Pendahuluan

Kegiatan menganalisa data adalah kegiatan untuk mencari korelasi dan menginvestigasi data. Korelasi dan investigasi akan mendapatkan hasil yang tepat apabila data yang digunakan sudah terstruktur dengan baik. Untuk mendapat data yang siap digunakan untuk analisa (terstruktur dengan baik), maka diperlukan persiapan data. Mempersiapkan data adalah bagian penting dalam analisa data.

Dalam materi kedua ini, kita akan belajar bagaimana cara mempersiapkan data menggunakan R. Namun demikian, untuk mempersiapkan data tidak harus menggunakan R. Banyak tools lain yang bisa digunakan untuk mempersiapkan data. Penggunaan R dalam persiapan data akan memudahkan kita dalam mengolah data apabila data yang kita gunakan berukuran besar.

1. Renaming Data

Penggunaan data frame memudahkan kita untuk memilih dan memfilter data berdasarkan nama baris atau kolom. Namun demikian, tidak semua data memiliki nama baris dan kolom sehingga diperlukan adanya renaming nama dataset. Contoh renaming data dilakukan dengan langkah sebagai berikut:

a. Baca data *karyawan* dan *gaji* yang telah dikerjakan pada tugas sebelumnya.

b. Gunakan fungsi names untuk melihat nama kolom

```
> names(karyawan)
[1] "X1" "X2" "X3" "X4" "X5" "X6"
> names(gaji)
[1] "X1" "X2" "X3" "X4"
```

c. Rename nama kolom tersebut dengan menggunakan *names* vector

```
names(karyawan) = c("ID", "Tgl Lahir", "Nama Depan","Nama Belakang",
                    "Jenis Kelamin", "Tgl Mulai Kerja")
```

d. Selain menggunakan *names*, bisa juga menggunakan fungsi *colnames*

```
colnames(gaji) = c("ID","Gaji","Mulai Kerja","Akhir Kerja")
```

2. Konversi Tipe Data

Data yang berhasil dibaca atau ter-import terkadang tidak sesuai dengan tipe data aslinya. Dengan demikian, sebelum melakukan pemrosesan data perlu dilakukan pengecekan tipe data, dan mengkonversi data tersebut apabila ada tipe data yang tidak sesuai. Langkah melakukan pengecekan dan pengkorvesian data adalah sebagai berikut:

a. Periksa setiap atribut data menggunakan fungsi *class*. Misal kita periksa atribut *Tgl Lahir* dari data *karyawan*

```
> class(karyawan$`Tgl Lahir`)
[1] "factor"
```

b. Atau periksa seluruh atribut secara bersamaan menggunakan fungsi *str*

```
> str(karyawan)
'data.frame':   10 obs. of  6 variables:
 $ ID             : int  10001 10002 10003 10004 10005 10006 10007 1(
 $ Tgl Lahir      : Factor w/ 10 levels "1952-04-19","1953-04-20",..:
 $ Nama Depan     : Factor w/ 10 levels "Anneke","Bezalel",..: 5 2 7
 $ Nama Belakang  : Factor w/ 10 levels "Bamford","Facello",..: 2 9 1
 $ Jenis Kelamin  : Factor w/ 2 levels "F","M": 2 1 2 2 2 1 1 2 1 1
 $ Tgl Mulai Kerja: Factor w/ 10 levels "1985-02-18","1985-11-21",..:
```

c. Apabila ada yang tidak sesuai dengan data aslinya, maka tipe data harus diubah. Hasil dari point b menunjukan atribut *Tgl Lahir* dan *Tgl Mulai Kerja* bertipe factor. Untuk merubah menjadi tipe date, maka digunakan *as.Date*

```
karyawan$`Tgl Lahir`= as.Date(karyawan$`Tgl Lahir`)
karyawan$`Tgl Mulai Kerja`= as.Date(karyawan$`Tgl Mulai Kerja`)
```

d. Atribut *Nama Depan* dan *Nama Belakang* dirubah menjadi tipe character menggunakan *as.character*

```
karyawan$`Nama Depan`= as.character(karyawan$`Nama Depan`)
karyawan$`Nama Belakang`= as.character(karyawan$`Nama Belakang`)
```

e. Lakukan untuk data hal sama pada *gaji.* Cek tipe data setiap atribut, apabila ada tipe data yang tidak sesuai maka rubahlah tipe data yang tidak sesuai tersebut.

```
> str(gaji)
'data.frame':   1000 obs. of  4 variables:
 $ ID          : int  10001 10001 10001 10001 10001 1
 $ Gaji        : int  60117 62102 66074 66596 66961 7
 $ Mulai Kerja: Factor w/ 964 levels "1985-02-18","1
 $ Akhir Kerja: Factor w/ 887 levels "1986-02-18","1
```

```
gaji$`Mulai Kerja`= as.Date(gaji$`Mulai Kerja`)
gaji$`Akhir Kerja`= as.Date(gaji$`Akhir Kerja`)
```

