webinaR pengantar pengolahan data

June 11, 2020

1 Apa itu R?

R merupakan bahasa pemrograman dan lingkungan perangkat lunak yang digunakan untuk analisis statistik, pemodelan data, visualiasi grafik, dan pelaporan data. R bersifat sumber terbuka dan gratis diunduh oleh siapapun. Webinar ini ditujukan untuk membantu pemula untuk memulai pengolahan data dengan menggunakan R.

2 Instalasi

1. Kunjungi https://docs.conda.io/projects/conda/en/latest/user-guide/install/ untuk instalasi Miniconda versi 3.
2. Ikuti prosedur instalasi (jangan lupa atur PATH -nya).
3. Buku *Command Line Interface* (CLI), jalankan:

```
conda create -n enviRonment r-essential r-base
```

4. Aktifkan lingkungan virtual dengan menjalankan perintah:

```
conda activate enviRonment
```

5. Lakukan instalasi Jupyter Notebook:

```
conda install -c anaconda jupyter
```

6. Lakukan instalasi IRkernel:

```
conda install -c r r-irkernel
```

7. Jalankan perintah:

```
jupyter notebook
```

8. Jika sudah terbuka di *browser* masing - masing sesi R interaktif dapat dimulai.
9. Untuk mengakhiri sesi tekan tombol `<CTRL> + C` di CLI, dan jalankan perintah: `(bash) conda deactivate`

3 PengantaR

3.1 Tipe - tipe data

1. Logical : `TRUE`, `FALSE`
2. Numeric: `1`, `1.2`, `1239`
3. Integer: 2,3,5,6
4. Complex: `4 + 3i`
5. Character: `"Halo"`, `'Halo'`
6. Raw: Bytes

[1]:
```
a <- TRUE
a = 10
a <- 3 + 4i
```

[2]:
```
print(a)
```

```
[1] 3+4i
```

[3]:
```
class(a) # a nerupakan bagian dari kelas ??
```

'complex'

[4]:
```
b <- 5
class(b)
```

'numeric'

[5]:
```
c <- FALSE
class(c)
```

'logical'

[6]:
```
d <- 3L
class(d)
```

'integer'

Secara *default*, seluruh bilangan yang kita tugaskan ke dalam suatu variabel akan berupa tipe data numerik. Jika kita ingin menjadikan bilangan tersebut integer, gunakan `L`.

[7]:
```
teks <- "Halo apa kabar?"
```

[8]:
```
jawaban <- 'Baik baik, Bro!'
```

[9]:
```
teks
```

'Halo apa kabar?'

[10]:
```
class(teks)
```

'character'

[11]:
```
class(jawaban)
```

'character'

[12]:
```
a <- charToRaw('Mohon maaf lahir batin Mas bro!')
a
```

```
 [1] 4d 6f 68 6f 6e 20 6d 61 61 66 20 6c 61 68 69 72 20 62 61 74 69 6e 20 4d 61
[26] 73 20 62 72 6f 21
```

a merupakan representasi bagaimana tipe data character disimpan secara internal di dalam memori komputer.

[13]:
```
class(a)
```

'raw'

3.2 Struktur data

Koleksi dari tipe - tipe data.

- Vectors
- Lists
- Matrix
- Arrays
- Data Frames
- Factors

3.2.1 Pengantar Vektor

Koleksi dari objek - objek dengan tipe data yg sama.

[14]:
```
kendaraan <- c('mobil', 'sepeda motor', 'bus')
kendaraan
```

1. 'mobil' 2. 'sepeda motor' 3. 'bus'

[15]:
```
class(kendaraan)
```

'character'

prioritas tipe data:

characters > numeric > integer > logical

[16]:
```
kendaraan <- c(3, 'sepeda motor', 'bus')
kendaraan # 3 dikonversi ke character
```

1. '3' 2. 'sepeda motor' 3. 'bus'

[17]:
```
num <- c(1L, 1.5, 2)
num
```

1. 1 2. 1.5 3. 2

[18]:
```
class(num)
```

'numeric'

3.2.2 Mendefinisikan vektor

Menggunakan : Sintaks → `awal:akhir`

[19]:
```
x <- 1:7
x
```

1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7

[20]:
```
x <- 2 : -2
x
```

1. 2 2. 1 3. 0 4. -1 5. -2

3.2.3 Menggunakan fungsi `seq()`

Sintaks → `seq(from, to, by, length.out)`

1. `from`: awal
2. `to` : akhir
3. `by` : peningkatan (secara default 1).
4. `length.out`: panjang sekuen.

[21]:
```
seq(5,10)
```

1. 5 2. 6 3. 7 4. 8 5. 9 6. 10

[22]:
```
seq(from = 0, to = 10, by = 2)
```

1. 0 2. 2 3. 4 4. 6 5. 8 6. 10

[23]:
```
seq(from = 1, to = 3, length.out = 5)
# dibagi jadi 5 bag. dgn jarak yg sama
```

1. 1 2. 1.5 3. 2 4. 2.5 5. 3

[24]:
```
seq(from=0, to=20, length.out = 5)
```

1. 0 2. 5 3. 10 4. 15 5. 20

Contoh lainnya:

[25]:
```
x <- 1:8
x
```

1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8

[26]:
```
x <- seq(1,8)
x
```

1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8

[27]:
```
x <- seq(1,8, by=2)
x
```

1. 1 2. 3 3. 5 4. 7

[28]:
```
x <- seq(1,8, length.out = 10)
x
```

1. 1 2. 1.77777777777778 3. 2.55555555555556 4. 3.33333333333333 5. 4.11111111111111 6. 4.88888888888889 7. 5.66666666666667 8. 6.44444444444444 9. 7.22222222222222 10. 8

3.2.4 Operator

[29]:
```
v <- 1:3
w <- 4:6
```

[30]:
```
print(v)
print(w)
```

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

[31]:
```
v - w
```

1. -3 2. -3 3. -3

[32]:
```
v + w
```

1. 5 2. 7 3. 9

[33]:
```
v * w
```

1. 4 2. 10 3. 18

[34]:
```
v / w
```

1. 0.25 2. 0.4 3. 0.5

Operator aritmatika:

- + : penjumlahan

- - : pengurangan
- * : perkalian
- %% : sisa bagi (remainder)
- ^ : pemangkatan

Operator relasi: * < : lebih kecil * > : lebih besar * == : sama dengan (operator kesamaan) * <= : lebih kecil sama dengan * >= : lebih besar sama dengan * != : tidak sama

[35]:
```
v <- c(1,2,3)
w <- 4:6
```

[36]:
```
v > w # bersifat element-wise
```

1. FALSE 2. FALSE 3. FALSE

[37]:
```
v != w
```

1. TRUE 2. TRUE 3. TRUE

3.2.5 Mengakses elemen vektor

[38]:
```
A <- seq(2,10, by=2)
A
```

1. 2 2. 4 3. 6 4. 8 5. 10

[39]:
```
A[c(1,2,3)]
```

1. 2 2. 4 3. 6

[40]:
```
A[1:3]
```

1. 2 2. 4 3. 6

[41]:
```
A[c(-1)] # tanpa indeks 1
```

1. 4 2. 6 3. 8 4. 10

[42]:
```
A[c(-1,-2)] # tanpa indeks 1 dan 2
```

1. 6 2. 8 3. 10

[43]:
```
A[c(-1,-2,-3)] # tanpa indeks 1,2,dan 3
```

1. 8 2. 10

[44]:
```
A[c(TRUE, FALSE, TRUE, TRUE, FALSE)]
```

1. 2 2. 6 3. 8

- Kita dapat menggunakan [] untuk mengakses elemen vektor (*indexing* dimulai dari 1).

- Nilai negatif digunakan untuk membuang elemen pada indeks yang tidak dikehendaki.
- Nilai - nilai Boolean, `TRUE` dan `FALSE` juga dapat digunakan untuk pengindeksan vektor.

