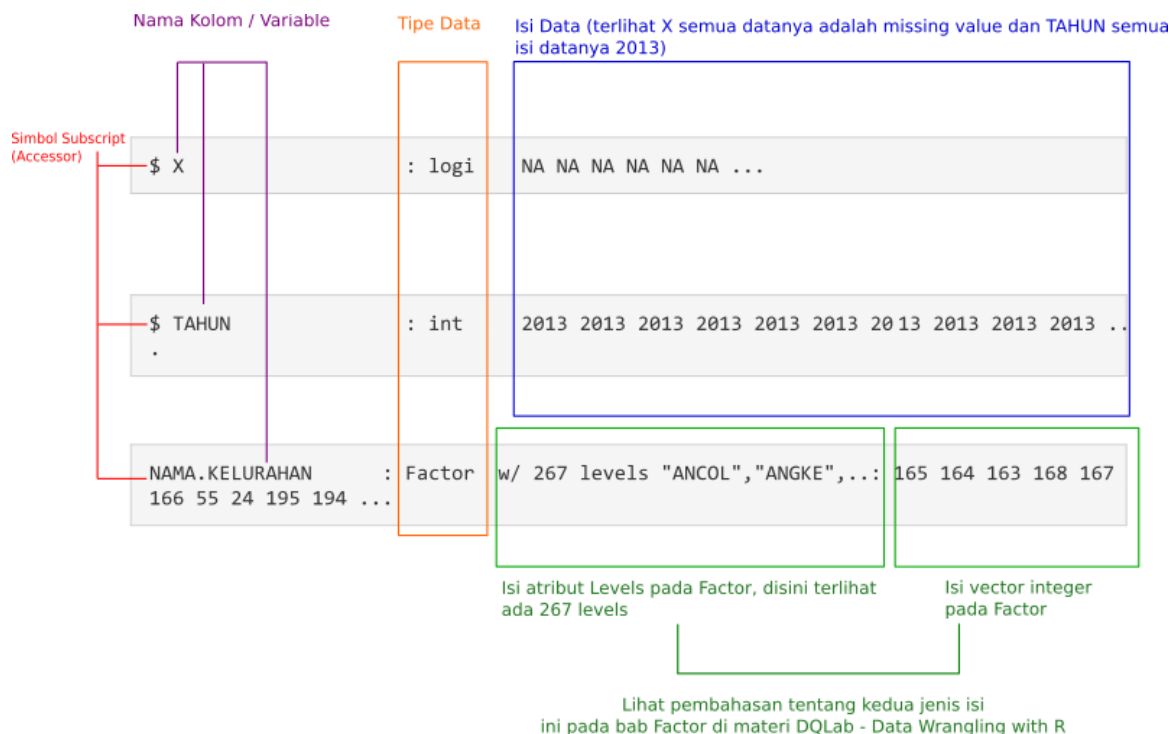
- ☐ 'data.frame': merupakan tipe data dari variable penduduk.dki.
- ☐ 267 obs: menunjukkan adanya 267 total baris data (row) dari dataset ini.

- 37 variables: menunjukkan adanya 37 kolom data pada dataset ini.

Untuk baris di bawahnya adalah penjelasan dari tiap kolom/variable data yang terdiri dari:

- Nama kolom
- Tipe data kolom
- Isi dari kolom tersebut
- Jika Factor maka ada tambahan indexnya

Berikut penjelasan hasil dalam bentuk ilustrasi dari 3 kolom, yaitu TAHUN, NAMA.KELURAHAN, dan X.



## Code Editor

```
#Membaca dataset dengan read.csv dan dimasukkan ke variable penduduk.dki
penduduk.dki <- read.csv("https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.csv", sep=",")
str(penduduk.dki) #[...1...]
```

## Console

```

> #Membaca dataset dengan read.csv dan dimasukkan ke variable penduduk.dki
> penduduk.dki <- read.csv("https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.csv", sep=",")

> str(penduduk.dki) #[...1...]
'data.frame':   267 obs. of  37 variables:
 $ TAHUN          : int  2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
 $ NAMA.PROVINSI   : Factor w/ 1 level "PROVINSI DKI JAKARTA": 1 1 1 1 1 1 1 1 1 1 1 ...
 $ NAMA.KABUPATEN.KOTA : Factor w/ 6 levels "JAKARTA BARAT",...: 6 6 6 6 6 6 2 2 2 2 2 ...
 ..
 $ NAMA.KECAMATAN   : Factor w/ 44 levels "CAKUNG","CEMPAKA PUTIH",...: 22 22 22 21 21 21 9 9 9 9 ...
 $ NAMA.KELURAHAN   : Factor w/ 267 levels "ANCOL","ANGKE",...: 165 164 163 168 167 166 55 24 195 194 ...
 $ LUAS.WILAYAH..KM2. : num  0.91 3.76 3.59 0.59 1.57 1.39 2.58 1.26 1.12 1.14 ...
 $ KEPADATAN..JIWA.KM2.: int  6779 1705 628 3625 3084 1968 1350 14584 18987 14465 ...
 $ X                : logi  NA NA NA NA NA NA NA ...
 $ X.1              : logi  NA NA NA NA NA NA NA ...
 $ X.2              : logi  NA NA NA NA NA NA NA ...
 $ X.3              : logi  NA NA NA NA NA NA NA ...
 $ X.4              : logi  NA NA NA NA NA NA NA ...
 $ X.5              : logi  NA NA NA NA NA NA NA ...
 $ X.6              : logi  NA NA NA NA NA NA NA ...
 $ X.7              : logi  NA NA NA NA NA NA NA ...
 $ X.8              : logi  NA NA NA NA NA NA NA ...
 $ X.9              : logi  NA NA NA NA NA NA NA ...
 $ X.10             : logi  NA NA NA NA NA NA NA ...
 $ X.11             : logi  NA NA NA NA NA NA NA ...
 $ X35.39.Laki.Laki : int  231 84 255 199 98 113 166 850 954 752 ...
 $ X35.39.Perempuan : int  235 88 238 185 75 112 174 748 920 675 ...
 $ X40.44.Laki.Laki : int  233 99 232 178 73 108 130 749 914 691 ...
 $ X40.44.Perempuan : int  210 88 234 176 94 80 165 798 943 691 ...
 $ X45.49.Laki.Laki : int  171 72 212 162 67 66 176 779 871 659 ...
 $ X45.49.Perempuan : int  158 63 193 139 69 62 162 766 823 631 ...
 $ X50.54.Laki.Laki : int  137 34 150 100 60 61 129 715 736 611 ...
 $ X50.54.Perempuan : int  126 29 161 119 40 63 97 662 679 514 ...
 $ X55.59.Laki.Laki : int  98 30 139 97 37 37 108 614 680 539 ...
 $ X55.59.Perempuan : int  106 39 101 83 32 36 90 537 510 466 ...
 $ X60.64.Laki.Laki : int  72 29 73 58 22 32 88 555 544 428 ...
 $ X60.64.Perempuan : int  65 24 56 56 13 26 42 343 421 279 ...
 $ X65.69.Laki.Laki : int  36 12 18 40 18 21 68 413 398 328 ...
 $ X65.69.Perempuan : int  33 21 35 54 15 14 34 215 235 160 ...
 $ X70.74.Laki.Laki : int  33 13 24 26 10 17 37 259 241 215 ...
 $ X70.74.Perempuan : int  20 5 25 27 18 11 32 142 132 116 ...
 $ X.75.Laki.Laki   : int  13 5 18 16 11 8 34 214 215 150 ...
 $ X.75..Perempuan  : int  27 8 26 13 17 7 23 165 159 121 ...

```

# Profile Dataset dengan Function summary

Selain perintah `str`, kita dapat juga menggunakan function `summary` untuk melihat kondisi dataset kita dalam bentuk ringkasan yang lebih detil.

Sebagai contoh, untuk melihat summary dari variable `penduduk.dki` maka kita gunakan konstruksi berikut.

```
summary(penduduk.dki)
```

## Tugas Praktek

Gantilah bagian [...1...] pada code editor dengan perintah `summary` yang menggunakan input variable `penduduk.dki`.

Jika berjalan dengan lancar, maka outputnya akan terlihat hasilnya sebagai berikut.

Untuk angka, maka akan muncul statistik minimum, maximum, mean, meadian, 1<sup>st</sup> quantile, dan 3<sup>rd</sup> quantile. Sedangkan untuk nilai lain akan muncul jumlahnya di dataset.