3. Manipulasi Tipe Data Date

Tipe data date dapat dimanipulasi untuk beberapa tujuan, misalnya menambah hari atau untuk menghitung usia seseorang.

a. Menambahkan hari pada tipe data date. Misal menambahkan 10 hari pada atribut *Tgl Mulai Kerja* pada data *karyawan*

```
karyawan$`Tgl Mulai Kerja`+30
```

b. Menghitung selisih hari. Misal untuk mengetahui usia hari saat mulai kerja pada data *karyawan.* Kita lakukan dengan mengurangkan atribut *Tgl Mulai Kerja* dengan *Tgl Lahir*

```
> karyawan$`Tgl Mulai Kerja`- karyawan$`Tgl Lahir`
Time differences in days
 [1] 11985  7842  9765 11902 12653 13192 11586 13357 11993  9581
```

c. Selain mencari selisih hari, kita juga bisa mencari selisih minggu dengan menggunakan fungsi *difftime*

```
> difftime(karyawan$`Tgl Mulai Kerja`,karyawan$`Tgl Lahir`,units = "weeks")
Time differences in weeks
 [1] 1712.143 1120.286 1395.000 1700.286 1807.571 1884.571 1655.143 1908.143
```

d. Untuk menghitung selisih tahun, diperlukan packages *lubridate*

```
install.packages("lubridate")
library(lubridate)
```

e. Gunakan fungsi *interval* untuk mencari selisih

```
time_tahun = interval(ymd(karyawan$`Tgl Lahir`),
                      ymd(karyawan$`Tgl Mulai Kerja`))
```

f. Rubah tipe datanya menjadi periode, kemudian gunakan fungsi *year* untuk menampilkan selisih tahun.

```
> time_tahun = as.period(time_tahun)
> year(time_tahun)
 [1] 32 21 26 32 34 36 31 36 32 26
```

g. Untuk mengecek waktu sekarang bisa menggunakan fungsi *now*

```
> now()
[1] "2019-03-26 23:32:21 +07"
```

h. Kemudian, untuk mengecek usia karyawan dengan menghitung interval waktu sekarang dengan atribut *Tgl Lahir*

```
> usia = interval(ymd(karyawan$`Tgl Lahir`),now())
> usia = as.period(usia)
> year(usia)
 [1] 65 54 59 64 64 65 61 61 66 55
```

4. Menambah Records Baru

Menambah records baru dapat dengan mudah dilakukan dengan menggunakan fungsi *rbind* dan *cbind. rbind* digunakan untuk menambahkan data perbaris sedangkan *cbind* digunakan untuk menambahkan suatu kolom.

a. Menambah record data karyawan

```
karyawan = rbind(karyawan,c(10011,"1990-06-28","Budi",
                            "Darmawan","M","2019-01-01"))
```

b. Menambah kolom Posisi pada data karyawan dengan isi NA

```
karyawan = cbind(karyawan, Posisi = NA)
```

c. Menambah kolom Usia pada data karyawan

```
waktu = interval(ymd(karyawan$`Tgl Lahir`),now())
waktu = as.period(waktu)
karyawan = cbind(karyawan, Usia=year(waktu))
```

5. Memfilter Data

Memfilter data merupakan kemampuan yang harus dimiliki saat akan menganalisa suatu data.

a. Subset tiga data teratas menggunakan *head* dan empat data terbawah menggunakan *tail*.

```
head(karyawan, 3)
tail(karyawan, 4)
```

b. Subset data dari baris 1 sampai 3

```
karyawan[1:3,]
```

c. Subset baris 1 sampai 3 dan kolom 2 sampai kolom 4

```
karyawan[1:3, 2:4]
```

d. Subset baris 2 dan baris 5, kolom satu dan kolom 3

```
karyawan[c(2,5), c(1,3)]
```

e. Atau bisa menggunakan nama kolomnya

```
karyawan[1:3, c("ID","Nama Depan")]
```

f. Menghapus kolom, misal menghapus kolom ke-8

```
karyawan[,-8]
```

g. Menghapus baris, misal menghapus baris ke-1

```
karyawan[-1,]
```

h. Menampilkan data tertentu, misalkan hanya menampilkan data yang berjenis kelamin laki-laki

```
karyawan[karyawan$'Jenis Kelamin' == 'M',]
```

i. Menampilkan data karyawan dg gaji antara 60000-70000

```
gaji[gaji$Gaji >= 60000 & gaji$Gaji < 70000,]
```

j. Menampilkan data karyawan depan nama depan 'Ge'

```
karyawan[substr(karyawan$'Nama Depan',0,2)=="Ge",]
```