3.2.6 Mendefinisikan matriks

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

Sintaks : `matrix(data, nrow, ncol, byrow, dimnames)`

`data`: elemen - elemen di dalam matriks,

`nrow`: jumlah baris,

`ncols`: jumlah kolom,

`byrow`: Jika `TRUE`, maka elemen - elemen matriks akan disusun berdasarkan kolom

`dimnames`: Nama yg ditugaskan pada baris dan kolom.

contoh:

[45]:
```
matrix(1:9, nrow=3, ncol=3)
```

A matrix: 3 × 3 of type int

1	4	7
2	5	8
3	6	9

[46]:
```
matrix(1:9, nrow=3, ncol=3, byrow=TRUE)
```

A matrix: 3 × 3 of type int

1	2	3
4	5	6
7	8	9

[47]:
```
baris <- c('baris1', 'baris2', 'baris3')
kolom <- c('kolom1', 'kolom2', 'kolom3')

matrix(1:9, nrow=3, ncol=3, byrow=FALSE, dimnames = list(baris,kolom))
```

A matrix: 3 × 3 of type int

	kolom1	kolom2	kolom3
baris1	1	4	7
baris2	2	5	8
baris3	3	6	9

3.2.7 Mengakses elemen matriks

[48]:
```
M = matrix(1:9, nrow=3, ncol=3)
M
```

A matrix: 3 × 3 of type int

1	4	7
2	5	8
3	6	9

Sintaks: `M[baris, kolom]`

[49]:
```
M[1,1]
```

1

[50]:
```
M[,1] # kolom 1
```

1. 1 2. 2 3. 3

[51]:
```
bar <- c('b1', 'b2', 'b3')
kol <- c('k1', 'k2', 'k3')

M <- matrix(1:9, nrow=3, ncol=3, dimnames=list(bar,kol))
M
```

A matrix: 3 × 3 of type int

	k1	k2	k3
b1	1	4	7
b2	2	5	8
b3	3	6	9

[52]:
```
M['b1', ] # baris 1
```

k1 1 **k2** 4 **k3** 7

[53]:
```
M[1,]
```

k1 1 **k2** 4 **k3** 7

3.2.8 DataFrame

DataFrame adalah suatu tabel atau struktur berbentuk menyerupai array dua dimensi, di mana setiap kolomnya menyimpan data dari satu variabel dan setiap barisnya menyimpan data untuk suatu "titik" data yang sama dengan variabel yang berbeda - beda. Ibaratnya seperti spreadsheet MS Excel.

Sintaks: `data.frame(#data)`

Contoh:

[54]:
```
nama <- c('Sandy', 'Evelyn', 'Brenda')
umur <- c(43, 23, 45)
data.frame(nama,umur)
```

A data.frame: 3 × 2

nama	umur
<fct>	<dbl>
Sandy	43
Evelyn	23
Brenda	45

Mengakses DataFrame

[55]:
```
nama <- c('Sandy', 'Evelyn', 'Brenda')
umur <- c(43, 23, 45)
gaji <- c(1000000, 9000000, 2000000)

D <- data.frame(nama,umur, gaji)
D
```

A data.frame: 3 × 3

nama	umur	gaji
<fct>	<dbl>	<dbl>
Sandy	43	1e+06
Evelyn	23	9e+06
Brenda	45	2e+06

Cara pertama:

[56]:
```
D$nama # akses kolom nama dari dataframe D
```

1. Sandy 2. Evelyn 3. Brenda

Levels: 1. 'Brenda' 2. 'Evelyn' 3. 'Sandy'

[57]:
```
D$nama[1] # baris pertama dari nama
```

Sandy *Levels*: 1. 'Brenda' 2. 'Evelyn' 3. 'Sandy'

Cara kedua:

[58]:
```
D[1,] # mengakses baris pertama
```

A data.frame: 1 × 3

	nama	umur	gaji
	<fct>	<dbl>	<dbl>
1	Sandy	43	1e+06

[59]:
```
D[c(1,2),] # akses seluruh kolom baris 1 - 2
```

A data.frame: 2 × 3

	nama	umur	gaji
	<fct>	<dbl>	<dbl>
1	Sandy	43	1e+06
2	Evelyn	23	9e+06

[60]:
```
D$gaji <- D$gaji /(1e6) # membagi gaji sebanyak 1jt dan menugaskan ulang ke␣
↪dataframe
D
```

A data.frame: 3 × 3

nama <fct>	umur <dbl>	gaji <dbl>
Sandy	43	1
Evelyn	23	9
Brenda	45	2

[61]:
```
# cara dapat intisari dataframe:

summary(D)
```

```
    nama        umur          gaji
 Brenda:1   Min.   :23   Min.   :1.0
 Evelyn:1   1st Qu.:33   1st Qu.:1.5
 Sandy :1   Median :43   Median :2.0
            Mean   :37   Mean   :4.0
            3rd Qu.:44   3rd Qu.:5.5
            Max.   :45   Max.   :9.0
```

3.2.9 Array

Array merupakan koleksi elemen - elemen multidimensional yang mempunyai tipe data seragam.

Sintaks: `array(data, dim, dimnames)`

Contoh:

[62]:
```
print(array(data=c("who", 'am', 'I'), dim = c(3,3,2)))
# dim => 3 baris, 3 kolom, 2 level (2 matriks)
```

```
, , 1

     [,1]  [,2]  [,3]
[1,] "who" "who" "who"
[2,] "am"  "am"  "am"
[3,] "I"   "I"   "I"

, , 2

     [,1]  [,2]  [,3]
[1,] "who" "who" "who"
[2,] "am"  "am"  "am"
[3,] "I"   "I"   "I"
```

Harus diingat bahwa *array* harus punya tipe data **YANG SAMA**

[63]:
```
bar <- c("B1", "B2", "B3")
kol <- c("K1", "K2", "K3")
mat <- c("M1", "M2")
```

```
print(array(data=c("halo","hai","hoi"), dim=c(3,3,2),␣
 ↪dimnames=list(bar,kol,mat)))
```

```
, , M1

   K1     K2     K3
B1 "halo" "halo" "halo"
B2 "hai"  "hai"  "hai"
B3 "hoi"  "hoi"  "hoi"

, , M2

   K1     K2     K3
B1 "halo" "halo" "halo"
B2 "hai"  "hai"  "hai"
B3 "hoi"  "hoi"  "hoi"
```

Mengakses array

[64]:
```
bar <- c("B1", "B2", "B3")
kol <- c("K1", "K2", "K3")
mat <- c("M1", "M2")

A <- array(data=c("who","am","I"), dim=c(3,3,2), dimnames=list(bar,kol,mat))
print(A)
```

```
, , M1

   K1    K2    K3
B1 "who" "who" "who"
B2 "am"  "am"  "am"
B3 "I"   "I"   "I"

, , M2

   K1    K2    K3
B1 "who" "who" "who"
B2 "am"  "am"  "am"
B3 "I"   "I"   "I"
```

[65]:
```
A[1,1,1] # A[baris, kolom, matriks]
```

'who'

[66]:
```
A[1,1,] # baris 1, kolom 1, semua matriks
```

M1 'who' **M2** 'who'

[67]:
```
A[1,,1] # baris 1, semua kolom, matriks 1
```

K1 'who' **K2** 'who' **K3** 'who'

[68]:
```
A["B1",,] # baris B1, semua kolom, semua matriks
```

A matrix: 3 × 2 of type chr

	M1	M2
K1	who	who
K2	who	who
K3	who	who

3.2.10 Mengenal list

[69]:
```
L <- list("PNP", 1:3, TRUE, matrix(1:6, nrow=2), array(5:10, dim = c(3,3,2)))
print(L)
```

```
[[1]]
[1] "PNP"

[[2]]
[1] 1 2 3

[[3]]
[1] TRUE

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

[[5]]
, , 1

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

, , 2

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

L memuat: karakter, vektor, boolean, matriks, array

List adalah koleksi berbagai macam struktur data

3.2.11 Mengenal factor

- Digunakan untuk menyelesaikan permasalahan pengolahan data *string* yang seringkali bersifat *memory intensive*.
- *Factor* meng-*encode string* menjadi nilai numerik.
- Baik untuk optimisasi algoritma.

[70]:
```
jenisKel <- c("Pria", "Wanita")
factorGen <- factor(jenisKel)
print(factorGen)
```

```
[1] Pria   Wanita
Levels: Pria Wanita
```

[71]:
```
str(factorGen) # ada dua level: Pria dan Wanita
```