TAHUN	NAMA.PROVINSI	NAMA.KABUPATEN.KOTA
Min. :2013	PROVINSI DKI JAKARTA:267	JAKARTA BARAT :56
1st Qu.:2013		JAKARTA PUSAT :44
Median :2013		JAKARTA SELATAN :65
Mean :2013		JAKARTA TIMUR :65
3rd Qu.:2013		JAKARTA UTARA :31
Max. :2013		KAB.ADM.KEP.SERIBU: 6

NAMA.KECAMATAN	NAMA.KELURAHAN	LUAS.WILAYAH..KM2.
TAMBORA : 11	ANCOL : 1	Min. : 0.270
KEBAYORAN BARU: 10	ANGKE : 1	1st Qu.: 0.965
CIPAYUNG : 8	BALE KAMBANG: 1	Median : 1.800
JATINEGARA : 8	BALI MESTER : 1	Mean : 2.487
KEMAYORAN : 8	BAMBU APUS : 1	3rd Qu.: 3.315
SETIA BUDI : 8	BANGKA : 1	Max. :13.070
(Other) :214	(Other) :261	
KEPADATAN..JIWA.KM2. X	X.1	X.2
Min. : 628	Mode:logical	Mode:logical

```
1st Qu.:11734      NA's:267      NA's:267      NA's:267
Median :17304
Mean   :21974
3rd Qu.:29226
Max.   :94166
```

```
      X.3      X.4      X.5      X.6      X.7
Mode:logical Mode:logical Mode:logical Mode:logical Mode:logical
NA's:267      NA's:267      NA's:267      NA's:267      NA's:267
```

```
      X.8      X.9      X.10     X.11     X35.39.Laki.Laki
Mode:logical Mode:logical Mode:logical Mode:logical Min.   :   84
NA's:267      NA's:267      NA's:267      NA's:267      1st Qu.: 1186
                                           Median : 1880
                                           Mean   : 2264
                                           3rd Qu.: 2768
                                           Max.   :13011
```

```
X35.39.Perempuan X40.44.Laki.Laki X40.44.Perempuan X45.49.Laki.Laki
Min.   :   75      Min.   :   73      Min.   :   80      Min.   :  66.0
1st Qu.:1062      1st Qu.:1023      1st Qu.: 1084      1st Qu.:  957.5
Median :1631      Median :1576      Median : 1714      Median :1404.0
Mean   :1741      Mean   :1676      Mean   : 2332      Mean   :1643.7
3rd Qu.:2213      3rd Qu.:2112      3rd Qu.: 2782      3rd Qu.:1949.0
Max.   :7488      Max.   :7243      Max.   :14731      Max.   :8822.0
```

```
X45.49.Perempuan X50.54.Laki.Laki X50.54.Perempuan X55.59.Laki.Laki
Min.   :   62      Min.   :   34      Min.   :  29.0      Min.   :  30.0
1st Qu.:  886      1st Qu.:  790      1st Qu.: 712.5      1st Qu.:  595.5
Median :1315      Median : 1216      Median :1107.0      Median :  909.0
```



Mean :1577	Mean : 2259	Mean :1336.1	Mean :1178.6
3rd Qu.:1867	3rd Qu.: 2624	3rd Qu.:1671.5	3rd Qu.:1405.5
Max. :8352	Max. :17174	Max. :7480.0	Max. :6846.0

X55.59.Perempuan	X60.64.Laki.Laki	X60.64.Perempuan	X65.69.Laki.Laki
Min. : 32	Min. : 22.0	Min. : 13.0	Min. : 12
1st Qu.: 557	1st Qu.: 419.5	1st Qu.: 366.0	1st Qu.: 253
Median : 889	Median : 650.0	Median : 587.0	Median : 413
Mean : 1867	Mean : 981.7	Mean : 876.7	Mean : 1403
3rd Qu.: 2342	3rd Qu.:1186.5	3rd Qu.:1052.5	3rd Qu.: 2098
Max. :14326	Max. :6333.0	Max. :5476.0	Max. :11809

X65.69.Perempuan	X70.74.Laki.Laki	X70.74.Perempuan	X.75.Laki.Laki
Min. : 14.0	Min. : 10.0	Min. : 5	Min. : 5.0
1st Qu.: 215.5	1st Qu.: 170.0	1st Qu.: 145	1st Qu.: 116.5
Median : 354.0	Median : 285.0	Median : 260	Median : 200.0
Mean : 683.6	Mean : 607.0	Mean :1083	Mean : 484.6
3rd Qu.: 928.5	3rd Qu.: 836.5	3rd Qu.:1784	3rd Qu.: 716.0
Max. :4758.0	Max. :4475.0	Max. :9233	Max. :3959.0

X.75..Perempuan

Min. : 7.0

1st Qu.: 121.5

Median : 204.0

Mean : 480.3

3rd Qu.: 675.5

Max. :3526.0

## Code Editor

#Membaca dataset dengan read.csv dan dimasukkan ke variable penduduk.dki

```
penduduk.dki <- read.csv("https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.csv", sep=",")
```

```
summary(penduduk.dki) #[...1...]
```

## Console

```
> #Membaca dataset dengan read.csv dan dimasukkan ke variable penduduk.dki
> penduduk.dki <- read.csv("https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.csv", sep=",")

> summary(penduduk.dki) #[...1...]
      TAHUN          NAMA.PROVINSI          NAMA.KABUPATEN.KOTA
Min.   :2013   PROVINSI DKI JAKARTA:267   JAKARTA BARAT      :56
1st Qu.:2013                                JAKARTA PUSAT       :44
Median :2013                                JAKARTA SELATAN    :65
Mean   :2013                                JAKARTA TIMUR      :65
3rd Qu.:2013                                JAKARTA UTARA      :31
Max.   :2013                                KAB.ADM.KEP.SERIBU: 6

      NAMA.KECAMATAN      NAMA.KELURAHAN LUAS.WILAYAH..KM2.
TAMBORA      : 11   ANCOL      : 1   Min.   : 0.270
KEBAYORAN BARU: 10   ANGKE     : 1   1st Qu.: 0.965
CIPAYUNG     : 8    BALE KAMBANG: 1   Median : 1.800
JATINEGARA   : 8    BALI MESTER : 1   Mean   : 2.487
KEMAYORAN    : 8    BAMBU APUS  : 1   3rd Qu.: 3.315
SETIA BUDI   : 8    BANGKA     : 1   Max.   :13.070
(Other)      :214   (Other)    :261

KEPADATAN..JIWA.KM2.   X      X.1      X.2
Min.   : 628           Mode:logical Mode:logical Mode:logical
1st Qu.:11734          NA's:267    NA's:267    NA's:267
Median :17304
Mean   :21974
3rd Qu.:29226
Max.   :94166

      X.3      X.4      X.5      X.6      X.7
Mode:logical Mode:logical Mode:logical Mode:logical Mode:logical
NA's:267     NA's:267 NA's:267 NA's:267 NA's:267

      X.8      X.9      X.10     X.11     X35.39.Laki.Laki
Mode:logical Mode:logical Mode:logical Mode:logical Min.   : 84
NA's:267     NA's:267 NA's:267 NA's:267 1st Qu.: 1186
Median : 1880
Mean   : 2264
3rd Qu.: 2768
```

Max. :13011

X35.39.Perempuan	X40.44.Laki.Laki	X40.44.Perempuan	X45.49.Laki.Laki
Min. : 75	Min. : 73	Min. : 80	Min. : 66.0
1st Qu.:1062	1st Qu.:1023	1st Qu.: 1084	1st Qu.: 957.5
Median :1631	Median :1576	Median : 1714	Median :1404.0
Mean :1741	Mean :1676	Mean : 2332	Mean :1643.7
3rd Qu.:2213	3rd Qu.:2112	3rd Qu.: 2782	3rd Qu.:1949.0
Max. :7488	Max. :7243	Max. :14731	Max. :8822.0

X45.49.Perempuan	X50.54.Laki.Laki	X50.54.Perempuan	X55.59.Laki.Laki
Min. : 62	Min. : 34	Min. : 29.0	Min. : 30.0
1st Qu.: 886	1st Qu.: 790	1st Qu.: 712.5	1st Qu.: 595.5
Median :1315	Median : 1216	Median :1107.0	Median : 909.0
Mean :1577	Mean : 2259	Mean :1336.1	Mean :1178.6
3rd Qu.:1867	3rd Qu.: 2624	3rd Qu.:1671.5	3rd Qu.:1405.5
Max. :8352	Max. :17174	Max. :7480.0	Max. :6846.0

X55.59.Perempuan	X60.64.Laki.Laki	X60.64.Perempuan	X65.69.Laki.Laki
Min. : 32	Min. : 22.0	Min. : 13.0	Min. : 12
1st Qu.: 557	1st Qu.: 419.5	1st Qu.: 366.0	1st Qu.: 253
Median : 889	Median : 650.0	Median : 587.0	Median : 413
Mean : 1867	Mean : 981.7	Mean : 876.7	Mean : 1403
3rd Qu.: 2342	3rd Qu.:1186.5	3rd Qu.:1052.5	3rd Qu.: 2098
Max. :14326	Max. :6333.0	Max. :5476.0	Max. :11809

X65.69.Perempuan	X70.74.Laki.Laki	X70.74.Perempuan	X.75.Laki.Laki
Min. : 14.0	Min. : 10.0	Min. : 5	Min. : 5.0
1st Qu.: 215.5	1st Qu.: 170.0	1st Qu.: 145	1st Qu.: 116.5
Median : 354.0	Median : 285.0	Median : 260	Median : 200.0
Mean : 683.6	Mean : 607.0	Mean :1083	Mean : 484.6
3rd Qu.: 928.5	3rd Qu.: 836.5	3rd Qu.:1784	3rd Qu.: 716.0
Max. :4758.0	Max. :4475.0	Max. :9233	Max. :3959.0

X.75..Perempuan

Min. : 7.0

1st Qu.: 121.5

Median : 204.0

Mean : 480.3

3rd Qu.: 675.5

Max. :3526.0

# Menggunakan argumen `check.names = FALSE`

Jika kita perhatikan pada eksekusi `read.csv` dan hasilnya, terlihat ada kolom dengan prefix X – yaitu X, X.1, X.2, dan seterusnya. Ini terjadi karena `read.csv` mendeteksi ada nama kolom yang kosong dan lebih dari. Kondisi ini akan secara otomatis "diperbaiki" oleh function `read.csv` dengan menambahkan prefix X di depan kolom.

Jika kita tidak menginginkan hal tersebut, kita bisa tambahkan argumen `check.names = FALSE` pada statement `read.csv` sehingga konstruksinya menjadi:

```
read.csv(..., ..., check.names = FALSE)
```

Cobalah lakukan tugas berikut untuk lebih jelasnya.