6. Merging Data

Merging data bermanfaat untuk mengetahui relasi antara satu data dengan data lainnya. Merging data dilakukan dengan menggunakan fungsi *merge* atau *join.*

a. Menggabungkan dua kolom menjadi satu, misal kolom nama depan dan nama belakang pada data *karyawan* digabung menjadi nama kolom baru *Nama Karyawan*

```
karyawan$`Nama Karyawan`=paste(karyawan$`Nama Depan`,karyawan$`Nama Belakang`)
```

b. Menggabungkan data karyawan dan gaji berdasarkan kolom ID

```
karyawan_gaji = merge(karyawan, gaji,by="ID")
```

c. Atau menggunakan fungsi *join*

```
install.packages("plyr")
library(plyr)
karyawan_gaji = join(karyawan, gaji,by="ID")
```

7. Mensortir Data

Sortir data bermanfaat untuk melihat susunan data sehingga kita bisa melakukan analisa lebih efesien. Untuk mensortir data, kita gunakan fungsi *order* dan *sort*.

a. Pertama, kita coba fungsi *sort* untuk mengurutkan data.

```
a = c(5,1,4,3,2,6,3)
sort(a)
sort(a, decreasing=TRUE)
```

b. Fungsi *order* untuk mengurutkan berdasarkan posisi urutan

```
a = c(5,1,4,3,2,6,3)
order(a)
order(a, decreasing = TRUE)
```

c. Mengurutkan data berdasar kolom tertentu. Misal mengurutkan data berdasarkan kolom *Gaji* pada data gaji

```
sortir_gaji = gaji[order(gaji$Gaji, decreasing = TRUE),]
```

d. Mengurutkan data berdasar dua kolom. Misal kolom *Gaji* dan kolom *Mulai Kerja*

```
sortir_gaji2 = gaji[order(gaji$Gaji, gaji$`Mulai Kerja`, decreasing = TRUE),]
```

8. Reshaping Data

R menyediakan package *reshape2* untuk mengubah bentuk data. Fungsi *dcast* digunakan untuk mengubah data yang panjang ke lebar. Fungsi *melt* digunakan sebaliknya, yaitu mengubah data yang lebar ke panjang.

a. Install dan load *reshape2*

```
install.packages("reshape2")
library(reshape2)
```

b. Melihat data secara melebar. Misal melihat gaji berdasarkan ID dan tahun kerja

```
gaji_pertahun = dcast(gaji,ID ~ year(ymd(`Mulai Kerja`)), value.var="Gaji")
```

c. Melihat gaji pertahun berdasarkan ID dan Nama Karyawan

```
gaji_pertahun2 = dcast(karyawan_gaji,ID +`Nama Karyawan` ~ year(ymd(`Mulai Kerja`)),
                       value.var="Gaji")
```

d. Membalik posisi data *gaji_pertahun*

```
balik_gajiPertahun = melt(gaji_pertahun, id.vars=c("ID"),
                          variable.name = "Tahun Mulai Kerja",
                          value.name="Gaji")
```

e. Menghilangkan value NA, misal dari data balik_gajiPertahun

```
balik_gajiPertahun_noNA = na.omit(balik_gajiPertahun)
```

9. Mendeteksi Data Missing

Missing data bisa disebabkan karena salah menginputkan data atau karena memang data yang cacat.

a. Mencari data yang missing pada kolom *Akhir Kerja* pada data *gaji.* Kemudian mengubah data missing tersebut dengan value NA

```
gaji[gaji$`Akhir Kerja` >"2100-01-01",]
gaji[gaji$`Akhir Kerja` > "2100-01-01","Akhir Kerja"] = NA
```

b. Mencari data dengan nilai NA, misal kolom *Akhir Kerja* pada data *gaji.* Kemudian menghitung total data dengan nilai NA

```
is.na(gaji$`Akhir Kerja`)
sum(is.na(gaji$`Akhir Kerja`))
```

c. Melihat rasio yang salah pada kolom *Akhir Kerja* pada data *gaji*

```
sum(is.na(gaji$`Akhir Kerja`))/length(gaji$`Akhir Kerja`)
```

10. Mengisi Data Missing

Data yang missing bisa kita isi dengan menggunakan data rata-rata.

Mengisi nilai kosong dari rata-rata nilai. Misal kita isi Gaji yang kosong pada karyawan dengan ID 10001

```
gaji[gaji$ID == 10001,]
mean(gaji$Gaji[gaji$ID == 10001], na.rm=TRUE)
gaji$Gaji[gaji$ID == 10001 & is.na(gaji$Gaji)] -> mean_Gaji
```