```
 Factor w/ 2 levels "Pria","Wanita": 1 2
```

3.3 Operator Penugasan

[72]:
```
# Kedua operator penugasan ini dapat digunakan

x1 = 13
x2 <- 13

x1 == x2
```

TRUE

Namun harus hati - hati juga tidak setiap saat kedua operator ini dapat berlaku dengan sama. Misalnya:

[73]:
```
mean(x = 1:10)
```

5.5

[74]:
```
x # variabel x hanya dapat diakses di dalam fungsi tsb (local var) (x bukan␣
 ↪variabel yg sama)
```

1. 1 2. 1.77777777777778 3. 2.55555555555556 4. 3.33333333333333 5. 4.11111111111111 6. 4.88888888888889 7. 5.66666666666667 8. 6.44444444444444 9. 7.22222222222222 10. 8

[75]:
```
mean(x <- 1:10)
```

5.5

[76]:
```
x # variabel x bersifat global
```

1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8 9. 9 10. 10

Jika kita menggunakan <-, maka nilai tsb akan menjadi variabel global, sementara = hanya bersifat lokal.

4 Eksplorasi data

Kita akan menggunakan Iris dataset dari situs University of California Irvine Machine Learning Repository.

Hal - hal penting yang harus kita catat berkaitan dengan data ini adalah sebagai berikut:

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class: * Iris Setosa * Iris Versicolour * Iris Virginica

4.1 Pengolahan data

4.1.1 Mengimpor data

Cara 1: Menggunakan tautan : "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

[77]:
```
tautan <- "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.
 ↪data"