## Tugas Praktek

Gantilah bagian [...1...] dengan argumen `check.names = FALSE`.

Jika berjalan dengan lancar, maka outputnya akan terlihat sebagai berikut.

```
'data.frame': 267 obs. of 37 variables:
 $ TAHUN          : int  2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
 $ NAMA PROVINSI  : Factor w/ 1 level "PROVINSI DKI JAKARTA": 1 1 1 1 1 1 1 1 1 1
1 ...
 $ NAMA KABUPATEN/KOTA : Factor w/ 6 levels "JAKARTA BARAT",...: 6 6 6 6 6 6 2 2 2 2 .
..
 $ NAMA KECAMATAN    : Factor w/ 44 levels "CAKUNG","CEMPAKA PUTIH",...: 22 22 22 21
21 21 9 9 9 9 ...
 $ NAMA KELURAHAN    : Factor w/ 267 levels "ANCOL","ANGKE",...: 165 164 163 168 167
166 55 24 195 194 ...
 $ LUAS WILAYAH (KM2) : num  0.91 3.76 3.59 0.59 1.57 1.39 2.58 1.26 1.12 1.14 ...
 $ KEPADATAN (JIWA/KM2): int  6779 1705 628 3625 3084 1968 1350 14584 18987 14465 ...
 $                  : logi  NA NA NA NA NA NA ...
 $                  : logi  NA NA NA NA NA NA ...
 $                  : logi  NA NA NA NA NA NA ...
```

Beberapa penjelasan terkait perbedaan hasil di atas dengan hasil pada praktek sebelumnya:

- ☐ Nama spasi tidak dijadikan titik.
- ☐ Nama yang kosong tidak diberi tanda X, ini akan menyulitkan ketika kita ingin mereferensikan nama kolom kosong tersebut.

## Code Editor

```
#Membaca dataset dengan read.csv dan dimasukkan ke variable penduduk.dki

penduduk.dki <- read.csv("https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.csv", sep=";",
check.names = FALSE) #[...1...]

str(penduduk.dki)
```

## Console

```
> #Membaca dataset dengan read.csv dan dimasukkan ke variable penduduk.dki
> penduduk.dki <- read.csv("https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.csv", sep=";", check.names = FALSE) #[...1...]

> str(penduduk.dki)
'data.frame': 267 obs. of 37 variables:
 $ TAHUN : int 2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
 $ NAMA PROVINSI : Factor w/ 1 level "PROVINSI DKI JAKARTA": 1 1 1 1 1 1 1 1 1 1 ...
 $ NAMA KABUPATEN/KOTA : Factor w/ 6 levels "JAKARTA BARAT",...: 6 6 6 6 6 6 2 2 2 2 ...
 $ NAMA KECAMATAN : Factor w/ 44 levels "CAKUNG","CEMPAKA PUTIH",...: 22 22 22 21 21 21 9 9 9 9 ...
 $ NAMA KELURAHAN : Factor w/ 267 levels "ANCOL","ANGKE",...: 165 164 163 168 167 166 55 24 195 194 ...
 $ LUAS WILAYAH (KM2) : num 0.91 3.76 3.59 0.59 1.57 1.39 2.58 1.26 1.12 1.14 ...
 $ KEPADATAN (JIWA/KM2): int 6779 1705 628 3625 3084 1968 1350 14584 18987 14465 ...
 $ : logi NA NA NA NA NA NA ...
 $ : logi NA NA NA NA NA NA ...
 $ : logi NA NA NA NA NA NA ...
 $ : logi NA NA NA NA NA NA ...
 $ : logi NA NA NA NA NA NA ...
 $ : logi NA NA NA NA NA NA ...
 $ : logi NA NA NA NA NA NA ...
 $ : logi NA NA NA NA NA NA ...
 $ : logi NA NA NA NA NA NA ...
 $ : logi NA NA NA NA NA NA ...
 $ : logi NA NA NA NA NA NA ...
 $ : logi NA NA NA NA NA NA ...
 $ : logi NA NA NA NA NA NA ...
 $ : logi NA NA NA NA NA NA ...
 $ : logi NA NA NA NA NA NA ...
 $ 35-39 Laki-Laki : int 231 84 255 199 98 113 166 850 954 752 ...
 $ 35-39 Perempuan : int 235 88 238 185 75 112 174 748 920 675 ...
 $ 40-44 Laki-Laki : int 233 99 232 178 73 108 130 749 914 691 ...
 $ 40-44 Perempuan : int 210 88 234 176 94 80 165 798 943 691 ...
 $ 45-49 Laki-Laki : int 171 72 212 162 67 66 176 779 871 659 ...
 $ 45-49 Perempuan : int 158 63 193 139 69 62 162 766 823 631 ...
 $ 50-54 Laki-Laki : int 137 34 150 100 60 61 129 715 736 611 ...
 $ 50-54 Perempuan : int 126 29 161 119 40 63 97 662 679 514 ...
 $ 55-59 Laki-Laki : int 98 30 139 97 37 37 108 614 680 539 ...
 $ 55-59 Perempuan : int 106 39 101 83 32 36 90 537 510 466 ...
```

```
$ 60-64 Laki-Laki      : int  72 29 73 58 22 32 88 555 544 428 ...
$ 60-64 Perempuan     : int  65 24 56 56 13 26 42 343 421 279 ...
$ 65-69 Laki-Laki     : int  36 12 18 40 18 21 68 413 398 328 ...
$ 65-69 Perempuan     : int  33 21 35 54 15 14 34 215 235 160 ...
$ 70-74 Laki-Laki     : int  33 13 24 26 10 17 37 259 241 215 ...
$ 70-74 Perempuan     : int  20 5 25 27 18 11 32 142 132 116 ...
$ >75 Laki-Laki       : int  13 5 18 16 11 8 34 214 215 150 ...
$ >75 Perempuan       : int  27 8 26 13 17 7 23 165 159 121 ...
```

# Membaca Tab Separated Value (TSV)

Untuk membaca file versi tsv dimana pemisah antar field adalah karakter tabulasi (tab) dari dataset kependudukan tersebut kita tetap gunakan function **read.csv**.

Perbedaannya hanyalah di argumen separator dimana sebelumnya adalah tanda koma (,), maka untuk tsv perlu diganti menjadi backslash t (\t).

```
sep="\t"
```

Dataset file untuk contoh tsv ini dapat Anda download di url berikut:

<https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.tsv>

Untuk membaca file tersebut dengan perintah read.csv adalah sebagai berikut.

```
penduduk.dki <-  
read.csv("https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.tsv",  
sep="\t")
```

## Tugas Praktek

Gantilah bagian [...1...] pada code editor untuk membaca file tsv seperti yang ditunjukkan pada contoh di Lesson, dan masukkan hasilnya pada variable penduduk.dki.

Jika berjalan lancar maka hasil outputnya akan terlihat seperti pada saat kita membaca file csv.

### Code Editor

```
#Membaca dataset dengan read.csv dan dimasukkan ke variable penduduk.dki  
  
penduduk.dki <- read.csv("https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.tsv",  
sep="\t") #[...1...]  
  
penduduk.dki
```

### Console

(LANGSUNG PRAKTEK)

# Membaca Dataset File Excel dengan read.xlsx

Jika membaca file teks berformat csv dan tsv hampir sama, maka untuk membaca file versi Excel dari dataset kependudukan tersebut kita perlu gunakan function **read.xlsx** dari library lain yang bernama openxlsx.

Function ini akan membaca isi dari file Excel menjadi data.frame di R.

Untuk praktek kita kali ini, file Excel yang akan kita baca berlokasi di url berikut:

<https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.xlsx>

Berikut adalah contoh konstruksi function read.xlsx untuk membaca file Excel tersebut.

```
penduduk.dki.xlsx <-
read.xlsx(xlsxFile="https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.xlsx")
```

## Tugas Praktek

Gantilah bagian [...1...] pada code editor untuk membaca file Excel seperti yang ditunjukkan pada contoh di Lesson, dan masukkan hasilnya pada variable penduduk.dki.xlsx. Pada baris terakhir code editor juga sudah ditambahkan function str sehingga kita bisa menganalisa struktur dan isi dari hasil pembacaan read.xlsx.

Jika berjalan lancar maka hasil outputnya akan terlihat sebagai berikut.

```
> str(penduduk.dki.xlsx)
'data.frame':   267 obs. of  25 variables:
 $ TAHUN                : num  2013 2013 2013 2013 2013 ...
 $ NAMA.PROVINSI        : chr  "PROVINSI DKI JAKARTA" "PROVINSI DKI JAKARTA" "PROVINSI DKI JAKARTA" "PROVINSI DKI JAKARTA" ...
 $ NAMA.KABUPATEN/KOTA : chr  "KAB.ADM.KEP.SERIBU" "KAB.ADM.KEP.SERIBU" "KAB.ADM.KEP.SERIBU" "KAB.ADM.KEP.SERIBU" ...
 $ NAMA.KECAMATAN       : chr  "KEP. SERIBU UTR" "KEP. SERIBU UTR" "KEP. SERIBU UTR" "KEP. SERIBU SLT" ...
 $ NAMA.KELURAHAN       : chr  "P. PANGGANG" "P. KELAPA" "P. HARAPAN" "P. UNTUNG JAWA" ...
 $ LUAS.WILAYAH.(KM2)   : num  0.91 3.76 3.59 0.59 1.57 1.39 2.58 1.26 1.12 1.14 ...
 $ KEPADATAN.(JIWA/KM2): num  6779 1705 628 3625 3084 ...
 $ 35-39.Laki-Laki      : num  231 84 255 199 98 113 166 850 954 752 ...
 $ 35-39.Perempuan      : num  235 88 238 185 75 112 174 748 920 675 ...
 $ 40-44.Laki-Laki      : num  233 99 232 178 73 108 130 749 914 691 ...
```



Terlihat perbedaan penting ketika kita membaca dengan `read.xlsx` dengan `read.csv` dan `read.tsv` secara *default*.