df <- read.csv(url(tautan), header=FALSE) # header nya tidak mau kita ikut␣
 ↪sertakan
head(df) # 6 baris pertama
```

A data.frame: 6 × 5

	V1 <dbl>	V2 <dbl>	V3 <dbl>	V4 <dbl>	V5 <fct>
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa

Cara 2: Menggunakan pustaka

[78]:
```
library(datasets) # iris termasuk ke dalam datasets pustaka ini
df <- iris # langsung terimpor ke workspace kita
head(df)
```

A data.frame: 6 × 5

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

Cara 3: Mengunduh file secara lokal Pergi ke https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data, unduh filenya, pindahkan ke direktori tempat kalian bekerja (atau di mana pun juga boleh yang penting kalian tahu PATH -nya), jalankan:

[79]:
```
df <- read.csv("./notebooks/iris.data", header=FALSE)
head(df)
```

A data.frame: 6 × 5

	V1 <dbl>	V2 <dbl>	V3 <dbl>	V4 <dbl>	V5 <fct>
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa

4.1.2 Menyiapkan data

Mengetahui dimensi DataFrame

[80]:
```
dim(df) # baris = 150, kolom = 5
```

1. 150 2. 5

Menamakan kolom

[81]:
```
tail(df) # belum mempunyai nama kolom
```

A data.frame: 6 × 5

	V1 <dbl>	V2 <dbl>	V3 <dbl>	V4 <dbl>	V5 <fct>
145	6.7	3.3	5.7	2.5	Iris-virginica
146	6.7	3.0	5.2	2.3	Iris-virginica
147	6.3	2.5	5.0	1.9	Iris-virginica
148	6.5	3.0	5.2	2.0	Iris-virginica
149	6.2	3.4	5.4	2.3	Iris-virginica
150	5.9	3.0	5.1	1.8	Iris-virginica

[82]:
```
colnames(df) = c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width",␣
 ↪"Species")
head(df)
```

A data.frame: 6 × 5

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa

Mengetahui nilai kosong Dataset Iris ini contoh data yang sempurna, karena tidak terdapat nilai kosong, namun sebagian besar data yang kita temui di dunia nyata tidaklah demikian. Akan banyak sekali nilai kosongnya. Ada banyak sekali metode yang digunakan untuk menangani nilai - nilai kosong (mis. mengganti dengan nilai rata - rata, dsb), namun karena fokus kita pada tutorial ini hanya mengangani dataset iris, maka kita tidak akan membahas topik penanganan nilai kosong ini.

Nilai kosong di R ditandai dengan nilai `NA`. Untuk mengetahui keberadaan nilai kosong, gunakan:

[112]:
```
head(is.na(df)) # seluruh entri FALSE: ga ada nilai kosong
```

A matrix: 6 × 5 of type lgl

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE

Tapi hal di atas susah dilihat mata, maka kita gunakan:

[84]:
```
any(is.na(df))
# buat mengecek ke seluruh nilai ada ga yg kosong kalo ada 1 aja maka nilainya␣
 ↪jd TRUE
```

FALSE

Kalau mau tahu berapa jumlah nilai `NA` di dataset, gunakan:

[85]:
```
sum(is.na(df))
```

0

Berikut ini contoh dataframe yang terdapat nilai kosongnya:

[86]:
```
df1 = data.frame(1:4, c(6,7,NA,NA))
df1
```

A data.frame: 4 × 2

X1.4 <int>	c.6..7..NA..NA. <dbl>
1	6
2	7
3	NA
4	NA

[87]:
```
is.na(df1)
```

A matrix: 4 × 2 of type lgl

X1.4	c.6..7..NA..NA.
FALSE	FALSE
FALSE	FALSE
FALSE	TRUE
FALSE	TRUE

[88]:
```
any(is.na(df1))
```

TRUE

[89]:
```
sum(is.na(df1))
```

2

4.2 Visualisasi data

Kita akan menggunakan pustaka graphics untuk visualisasi dasar (bukan menggunakan ggplot2).

[90]:
```
library(graphics)
```

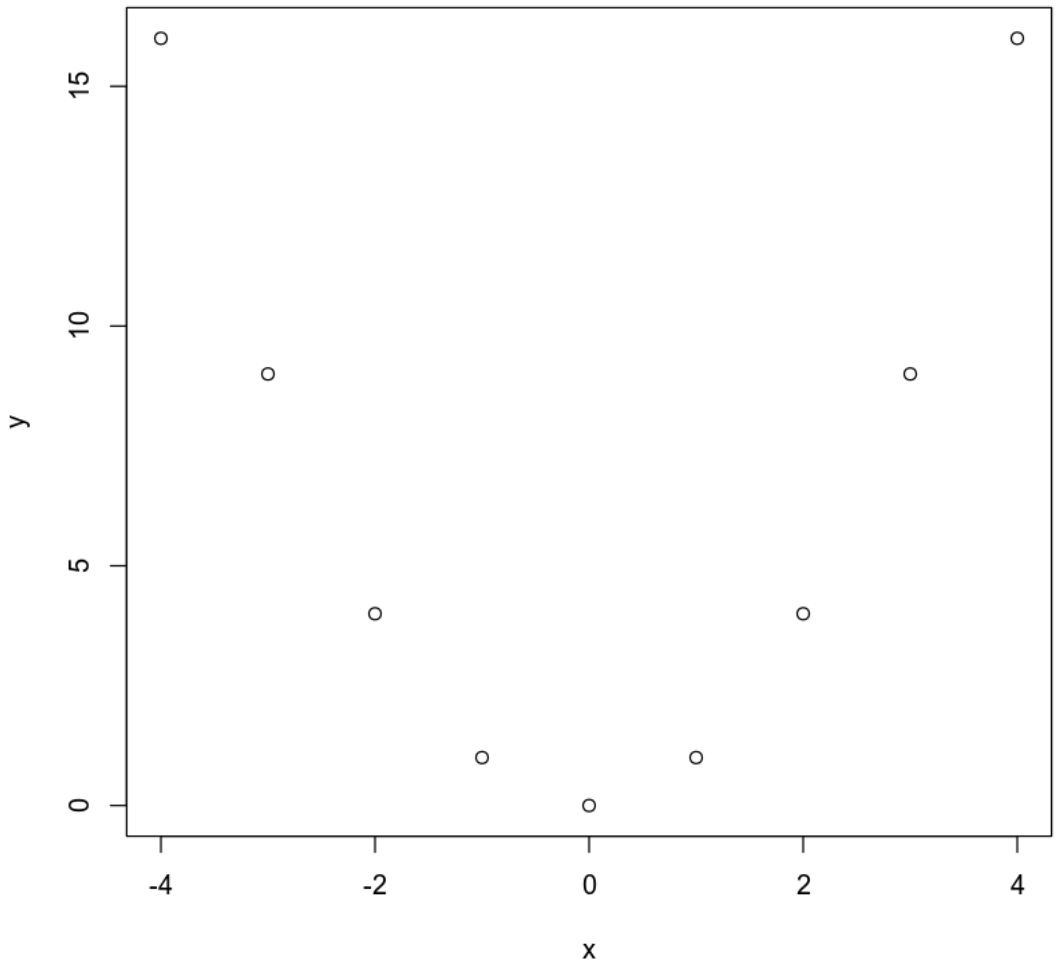
Setiap kali hendak menggunakan pustaka baru, diharapkan kalian mengunjungi situs:

https://www.rdocumentation.org/

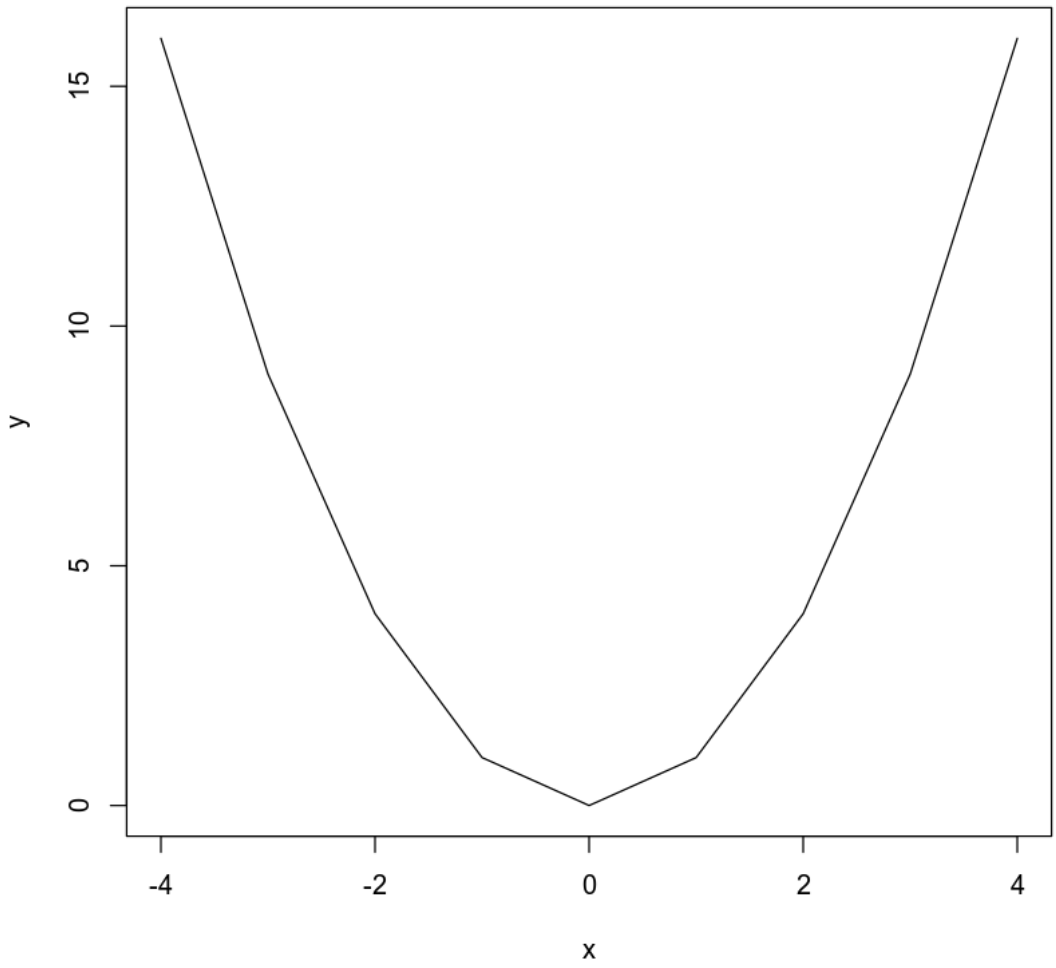
Untuk mencari petunjuk penggunaannya. Karena keunggulan R (bagi kami pribadi dibandingkan Python) adalah kelengkapan dokumentasinya (karena memang diciptakan untuk tujuan akademik???).

4.2.1 Scatterplot

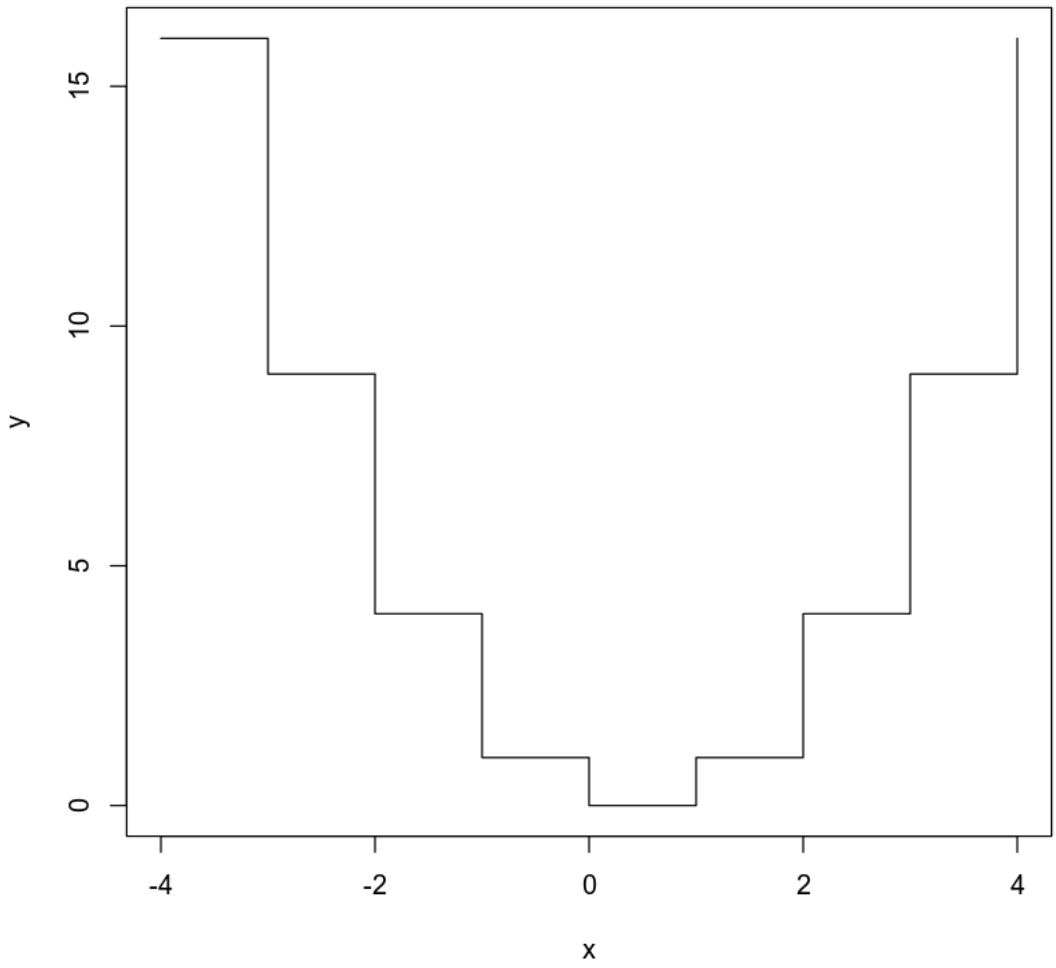
[91]:
```
x <- -4:4
y <- x^2

plot(x,y)
```



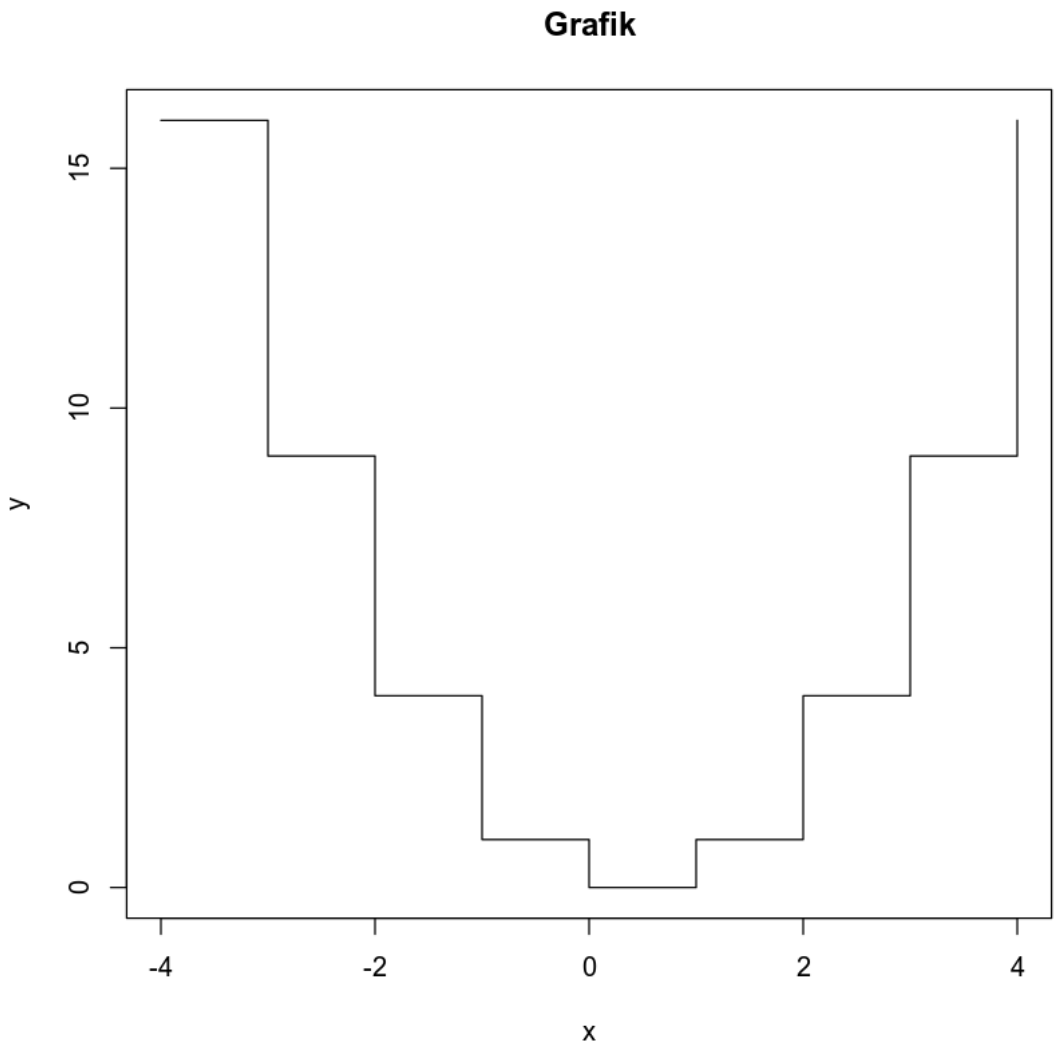
[92]:
```
plot(x,y, type="l")
```



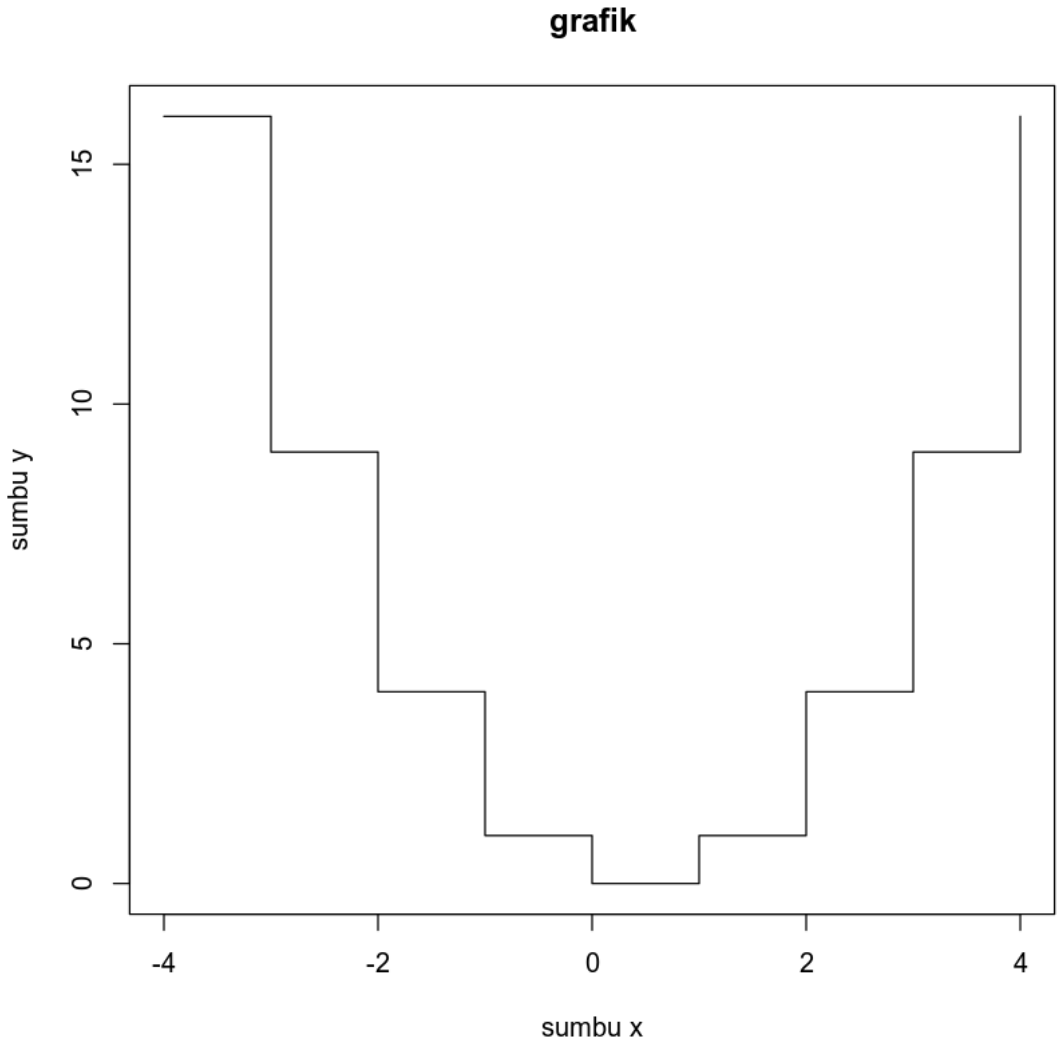
[93]:
```
plot(x,y, type="s")
```



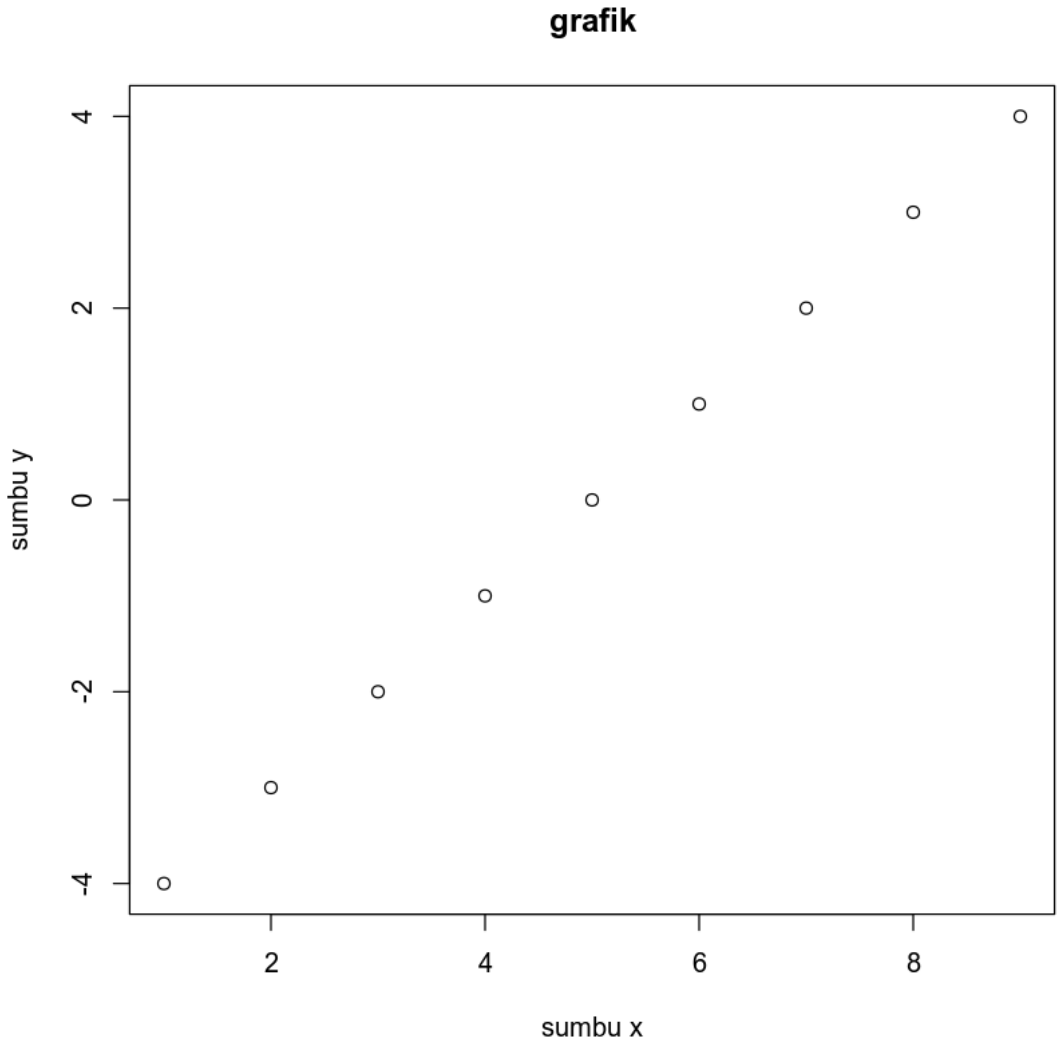
[94]:
```
plot(x,y, type="s", main="Grafik") # nambah judul
```



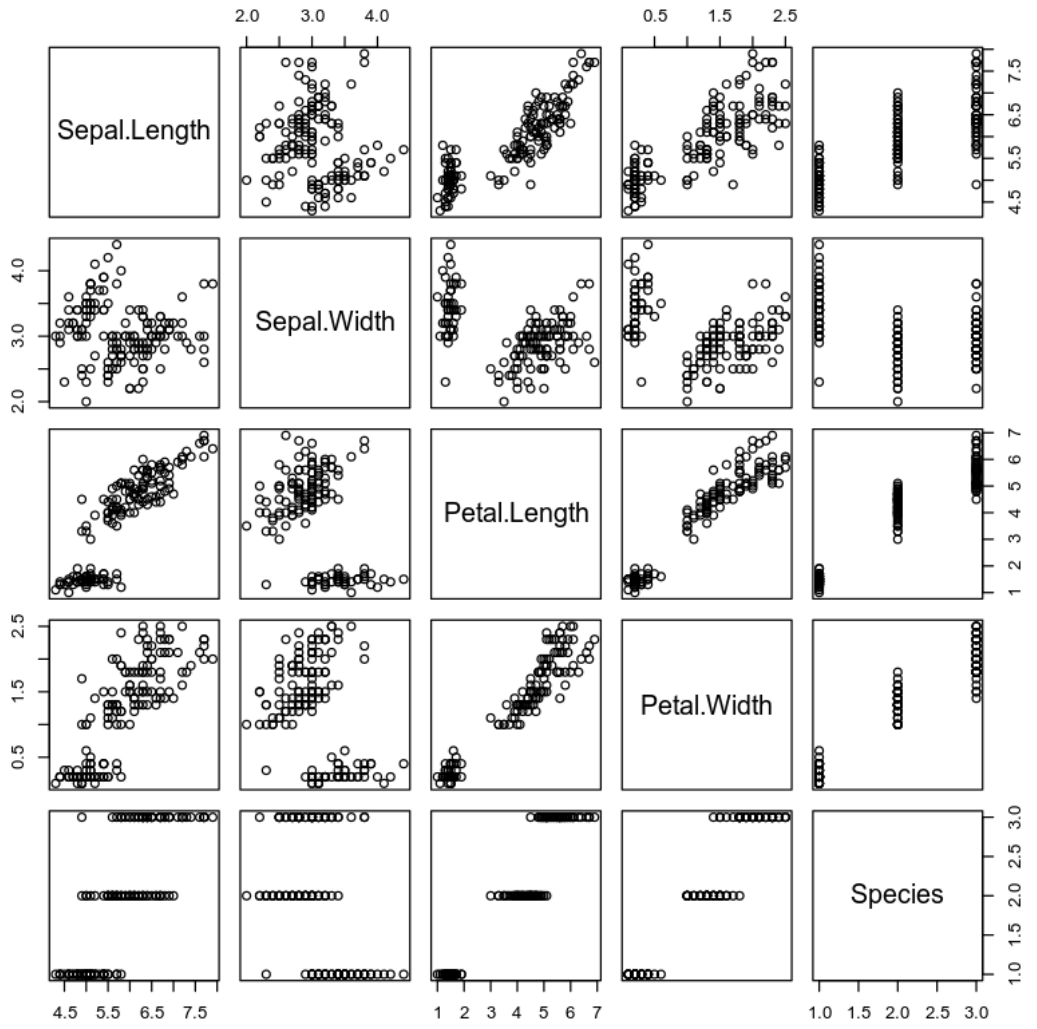
[95]:
```
plot(x,y, type="s", main="grafik", xlab="sumbu x", ylab="sumbu y")