Beberapa diantaranya adalah sebagai berikut:

- ❑ Karakter non karakter dan spasi seperti garis miring dan tanda kurung tetap ditampilkan apa adanya. Sedangkan jika menggunakan `read.csv` dan `read.tsv` akan diganti menjadi tanda titik.
- ❑ Seluruh kolom akan diubah menjadi vector, sedangkan pada `read.csv` beberapa kolom akan menjadi factor.
- ❑ Vector teks yang berulang akan tetap dibiarkan apa adanya, tidak diubah menjadi factor. Ini akan mengakibatkan beberapa masalah, salah satunya adalah pada saat plotting data – yang akan kita lihat pada praktek berikutnya.
- ❑ Kolom kosong – dimana jika kita gunakan `read.csv` akan diisi dengan nama X, X.1, X.2, dan seterusnya – dihilangkan oleh fungsi `read.xlsx` ini.

Code Editor

```
library(openxlsx)
```

```
#Membaca dataset dengan read.xlsx dan dimasukkan ke variable penduduk.dki
```

```
penduduk.dki.xlsx <-  
read.xlsx(xlsxFile="https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.xlsx") #[...1...]
```

```
str(penduduk.dki.xlsx)
```

Console

```
> library(openxlsx)

> #Membaca dataset dengan read.xlsx dan dimasukkan ke variable penduduk.dki
> penduduk.dki.xlsx <- read.xlsx(xlsxFile="https://academy.dqlab.id/dataset/dkikepada
tanelurahan2013.xlsx") #[...1...]

> str(penduduk.dki.xlsx)
'data.frame':   267 obs. of  25 variables:
 $ TAHUN          : num  2013 2013 2013 2013 2013 ...
 $ NAMA.PROVINSI   : chr  "PROVINSI DKI JAKARTA" "PROVINSI DKI JAKARTA" "PROVINSI
DKI JAKARTA" "PROVINSI DKI JAKARTA" ...
 $ NAMA.KABUPATEN/KOTA : chr  "KAB.ADM.KEP.SERIBU" "KAB.ADM.KEP.SERIBU" "KAB.ADM.KEP.
SERIBU" "KAB.ADM.KEP.SERIBU" ...
 $ NAMA.KECAMATAN   : chr  "KEP. SERIBU UTR" "KEP. SERIBU UTR" "KEP. SERIBU UTR" "
KEP. SERIBU SLT" ...
 $ NAMA.KELURAHAN   : chr  "P. PANGGANG" "P. KELAPA" "P. HARAPAN" "P. UNTUNG JAWA"
...
 $ LUAS.WILAYAH.(KM2) : num  0.91 3.76 3.59 0.59 1.57 1.39 2.58 1.26 1.12 1.14 ...
 $ KEPADATAN.(JIWA/KM2): num  6779 1705 628 3625 3084 ...
 $ 35-39.Laki-Laki    : num  231 84 255 199 98 113 166 850 954 752 ...
 $ 35-39.Perempuan    : num  235 88 238 185 75 112 174 748 920 675 ...
```

```
$ 40-44.Laki-Laki      : num  233 99 232 178 73 108 130 749 914 691 ...
$ 40-44.Perempuan     : num  210 88 234 176 94 80 165 798 943 691 ...
$ 45-49.Laki-Laki     : num  171 72 212 162 67 66 176 779 871 659 ...
$ 45-49.Perempuan     : num  158 63 193 139 69 62 162 766 823 631 ...
$ 50-54.Laki-Laki     : num  137 34 150 100 60 61 129 715 736 611 ...
$ 50-54.Perempuan     : num  126 29 161 119 40 63 97 662 679 514 ...
$ 55-59.Laki-Laki     : num   98 30 139 97 37 37 108 614 680 539 ...
$ 55-59.Perempuan     : num  106 39 101 83 32 36 90 537 510 466 ...
$ 60-64.Laki-Laki     : num   72 29 73 58 22 32 88 555 544 428 ...
$ 60-64.Perempuan     : num   65 24 56 56 13 26 42 343 421 279 ...
$ 65-69.Laki-Laki     : num   36 12 18 40 18 21 68 413 398 328 ...
$ 65-69.Perempuan     : num   33 21 35 54 15 14 34 215 235 160 ...
$ 70-74.Laki-Laki     : num   33 13 24 26 10 17 37 259 241 215 ...
$ 70-74.Perempuan     : num   20 5 25 27 18 11 32 142 132 116 ...
$ >75.Laki-Laki       : num   13 5 18 16 11 8 34 214 215 150 ...
$ >75.Perempuan       : num   27 8 26 13 17 7 23 165 159 121 ...
```

# Kesimpulan

Anda telah menyelesaikan bab tentang membaca sumber data dengan tiga format, yaitu:

- ☐ File teks berformat comma separated value (csv)
- ☐ File teks berformat tab separated value (tsv)
- ☐ File Excel berformat xlsx

Walaupun terlihat sederhana dan pendek dibanding bab lain, namun beberapa praktek pada bab ini memiliki metodologi dan fungsi penting yakni:

- ☐ Bagaimana kita melakukan profil dengan function str dari tiap kali pembacaan file.
- ☐ Dapat mengerti output yang dihasilkan oleh function str.
- ☐ Dengan demikian, kita menjadi aware atau lebih perhatian karena perilaku yang berbeda ketika menangani kolom kosong dan juga pada saat penamaan variable kolom.

Dengan menguasai praktek-praktek ini, Anda akan lebih siap untuk "memperbaiki" struktur dan isi file tersebut jika diperlukan.

Klik tombol Next untuk melanjutkan.

# Pendahuluan

Pada bab sebelumnya, terlihat file teks dan Excel yang dibaca memiliki masalahnya sendiri-sendiri. Sebagai contoh, `read.csv` mengakomodir kolom-kolom kosong sehingga harus kita buang. Sedangkan `read.xlsx` tidak menghasilkan Factor sehingga kita perlu waktu dan tenaga tambahan untuk mengidentifikasi dan mengolah kolom yang bersifat kategorik tersebut dengan melakukan konversi kolom tersebut menjadi Factor.

Selain itu terlihat juga ada beberapa kolom yang harusnya bisa dijumlahkan menjadi satu kolom – yaitu kolom 35-39.Perempuan, 40-44.Perempuan, dan seterusnya – menjadi kolom jumlah. Kemudian informasi umur di nama kolom tersebut sebenarnya dapat kita pisahkan menjadi kolom tersendiri.

Perkenalan function dan latihan untuk transformasi struktur seperti ini akan dilakukan sepanjang bab ini.

Klik tombol Next untuk melanjutkan.

# Function names

Pada bab sebelumnya, kita menggunakan function `str` untuk melihat ringkasan dari struktur nama, tipe data, dan isi dari `data.frame` hasil pembacaan file.

Namun jika kita hanya ingin melihat nama-nama kolom saja, bisa menggunakan function bernama **names**, dengan konstruksi berikut.

```
names(variable)
```

## Tugas Praktek

Gantilah bagian `[...1...]` dengan function `names` dengan input variable `variable` `data.frame` hasil pembacaan file `csv`.

Jika berjalan dengan lancar, Anda akan mendapatkan hasil sebagai berikut.

```
[1] "TAHUN" "NAMA.PROVINSI" "NAMA.KABUPATEN.KOTA"
[4] "NAMA.KECAMATAN" "NAMA.KELURAHAN" "LUAS.WILAYAH..KM2."
[7] "KEPADATAN..JIWA.KM2." "X" "X.1"
[10] "X.2" "X.3" "X.4"
[13] "X.5" "X.6" "X.7"
[16] "X.8" "X.9" "X.10"
[19] "X.11" "X35.39.Laki.Laki" "X35.39.Perempuan"
[22] "X40.44.Laki.Laki" "X40.44.Perempuan" "X45.49.Laki.Laki"
[25] "X45.49.Perempuan" "X50.54.Laki.Laki" "X50.54.Perempuan"
[28] "X55.59.Laki.Laki" "X55.59.Perempuan" "X60.64.Laki.Laki"
[31] "X60.64.Perempuan" "X65.69.Laki.Laki" "X65.69.Perempuan"
[34] "X70.74.Laki.Laki" "X70.74.Perempuan" "X.75.Laki.Laki"
[37] "X.75..Perempuan"
```

## Code Editor

```
#Membaca dataset csv
```

```
penduduk.dki.csv <-
read.csv("https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.csv", sep=",")
```

```
#Menggunakan names untuk variable penduduk.dki.csv
```

```
names(penduduk.dki.csv) #[...1...]
```

## Console

```

> #Membaca dataset csv
> penduduk.dki.csv <- read.csv("https://academy.dqlab.id/dataset/dkikepadatankelurahan
2013.csv", sep=",")

> #Menggunakan names untuk variable penduduk.dki.csv
> names(penduduk.dki.csv) #[...1...]
[1] "TAHUN" "NAMA.PROVINSI" "NAMA.KABUPATEN.KOTA"
[4] "NAMA.KECAMATAN" "NAMA.KELURAHAN" "LUAS.WILAYAH..KM2."
[7] "KEPADATAN..JIWA.KM2." "X" "X.1"
[10] "X.2" "X.3" "X.4"
[13] "X.5" "X.6" "X.7"
[16] "X.8" "X.9" "X.10"
[19] "X.11" "X35.39.Laki.Laki" "X35.39.Perempuan"
[22] "X40.44.Laki.Laki" "X40.44.Perempuan" "X45.49.Laki.Laki"
[25] "X45.49.Perempuan" "X50.54.Laki.Laki" "X50.54.Perempuan"
[28] "X55.59.Laki.Laki" "X55.59.Perempuan" "X60.64.Laki.Laki"
[31] "X60.64.Perempuan" "X65.69.Laki.Laki" "X65.69.Perempuan"
[34] "X70.74.Laki.Laki" "X70.74.Perempuan" "X.75.Laki.Laki"
[37] "X.75..Perempuan"