```



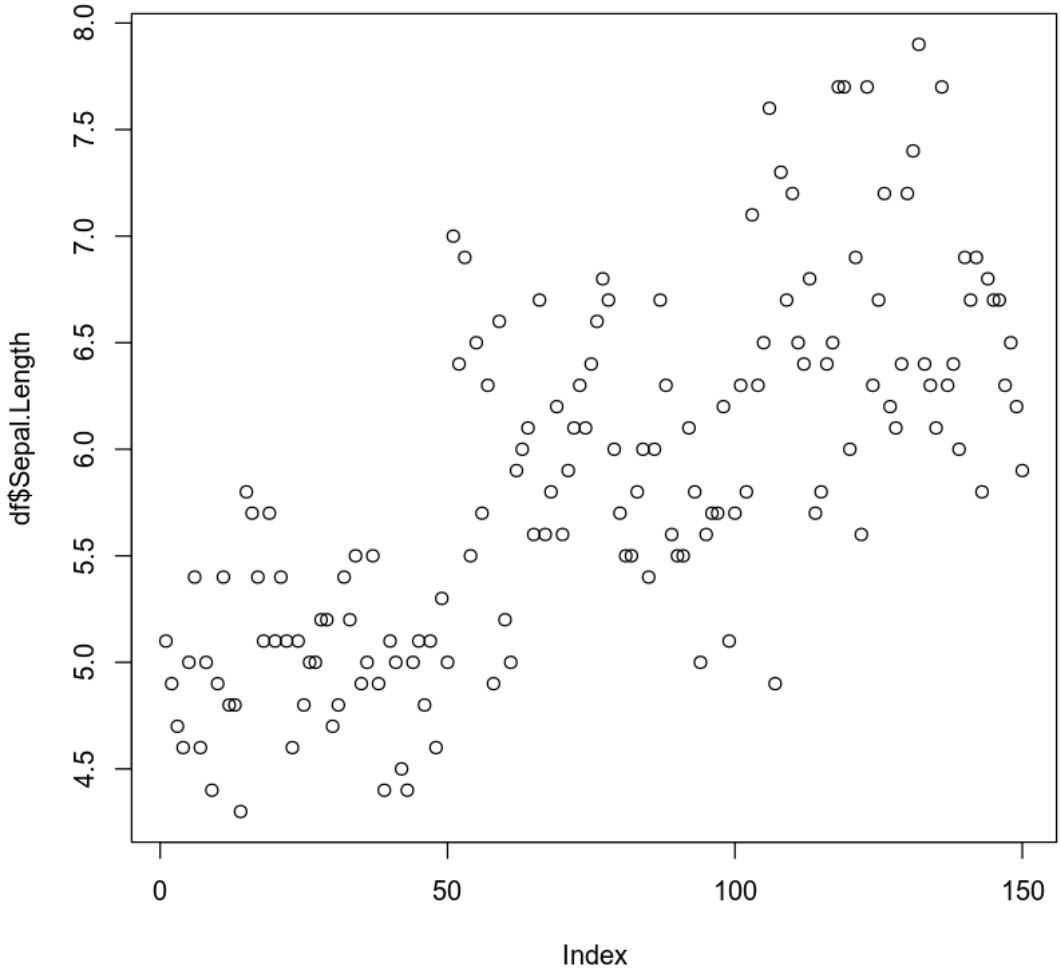
[96]:
```
plot(x, main="grafik", xlab="sumbu x", ylab="sumbu y") # nilai sumbu-y bersifat␣
↪opsional
# default x = y
```



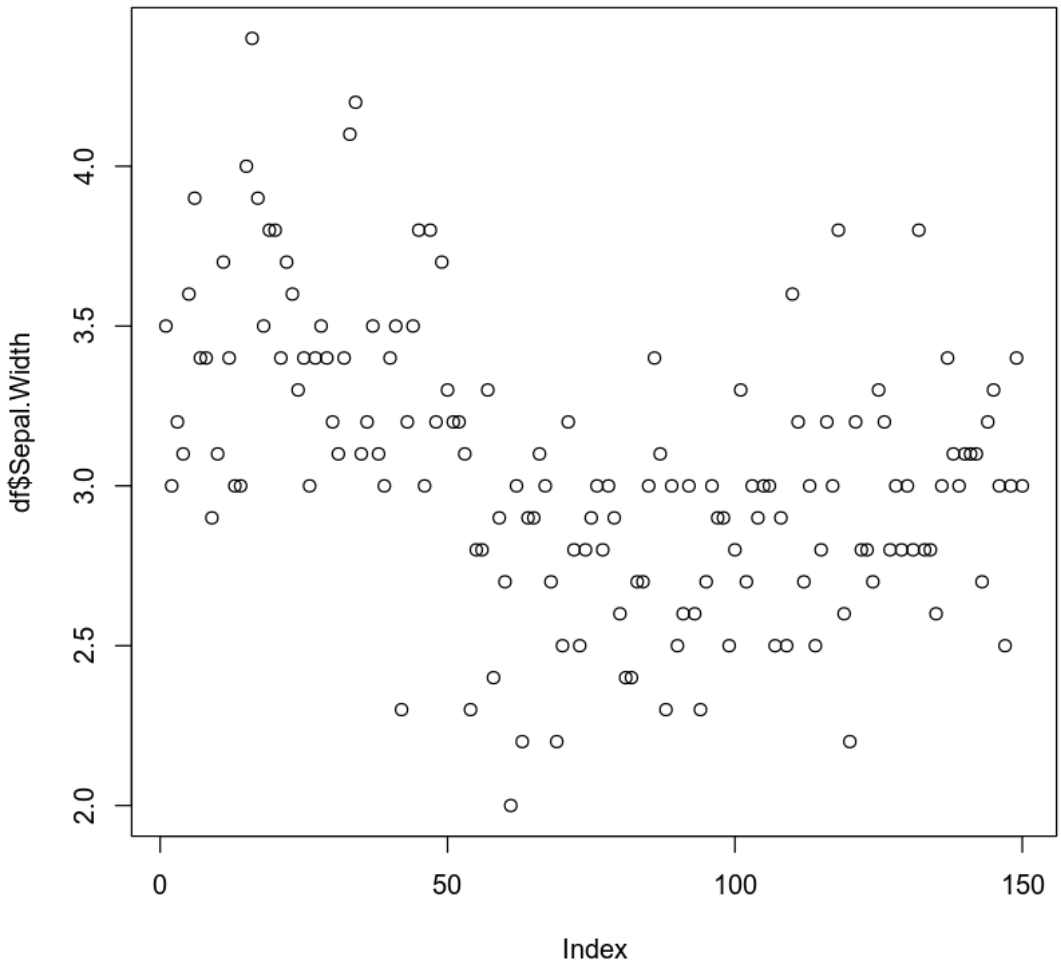
[97]:
```
plot(df) # Plot seluruh dataset Iris
```



[98]:
```
# Scatterplot individual
plot(df$Sepal.Length)
```