```

# Merubah Satu Nama Kolom

Function `names` ini juga bisa digunakan merubah nama kolom pada `data.frame`. Untuk merubahnya kita gunakan konstruksi berikut:

```
names(variable)[posisi] <- "nama_baru"
```

Dengan posisi adalah nomor indeks dari posisi nama kolom yang ingin diubah. Berikut adalah contoh dimana kita merubah nama kolom dari `TAHUN` ke `PERIODE` (posisi 1) dari dataset kita.

```
names(penduduk.dki.csv)[1] <- "PERIODE"
```

## Tugas Praktek

Gantilah bagian `[...1...]` dan `[...2...]` dengan perintah yang merubah nama kolom dari dataset `kependudukan.dki.csv` pada posisi pertama dengan `"PERIODE"` dan posisi kedua dengan `"PROPINSI"`.

Jika berjalan dengan lancar, Anda akan mendapatkan hasil sebagai berikut.

[1] "PERIODE"	"PROPINSI"	"NAMA.KABUPATEN.KOTA"
[4] "NAMA.KECAMATAN"	"NAMA.KELURAHAN"	"LUAS.WILAYAH..KM2."
[7] "KEPADATAN..JIWA.KM2."	"X"	"X.1"
[10] "X.2"	"X.3"	"X.4"
[13] "X.5"	"X.6"	"X.7"
[16] "X.8"	"X.9"	"X.10"
[19] "X.11"	"X35.39.Laki.Laki"	"X35.39.Perempuan"
[22] "X40.44.Laki.Laki"	"X40.44.Perempuan"	"X45.49.Laki.Laki"
[25] "X45.49.Perempuan"	"X50.54.Laki.Laki"	"X50.54.Perempuan"
[28] "X55.59.Laki.Laki"	"X55.59.Perempuan"	"X60.64.Laki.Laki"
[31] "X60.64.Perempuan"	"X65.69.Laki.Laki"	"X65.69.Perempuan"
[34] "X70.74.Laki.Laki"	"X70.74.Perempuan"	"X.75.Laki.Laki"
[37] "X.75..Perempuan"		

## Code Editor

#Membaca dataset csv

```
penduduk.dki.csv <-  
read.csv("https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.csv", sep=",")
```

```
names(penduduk.dki.csv)[1] <- "PERIODE" #[...1...]
```

```
names(penduduk.dki.csv)[2] <- "PROPINSI" #[...2...]
```

```
names(penduduk.dki.csv)
```

## Console

```
> #Membaca dataset csv  
> penduduk.dki.csv <- read.csv("https://academy.dqlab.id/dataset/dkikepadatankelurahan  
2013.csv", sep=",")  
  
> names(penduduk.dki.csv)[1] <- "PERIODE" #[...1...]  
> names(penduduk.dki.csv)[2] <- "PROPINSI" #[...2...]  
  
> names(penduduk.dki.csv)  
[1] "PERIODE"          "PROPINSI"          "NAMA.KABUPATEN.KOTA"  
[4] "NAMA.KECAMATAN"   "NAMA.KELURAHAN"    "LUAS.WILAYAH..KM2."  
[7] "KEPADATAN..JIWA.KM2." "X"                  "X.1"  
[10] "X.2"              "X.3"               "X.4"  
[13] "X.5"              "X.6"               "X.7"  
[16] "X.8"              "X.9"               "X.10"  
[19] "X.11"             "X35.39.Laki.Laki"   "X35.39.Perempuan"  
[22] "X40.44.Laki.Laki" "X40.44.Perempuan"  "X45.49.Laki.Laki"  
[25] "X45.49.Perempuan" "X50.54.Laki.Laki"   "X50.54.Perempuan"  
[28] "X55.59.Laki.Laki" "X55.59.Perempuan"  "X60.64.Laki.Laki"  
[31] "X60.64.Perempuan" "X65.69.Laki.Laki"   "X65.69.Perempuan"  
[34] "X70.74.Laki.Laki" "X70.74.Perempuan"  "X.75.Laki.Laki"  
[37] "X.75..Perempuan"
```



# Merubah Sejumlah Nama Kolom

Perintah untuk merubah dua kolom pada praktek sebelumnya bisa disingkat dengan konstruksi berikut:

```
names(variable)[c(rentang_posisi)] <- c("nama_baru_1", "nama_baru_2")
```

Dengan `rentang_posisi` adalah daftar nomor dari posisi indeks dari nama kolom yang ingin diubah. Berikut adalah contoh dimana kita merubah nama kolom dari `TAHUN` ke `PERIODE` (posisi 1) dari dataset kita.

```
names(penduduk.dki.csv)[c(1:2)] <- c("PERIODE", "PROPINSI")
```

## Tugas Praktek

Gantilah bagian [...1...] dengan perintah yang merubah nama kolom dari dataset `kependudukan.dki.csv` pada posisi pertama dengan `"PERIODE"` dan posisi kedua dengan `"PROPINSI"`.

Jika berjalan dengan lancar, Anda akan mendapatkan hasil sebagai berikut.

```
[1] "PERIODE"          "PROPINSI"          "NAMA.KABUPATEN.KOTA"
[4] "NAMA.KECAMATAN"   "NAMA.KELURAHAN"    "LUAS.WILAYAH..KM2."
[7] "KEPADATAN..JIWA.KM2." "X"                  "X.1"
[10] "X.2"              "X.3"                "X.4"
[13] "X.5"              "X.6"                "X.7"
[16] "X.8"              "X.9"                "X.10"
[19] "X.11"             "X35.39.Laki.Laki"   "X35.39.Perempuan"
[22] "X40.44.Laki.Laki" "X40.44.Perempuan"   "X45.49.Laki.Laki"
[25] "X45.49.Perempuan" "X50.54.Laki.Laki"   "X50.54.Perempuan"
[28] "X55.59.Laki.Laki" "X55.59.Perempuan"   "X60.64.Laki.Laki"
[31] "X60.64.Perempuan" "X65.69.Laki.Laki"   "X65.69.Perempuan"
[34] "X70.74.Laki.Laki" "X70.74.Perempuan"   "X.75.Laki.Laki"
[37] "X.75..Perempuan"
```

## Code Editor

## #Membaca dataset csv

```
penduduk.dki.csv <-
read.csv("https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.csv", sep=",")

names(penduduk.dki.csv)[c(1:2)] <- c("PERIODE", "PROPINSI") #[...1...]

names(penduduk.dki.csv)
```

## Console

```
> #Membaca dataset csv
> penduduk.dki.csv <- read.csv("https://academy.dqlab.id/dataset/dkikepadatankelurahan
2013.csv", sep=",")

> names(penduduk.dki.csv)[c(1:2)] <- c("PERIODE", "PROPINSI") #[...1...]

> names(penduduk.dki.csv)
[1] "PERIODE"          "PROPINSI"          "NAMA.KABUPATEN.KOTA"
[4] "NAMA.KECAMATAN"   "NAMA.KELURAHAN"    "LUAS.WILAYAH..KM2."
[7] "KEPADATAN..JIWA.KM2." "X"                  "X.1"
[10] "X.2"              "X.3"               "X.4"
[13] "X.5"              "X.6"               "X.7"
[16] "X.8"              "X.9"               "X.10"
[19] "X.11"             "X35.39.Laki.Laki"  "X35.39.Perempuan"
[22] "X40.44.Laki.Laki" "X40.44.Perempuan" "X45.49.Laki.Laki"
[25] "X45.49.Perempuan" "X50.54.Laki.Laki"  "X50.54.Perempuan"
[28] "X55.59.Laki.Laki" "X55.59.Perempuan" "X60.64.Laki.Laki"
[31] "X60.64.Perempuan" "X65.69.Laki.Laki"  "X65.69.Perempuan"
[34] "X70.74.Laki.Laki" "X70.74.Perempuan" "X.75.Laki.Laki"
[37] "X.75..Perempuan"
```

# Membuang Kolom dengan Bantuan Operator %in%

Pada bab sebelumnya, data.frame hasil pembacaan read.csv memiliki kolom X, X.1, X.2, X.3 s/d kolom X.11 yang kosong semua dan sebenarnya tidak diperlukan.

Kita dapat menghilangkan kolom-kolom ini dengan contoh berikut:

```
penduduk.dki.csv <- penduduk.dki.csv[,!names(penduduk.dki.csv) %in% c("X", "X.1","X.2", "X.3", "X.4", "X.5", "X.6", "X.7", "X.8", "X.9", "X.10")]
```

Penjelasan tiap elemen dari perintah tersebut adalah sebagai berikut.

Komponen	Deskripsi
penduduk.dki.csv	Variable data frame dari hasil pembacaan dataset
[ , ...]	Artinya data frame mengambil kolom ke ...
!	Tanda bukan
names(penduduk.dki.csv)	Daftar dari semua kolom dari variable penduduk.dki.csv
%in%	Operator untuk mengambil data dari vector setelahnya
c("X", "X.1", "X.2", "X.3", "X.4", "X.5", "X.6", "X.7", "X.8", "X.9", "X.10")	Vector dari nama semua kolom yang akan dibuang

Atau terjemahan dari seluruh konstruksi tersebut adalah mengambil data dari data.frame penduduk.dki.csv dengan kolom-kolom yang tidak termasuk pada "X", "X.1", "X.2", "X.3", dan seterusnya sampai dengan "X.10".