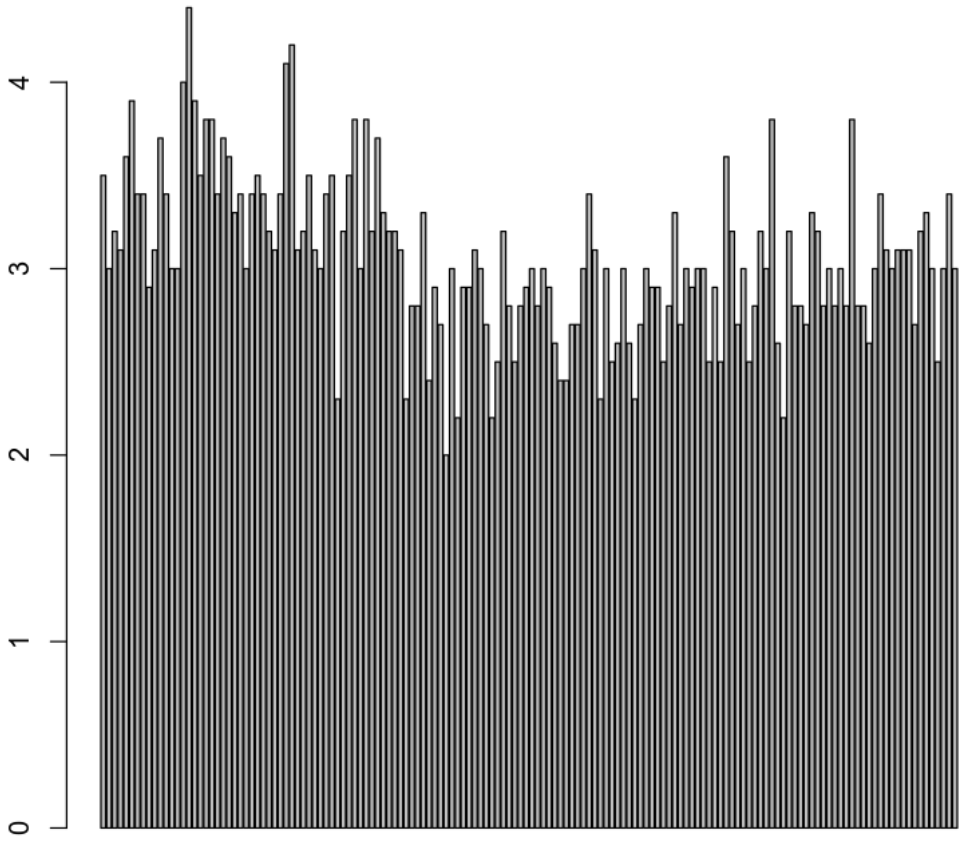
[99]:
```
plot(df$Sepal.Width)
```



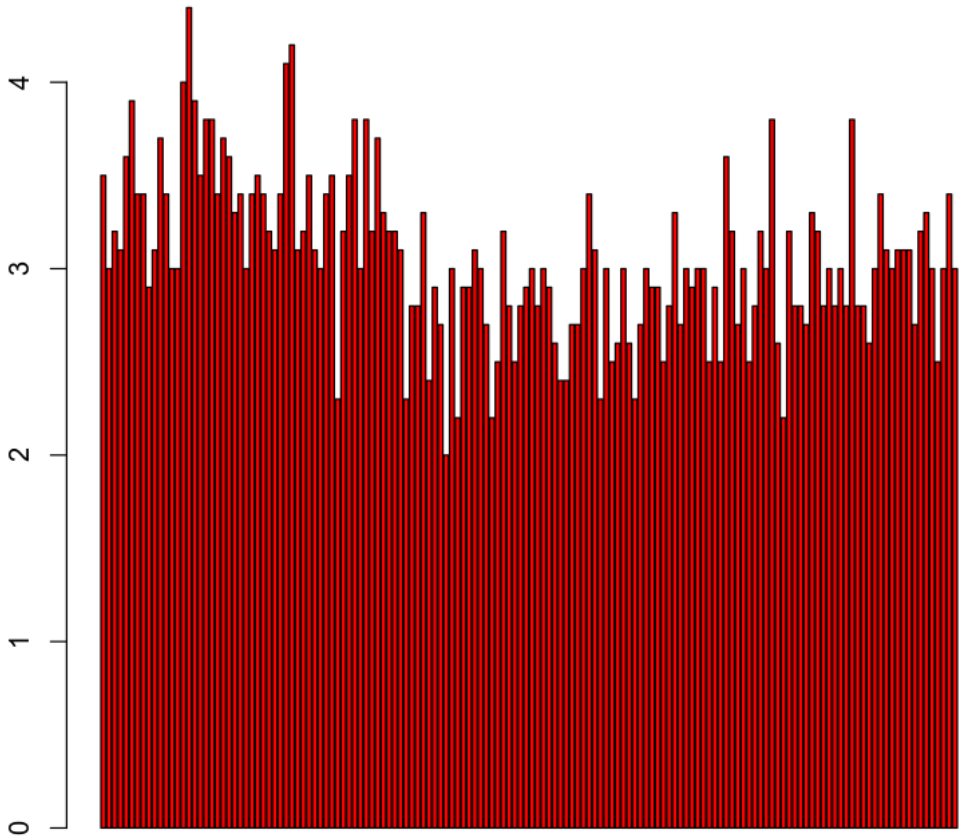
Lebih lanjut, baca: https://www.rdocumentation.org/packages/graphics/versions/3.6.2/topics/plot.

4.2.2 Barplot

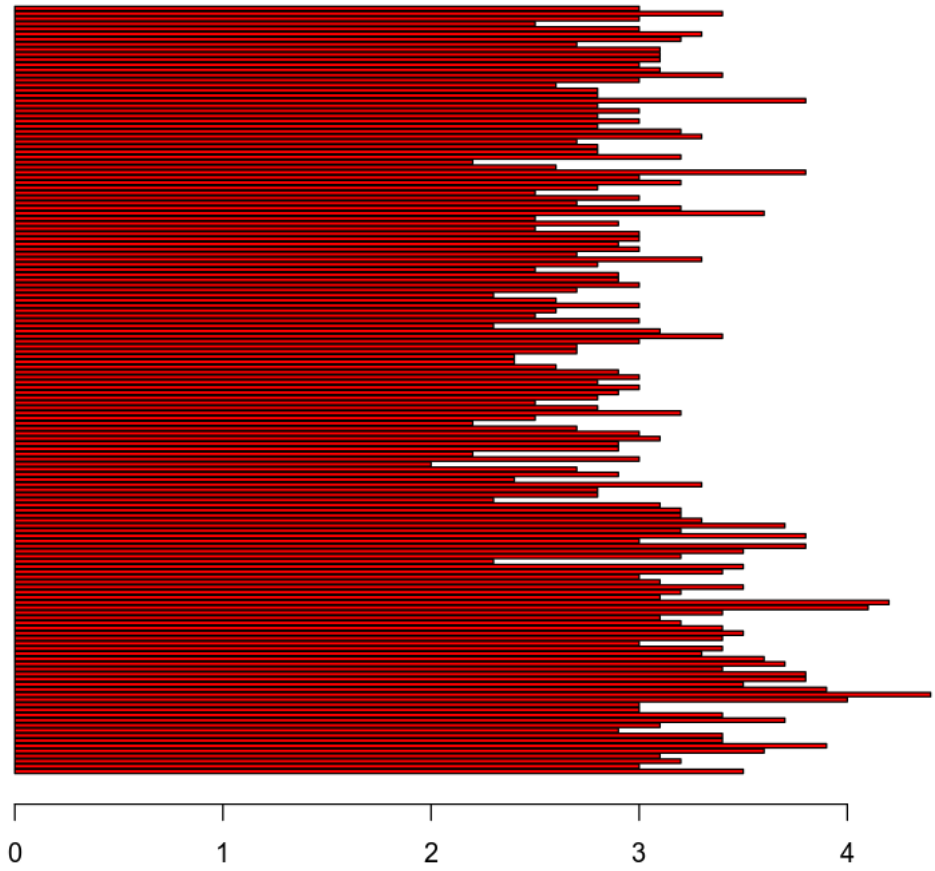
[100]:
```
barplot(df$Sepal.Width)
```



[101]:
```
barplot(df$Sepal.Width, col = 'red')
```