## Tugas Praktek

Gantilah bagian [...1...] pada code editor untuk membaca file Excel seperti yang ditunjukkan pada contoh di Lesson, namun dengan tambahan kolom "X.11".

### Code Editor

#Membaca dataset csv

```
penduduk.dki.csv <-  
read.csv("https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.csv", sep=",")
```

#Membuang kolom X, X.1, X.2 s/d X.11

```
penduduk.dki.csv <- penduduk.dki.csv[!names(penduduk.dki.csv) %in% c("X",  
"X.1", "X.2", "X.3", "X.4", "X.5", "X.6", "X.7", "X.8", "X.9", "X.10", "X.11")] # [...1...]
```

```
names(penduduk.dki.csv)
```

### Console

```
> #Membaca dataset csv  
> penduduk.dki.csv <- read.csv("https://academy.dqlab.id/dataset/dkikepadatankelurahan  
2013.csv", sep=",")  
  
> #Membuang kolom X, X.1, X.2 s/d X.11  
> penduduk.dki.csv <- penduduk.dki.csv[, !names(penduduk.dki.csv) %in% c("X", "X.1", "X  
.2", "X.3", "X.4", "X.5", "X.6", "X.7", "X.8", "X.9", "X.10", "X.11")] # [...1...]  
  
> names(penduduk.dki.csv)
```

[1]	"TAHUN"	"NAMA.PROVINSI"	"NAMA.KABUPATEN.KOTA"
[4]	"NAMA.KECAMATAN"	"NAMA.KELURAHAN"	"LUAS.WILAYAH..KM2."
[7]	"KEPADATAN..JIWA.KM2."	"X35.39.Laki.Laki"	"X35.39.Perempuan"
[10]	"X40.44.Laki.Laki"	"X40.44.Perempuan"	"X45.49.Laki.Laki"
[13]	"X45.49.Perempuan"	"X50.54.Laki.Laki"	"X50.54.Perempuan"
[16]	"X55.59.Laki.Laki"	"X55.59.Perempuan"	"X60.64.Laki.Laki"
[19]	"X60.64.Perempuan"	"X65.69.Laki.Laki"	"X65.69.Perempuan"
[22]	"X70.74.Laki.Laki"	"X70.74.Perempuan"	"X.75.Laki.Laki"
[25]	"X.75..Perempuan"		

# Merubah Tipe Kolom menjadi Factor

Kita akan beralih sekarang ke dataset kependudukan DKI versi Excel, dimana terdapat perbedaan perilaku antara read.xlsx dan read.csv, yaitu secara default read.xlsx tidak mengkonversi kolom kategorik sebagai teks (character).

Untuk melakukan konversi sebagai factor, kita gunakan function as.factor. Berikut adalah konstruksi konversi dari satu kolom data.frame.

```
as.factor(data.frame$namakolom)
```

## Tugas Praktek

Gantilah bagian [...1...] pada code editor untuk melakukan konversi kolom NAMA.PROVINSI dari variable penduduk.dki.xlsx.

Jika berjalan dengan lancar, maka hasilnya akan tampak sebagai berikut – perhatikan jika NAMA.PROVINSI sudah bertipe Factor, sedangkan kolom lain seperti NAMA.KABUPATEN/KOTA yang harusnya bertipe Factor masih bertipe character (chr).

## Code Editor

```
library(openxlsx)

#Membaca dataset dengan read.xlsx dan dimasukkan ke variable penduduk.dki

penduduk.dki.xlsx <-
read.xlsx(xlsxFile="https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.xlsx")

penduduk.dki.xlsx$NAMA.PROVINSI <- as.factor(penduduk.dki.xlsx$NAMA.PROVINSI) #[...1...]

str(penduduk.dki.xlsx)
```

## Console

```
> library(openxlsx)

> #Membaca dataset dengan read.xlsx dan dimasukkan ke variable penduduk.dki
> penduduk.dki.xlsx <- read.xlsx(xlsxFile="https://academy.dqlab.id/dataset/dkikepada
tanelurahan2013.xlsx")

> penduduk.dki.xlsx$NAMA.PROVINSI <- as.factor(penduduk.dki.xlsx$NAMA.PROVINSI) #[...
1...]

> str(penduduk.dki.xlsx)
'data.frame':   267 obs. of  25 variables:
 $ TAHUN      : num  2013 2013 2013 2013 2013 ...
```

```

$ NAMA.PROVINSI      : Factor w/ 1 level "PROVINSI DKI JAKARTA": 1 1 1 1 1 1 1 1 1
1 ...
$ NAMA.KABUPATEN/KOTA : chr  "KAB.ADM.KEP.SERIBU" "KAB.ADM.KEP.SERIBU" "KAB.ADM.KEP.
SERIBU" "KAB.ADM.KEP.SERIBU" ...
$ NAMA.KECAMATAN      : chr  "KEP. SERIBU UTR" "KEP. SERIBU UTR" "KEP. SERIBU UTR" "
KEP. SERIBU SLT" ...
$ NAMA.KELURAHAN      : chr  "P. PANGGANG" "P. KELAPA" "P. HARAPAN" "P. UNTUNG JAWA"
...
$ LUAS.WILAYAH.(KM2)  : num  0.91 3.76 3.59 0.59 1.57 1.39 2.58 1.26 1.12 1.14 ...
$ KEPADATAN.(JIWA/KM2): num  6779 1705 628 3625 3084 ...
$ 35-39.Laki-Laki     : num  231 84 255 199 98 113 166 850 954 752 ...
$ 35-39.Perempuan     : num  235 88 238 185 75 112 174 748 920 675 ...
$ 40-44.Laki-Laki     : num  233 99 232 178 73 108 130 749 914 691 ...
$ 40-44.Perempuan     : num  210 88 234 176 94 80 165 798 943 691 ...
$ 45-49.Laki-Laki     : num  171 72 212 162 67 66 176 779 871 659 ...
$ 45-49.Perempuan     : num  158 63 193 139 69 62 162 766 823 631 ...
$ 50-54.Laki-Laki     : num  137 34 150 100 60 61 129 715 736 611 ...
$ 50-54.Perempuan     : num  126 29 161 119 40 63 97 662 679 514 ...
$ 55-59.Laki-Laki     : num  98 30 139 97 37 37 108 614 680 539 ...
$ 55-59.Perempuan     : num  106 39 101 83 32 36 90 537 510 466 ...
$ 60-64.Laki-Laki     : num  72 29 73 58 22 32 88 555 544 428 ...
$ 60-64.Perempuan     : num  65 24 56 56 13 26 42 343 421 279 ...
$ 65-69.Laki-Laki     : num  36 12 18 40 18 21 68 413 398 328 ...
$ 65-69.Perempuan     : num  33 21 35 54 15 14 34 215 235 160 ...
$ 70-74.Laki-Laki     : num  33 13 24 26 10 17 37 259 241 215 ...
$ 70-74.Perempuan     : num  20 5 25 27 18 11 32 142 132 116 ...
$ >75.Laki-Laki       : num  13 5 18 16 11 8 34 214 215 150 ...
$ >75.Perempuan       : num  27 8 26 13 17 7 23 165 159 121 ...

```

# Mengambil Kolom Laki.Laki / Perempuan dengan grep

Kalau kita perhatikan pada dataset kependudukan DKI, kolom-kolom untuk jumlah penduduk berdasarkan umur dan jenis kelamin bentuknya adalah seperti berikut.

T	U	V	W	X	Y	Z
35-39 Laki-Laki	35-39 Perempuan	40-44 Laki-Laki	40-44 Perempuan	45-49 Laki-Laki	45-49 Perempuan	50-54 Laki
231	235	233	210	171	158	
84	88	99	88	72	63	
255	238	232	234	212	193	
199	185	178	176	162	139	
98	75	73	94	67	69	
113	112	108	80	66	62	
166	174	130	165	176	162	
850	748	749	798	779	766	
954	920	914	943	871	823	
752	675	691	691	659	631	
592	491	447	522	463	487	

Dengan ada pola nama seperti itu, kita bisa mengambil dengan function grep dengan syntax berikut.

```
grep(pattern="pola", x = vector, ignore.case=TRUE)
```

Pola teks yang digunakan oleh grep adalah menggunakan regular expression (regex). Regex sendiri adalah konstruksi pola yang sangat komplis untuk mengolah teks, namun untuk case kita gunakan teks sederhana saja sebagai berikut:

- ❑ "laki-laki": untuk mengambil pola kolom yang mengandung teks Laki-Laki.
- ❑ "perempuan": untuk mengambil pola kolom yang mengandung teks "Perempuan".
- ❑ "(perempuan|laki-laki)": untuk mengambil pola kolom yang mengandung teks "Perempuan" ataupun "Laki-Laki".

Argumen x dimana kita memasukkan vector. Dan ignore.case = TRUE menyatakan bahwa pola yang kita tidak bersifat case sensitive – artinya huruf besar tidak dibedakan dari huruf kecil.

Function grep ini akan mengembalikan posisi-posisi indeks dari vector yang ditemukan polanya.

Sebagai contoh, untuk mengambil kolom yang mengandung "Perempuan" pada nama-nama kolom dari dataset kependudukan kita gunakan perintah berikut:

```
grep(pattern="perempuan", x = names(penduduk.dki.xlsx), ignore.case=TRUE)
```

Ini akan menghasilkan vector berikut.

```
[1] 9 11 13 15 17 19 21 23 25
```

Yang merupakan daftar posisi dari nama kolom yang mengandung teks "perempuan".

### Tugas Praktek

Ganti bagian [...1...] dengan function grep menampilkan nama-nama kolom yang mengandung kata "perempuan" - seluruhnya menggunakan huruf kecil.

Kemudian pada bagian [...2...] tampilkan data-data dari nama-nama kolom yang mengandung kata "laki-laki" - seluruhnya menggunakan huruf kecil..

Code Editor

```
library(openxlsx)

penduduk.dki.xlsx <-
read.xlsx(xlsxFile="https://academy.dqlab.id/dataset/dkikepadatan_kelurahan2013.xlsx")

#Tampilkan nama-nama kolom yang mengandung kata "perempuan".

pola_nama_perempuan <- grep(pattern="perempuan", x = names(penduduk.dki.xlsx),
ignore.case=TRUE) #[...1...]

names(penduduk.dki.xlsx[pola_nama_perempuan])

#Tampilkan nama-nama kolom yang mengandung kata "laki-laki"

pola_nama_laki_laki <- grep(pattern="laki-laki", x = names(penduduk.dki.xlsx),
ignore.case=TRUE) #[...2...]

names(penduduk.dki.xlsx[pola_nama_laki_laki])
```

Console

```
> library(openxlsx)

> penduduk.dki.xlsx <- read.xlsx(xlsxFile="https://academy.dqlab.id/dataset/dkikepada
tan_kelurahan2013.xlsx")

> #Tampilkan nama-nama kolom yang mengandung kata "perempuan".
> pola_nama_perempuan <- grep(pattern="perempuan", x = names(penduduk.dki.xlsx), igno
re.case=TRUE) #[...1...]

> names(penduduk.dki.xlsx[pola_nama_perempuan])
[1] "35-39.Perempuan" "40-44.Perempuan" "45-49.Perempuan" "50-54.Perempuan"
```



```
[5] "55-59.Perempuan" "60-64.Perempuan" "65-69.Perempuan" "70-74.Perempuan"
[9] ">75.Perempuan"

> #Tampilkan nama-nama kolom yang mengandung kata "laki-laki"
> pola_nama_laki_laki <- grep(pattern="laki-laki", x = names(penduduk.dki.xlsx), ignore.case=TRUE) #[...2...]

> names(penduduk.dki.xlsx[pola_nama_laki_laki])
[1] "35-39.Laki-Laki" "40-44.Laki-Laki" "45-49.Laki-Laki" "50-54.Laki-Laki"
[5] "55-59.Laki-Laki" "60-64.Laki-Laki" "65-69.Laki-Laki" "70-74.Laki-Laki"
[9] ">75.Laki-Laki"
```

# Menambahkan kolom Penjumlahan

Kita dapat menambahkan kolom pada data.frame langsung dengan menggunakan simbol subscript \$ diikuti dengan nama kolom baru yang kita kehendaki. Misalkan untuk membuat kolom PEREMPUAN35TAHUNKEATAS pada variable penduduk.dki.xlsx, kita bisa menuliskan:

```
penduduk.dki.xlsx$PEREMPUAN35TAHUNKEATAS
```

Kolom baru ini kita bisa isi, misalkan dengan penjumlahan dari kolom-kolom dari dataset itu sendiri menggunakan function rowSums dari kolom-kolom data.frame. Syntaxnya adalah sebagai berikut:

```
rowSums(data.frame)
```

Sebagai contoh – menyambung dari praktek sebelumnya juga – untuk menjumlahkan kolom yang mengandung teks "perempuan" sebagai berikut:

```
rowSums(penduduk.dki.xlsx[,grep(pattern="perempuan", x = names(penduduk.dki.xlsx), ignore.case=TRUE)])
```

Dan untuk menambahkan hasil penjumlahan ini ke kolom baru di atas, kita gabungkan perintahnya dalam bentuk sebagai berikut.

```
penduduk.dki.xlsx$PEREMPUAN35TAHUNKEATAS <- rowSums(penduduk.dki.xlsx[grep(pattern="perempuan", x = names(penduduk.dki.xlsx), ignore.case=TRUE)])
```

## Tugas Praktek

Ganti bagian [...1...] dan [...2...] dengan perintah menjumlahkan dataset dengan nama-nama kolom yang mengandung kata "perempuan". Hasilnya masukkan sebagai kolom baru dengan nama **PEREMPUAN35TAHUNKEATAS** dari dataset **penduduk.dki.xlsx**.

Perhatikan juga bahwa pola untuk nama kolom mengandung teks "perempuan" sudah disimpan ke variable bernama **pola\_nama\_perempuan**. Gunakan variable tersebut sebagai bagian jawaban.

Code Editor

```
library(openxlsx)
```

```
penduduk.dki.xlsx <-  
read.xlsx(xlsxFile="https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.xlsx")
```

```
#Tampilkan nama-nama kolom yang mengandung kata "perempuan".
```

```
pola_nama_perempuan <- grep(pattern="perempuan", x = names(penduduk.dki.xlsx),  
ignore.case=TRUE)
```

```
penduduk.dki.xlsx$PEREMPUAN35TAHUNKEATAS <-  
rowSums(penduduk.dki.xlsx[pola_nama_perempuan])
```

#### Console

```
> library(openxlsx)  
  
> penduduk.dki.xlsx <- read.xlsx(xlsxFile="https://academy.dqlab.id/dataset/dkikepada  
tankeurahan2013.xlsx")  
  
> #Tampilkan nama-nama kolom yang mengandung kata "perempuan".  
> pola_nama_perempuan <- grep(pattern="perempuan", x = names(penduduk.dki.xlsx), ignore.case=TRUE)  
  
> penduduk.dki.xlsx$PEREMPUAN35TAHUNKEATAS <- rowSums(penduduk.dki.xlsx[pola_nama_perempuan])
```

# Normalisasi Data dari Kolom ke Baris

Kalau kita perhatikan lima kolom data pada table berikut di bawah ini – yang diambil dari dataset kependudukan DKI – maka dua kolom terakhir (berwarna kuning dan hijau) sebenarnya mengandung tiga variable, yaitu: rentang umur, jenis kelamin, dan jumlah penduduk.

NAMA.KECAMATAN	NAMA.KELURAHAN	35-39.Laki-Laki	35-39.Perempuan
GAMBIR	GAMBIR	166	174
GAMBIR	CIDENG	850	748
GAMBIR	PETOJO UTARA	954	920

Dan dari sisi prinsip kerapian data, satu variable data harusnya masing-masing menempati satu kolom tersendiri. Dan berdasarkan prinsip tersebut, kolom "NAMA.KECAMATAN" dan "NAMA.KELURAHAN" yang berwarna biru sudah memenuhi persyaratan. Namun kolom "35-39.Laki-Laki" dan kolom "35-39.Perempuan" tidak memenuhi persyaratan.

Kita akan coba merapikan ini dengan dua tahap:

- ☐ Melakukan normalisasi kolom dari baris ke kolom
- ☐ Memisahkan kolom

Kita mulai dari tahap pertama pada praktek kali ini, dimana kita akan merubah struktur data di atas menjadi berikut.

NAMA.KECAMATAN	NAMA.KELURAHAN	DEMOGRAFIK	JUMLAH
GAMBIR	GAMBIR	35-39.Laki-Laki	166
GAMBIR	GAMBIR	35-39.Perempuan	174
GAMBIR	CIDENG	35-39.Laki-Laki	850
GAMBIR	CIDENG	35-39.Perempuan	748
GAMBIR	PETOJO UTARA	35-39.Laki-Laki	954
GAMBIR	PETOJO UTARA	35-39.Perempuan	920

Terlihat nama kolom "35-39.Laki-Laki" dan "35-39.Perempuan" dipivot menjadi nilai baris data di bawah kolom "DEMOGRAFIK".

Kemudian angka-angka jumlah penduduk yang tadinya di bawah kedua kolom tersebut sekarang ada di bawah kolom "JUMLAH".

Untuk melakukan transformasi struktur ini, kita akan gunakan function **melt** dari package **reshape2**. Berikut adalah contoh penggunaannya:

```
melt(data=penduduk.dki.xlsx, id.vars=c( "NAMA.KECAMATAN", "NAMA.KELURAHAN"), measure.
vars = c("35-39.Laki-Laki", "35-39.Perempuan"), variable.name = "DEMOGRAFIK", value.n
ame="JUMLAH")
```

Penjelasan dari perintah tersebut adalah sebagai berikut:

Elemen	Deskripsi
melt	Function untuk melakukan transformasi dari kolom ke baris
data=penduduk.dki.xlsx	Argumen data yang diisi dengan variable penduduk.dki.xlsx, yaitu data frame hasil pembacaan dari file Excel data kependudukan DKI
id.vars= c("NAMA.KECAMATAN", "NAMA.KELURAHAN")	Vector dari field-field kolom yang akan menjadi identitas – dimana field tidak akan mengalami transformasi tapi tetap diambil
measure.vars = c("35-39.Laki-Laki", "35-39.Perempuan")	Nama variable hasil transformasi
variable.name="DEMOGRAFIK"	Nama variable untuk menampung nama kolom transformasi
value.name="JUMLAH"	Nama variable untuk menampung isi data dari kolom transformasi

### Tugas Praktek

Ganti bagian [...1...] dengan perintah melt yang sama persis dengan contoh di Lesson.

Jika semua berjalan lancar, maka potongan tampilan hasil transformasi data akan terlihat sebagai berikut.

```
> penduduk.dki.transform
```

	NAMA.KECAMATAN	NAMA.KELURAHAN	DEMOGRAFIK	JUMLAH
1	KEP. SERIBU UTR	P. PANGGANG	35-39.Laki-Laki	231
2	KEP. SERIBU UTR	P. KELAPA	35-39.Laki-Laki	84
3	KEP. SERIBU UTR	P. HARAPAN	35-39.Laki-Laki	255
4	KEP. SERIBU SLT	P. UNTUNG JAWA	35-39.Laki-Laki	199
5	KEP. SERIBU SLT	P. TIDUNG	35-39.Laki-Laki	98
6	KEP. SERIBU SLT	P. PARI	35-39.Laki-Laki	113
7	GAMBIR	GAMBIR	35-39.Laki-Laki	166
8	GAMBIR	CIDENG	35-39.Laki-Laki	850
...				
...				
268	KEP. SERIBU UTR	P. PANGGANG	35-39.Perempuan	235
269	KEP. SERIBU UTR	P. KELAPA	35-39.Perempuan	88
270	KEP. SERIBU UTR	P. HARAPAN	35-39.Perempuan	238
271	KEP. SERIBU SLT	P. UNTUNG JAWA	35-39.Perempuan	185

272	KEP. SERIBU SLT	P. TIDUNG	35-39.Perempuan	75
273	KEP. SERIBU SLT	P. PARI	35-39.Perempuan	112
274	GAMBIR	GAMBIR	35-39.Perempuan	174
275	GAMBIR	CIDENG	35-39.Perempuan	748
...				
...				
534	CIPAYUNG	CEGER	35-39.Perempuan	930

## Code Editor

```
library(openxlsx)
```

```
library(reshape2)
```

```
penduduk.dki.xlsx <-  
read.xlsx(xlsxFile="https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.xlsx")
```

```
#Transformasi kolom dataset penduduk.dki.xlsx, disimpan ke variable penduduk.dki.transform
```

```
penduduk.dki.transform <- melt(data=penduduk.dki.xlsx, id.vars=c( "NAMA.KECAMATAN",  
"NAMA.KELURAHAN"), measure.vars = c("35-39.Laki-Laki", "35-39.Perempuan"),  
variable.name = "DEMOGRAFIK", value.name="JUMLAH")
```

```
#Menampilkan variable penduduk.dki.transform
```

```
penduduk.dki.transform
```

## Console

```
(LANGSUNG PRAKTEK)
```

# Split Fields

Pada praktek sebelumnya, kita telah melakukan transformasi data kolom ke baris seperti berikut.

NAMA.KECAMATAN	NAMA.KELURAHAN	DEMOGRAFIK	JUMLAH
GAMBIR	GAMBIR	35-39.Laki-Laki	166
GAMBIR	GAMBIR	35-39.Perempuan	174
GAMBIR	CIDENG	35-39.Laki-Laki	850
GAMBIR	CIDENG	35-39.Perempuan	748
GAMBIR	PETOJO UTARA	35-39.Laki-Laki	954
GAMBIR	PETOJO UTARA	35-39.Perempuan	920

Tahap selanjutnya adalah memisahkan kolom "DEMOGRAFIK" yang memiliki dua informasi ini menjadi dua kolom – yaitu "RENTANG UMUR" dan "JENIS KELAMIN".

NAMA.KECAMATAN	NAMA.KELURAHAN	RENTANG.UMUR	JENIS.KELAMIN	JUMLAH
GAMBIR	GAMBIR	35-39.	Laki-Laki	166
GAMBIR	GAMBIR	35-39	Perempuan	174
GAMBIR	CIDENG	35-39	Laki-Laki	850
GAMBIR	CIDENG	35-39	Perempuan	748
GAMBIR	PETOJO UTARA	35-39	Laki-Laki	954
GAMBIR	PETOJO UTARA	35-39	Perempuan	920

Untuk mencapai tujuan ini, kita akan menggunakan tiga konstruksi perintah.

Perintah pertama adalah menggunakan function `colsplit` sebagai berikut.

```
colsplit(penduduk.dki.transform$DEMOGRAFIK,"\\.",c("RENTANG.UMUR","JENIS.KELAMIN"))
```

Berikut adalah penjelasan dari perintah di atas.

Komponen	Deskripsi
<code>colsplit</code>	Function untuk memecah satu variable menjadi beberapa variable
<code>penduduk.dki.transform\$DEMOGRAFIK</code>	Artinya data frame mengambil kolom ke ...
<code>"\\."</code>	Karakter pemisah, dalam hal ini tanda titik (.). Khusus untuk tanda titik harus diisi dengan didahului backslash dua kali
<code>c("RENTANG.UMUR", "JENIS.KELAMIN")</code>	Vector yang berisi nama-nama variable baru hasil pemecahan variable

Perintah kedua adalah perintah untuk memasukkan variable hasil pecahan ke dalam dua kolom di data.frame dengan cara berikut.

```
penduduk.dki.transform[c("RENTANG.UMUR", "JENIS.KELAMIN")] <- ...
```

Berikut adalah penjelasan dari perintah di atas.

Komponen	Deskripsi
penduduk.dki.transform	Artinya data frame mengambil kolom ke ...
[...]	Simbol karakter untuk index
c("RENTANG.UMUR", "JENIS.KELAMIN")	Vector yang berisi nama-nama variable baru

Dan perintah ketiga adalah menghilangkan kolom DEMOGRAFIK dari data frame sebagai berikut.

```
penduduk.dki.transform$DEMOGRAFIK <- NULL
```

Dengan memasukkan NULL maka kolom akan DEMOGRAFIK akan dihilangkan dari data.frame penduduk.dki.transform.

### Tugas Praktek

Ganti bagian [...1...] pada code editor dengan perintah untuk memecah isi dari field "DEMOGRAFIK" menjadi "RENTANG.UMUR" dan "JENIS.KELAMIN". Tanda pemisah adalah tanda titik.

Kemudian ganti [...2...] dengan penambahan kolom "RENTANG.UMUR" dan "JENIS.KELAMIN" data.frame penduduk.dki.xlsx.

Terakhir, ganti [...3...] dengan kolom "DEMOGRAFIK" dari variable penduduk.dki.transform.

Jika semua berjalan lancar, maka potongan tampilan hasil split data akan terlihat sebagai berikut.

	NAMA.KECAMATAN	NAMA.KELURAHAN	JUMLAH	RENTANG.UMUR	JENIS.KELAMIN
1	KEP. SERIBU UTR	P. PANGGANG	231	35-39	Laki-Laki
2	KEP. SERIBU UTR	P. KELAPA	84	35-39	Laki-Laki
3	KEP. SERIBU UTR	P. HARAPAN	255	35-39	Laki-Laki
4	KEP. SERIBU SLT	P. UNTUNG JAWA	199	35-39	Laki-Laki
5	KEP. SERIBU SLT	P. TIDUNG	98	35-39	Laki-Laki
6	KEP. SERIBU SLT	P. PARI	113	35-39	Laki-Laki



## Code Editor

```
library(openxlsx)

library(reshape2)

penduduk.dki.xlsx <-
read.xlsx(xlsxFile="https://academy.dqlab.id/dataset/dkikepadatankelurahan2013.xlsx")

penduduk.dki.transform <- melt(data=penduduk.dki.xlsx, id.vars=c( "NAMA.KECAMATAN",
"NAMA.KELURAHAN"), measure.vars = c("35-39.Laki-Laki", "35-39.Perempuan"),
variable.name = "DEMOGRAFIK", value.name="JUMLAH")

#Memecah isi kolom DEMOGRAFIK menjadi "RENTANG.UMUR" dan "JENIS.KELAMIN"

penduduk.dki.transform[c("RENTANG.UMUR", "JENIS.KELAMIN")] <-
colsplit(penduduk.dki.transform$DEMOGRAFIK,"\\.",c("RENTANG.UMUR","JENIS.KELAMIN"))

penduduk.dki.transform$DEMOGRAFIK <- NULL

penduduk.dki.transform
```

## Console

(LANGSUNG PRAKTEK)

# Kesimpulan

Pada bab ini Anda telah mempelajari bagaimana merubah struktur suatu data.frame sehingga nantinya akan lebih mudah untuk diolah pada proses berikutnya.

Operasi perubahan itu antara lain adalah:

- ☐ Merubah nama satu dan beberapa kolom
- ☐ Membuang kolom
- ☐ Merubah tipe kolom
- ☐ Mengambil kolom dengan pola tertentu
- ☐ Menambah kolom
- ☐ Normalisasi atau transpose data
- ☐ Memecah data dengan split data

Kami sarankan agar Anda sering melakukan latihan untuk hal ini. Sehingga pada saat masuk ke materi Part 2 dari data wrangling akan lebih lancar.

Klik tombol Next untuk melanjutkan ke bab Penutup.

# Penutup - Part 1

Selamat, Anda telah menuntaskan bagian pertama dari course Data Wrangling with R yang mempelajari dan mempraktekkan materi berikut:

- ☐ Missing Value.
- ☐ Struktur data kategori bernama Factor.
- ☐ Membaca file-file teks dan Excel – yang paling banyak ditemui sehari-hari.
- ☐ Melakukan perubahan struktur data sehingga cocok digunakan lebih lanjut.

## What Next?

Setelah course Part 1 ini selesai, maka fokus selanjutnya pada Part 2 adalah melakukan transformasi isi dimana kita perlu bekerja dengan teks, tanggal, angka, dan missing value.

Selain itu, pada Part 2 juga akan dipraktekkan metode pembacaan data dari database, enrichment dan pencarian data duplikat.

Klik tombol Next untuk menyelesaikan course ini.