[102]:
```
# jadi horizontal
barplot(df$Sepal.Width, col = 'red', horiz=TRUE)
```

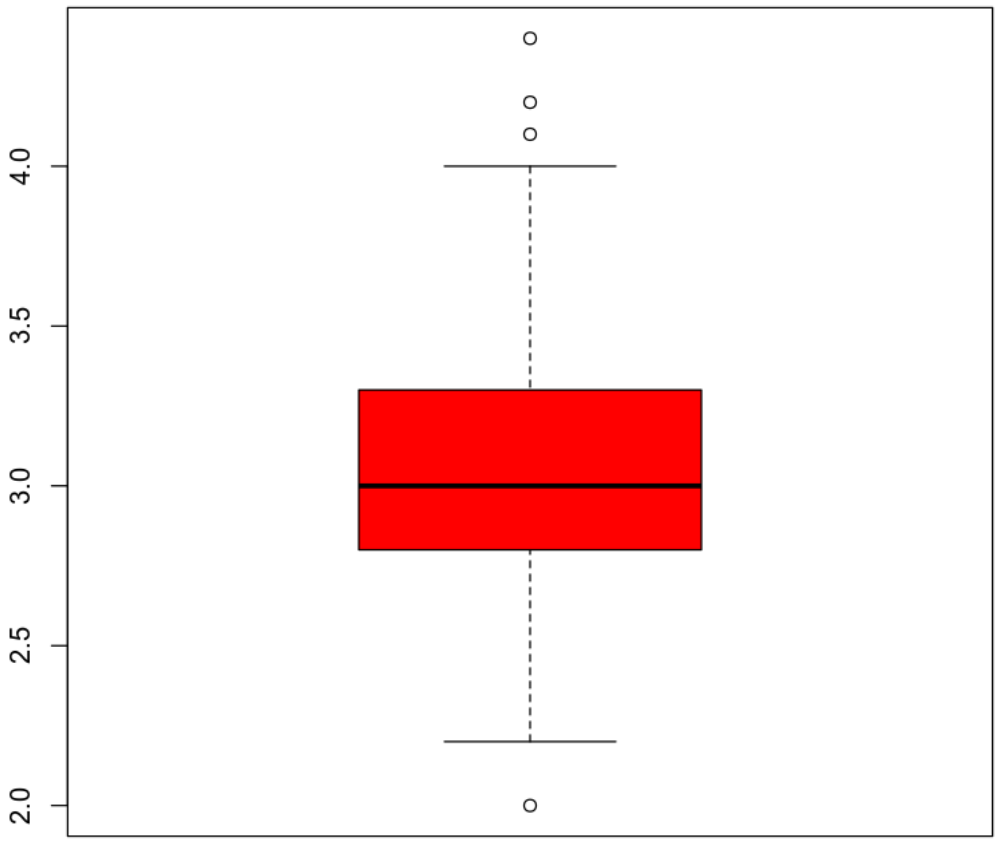


Baca: https://www.rdocumentation.org/packages/graphics/versions/3.6.2/topics/barplot.

4.2.3 Boxplot

Untuk lebih memahami apa itu boxplot, pembaca disarankan untuk mengunjungi: https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51.

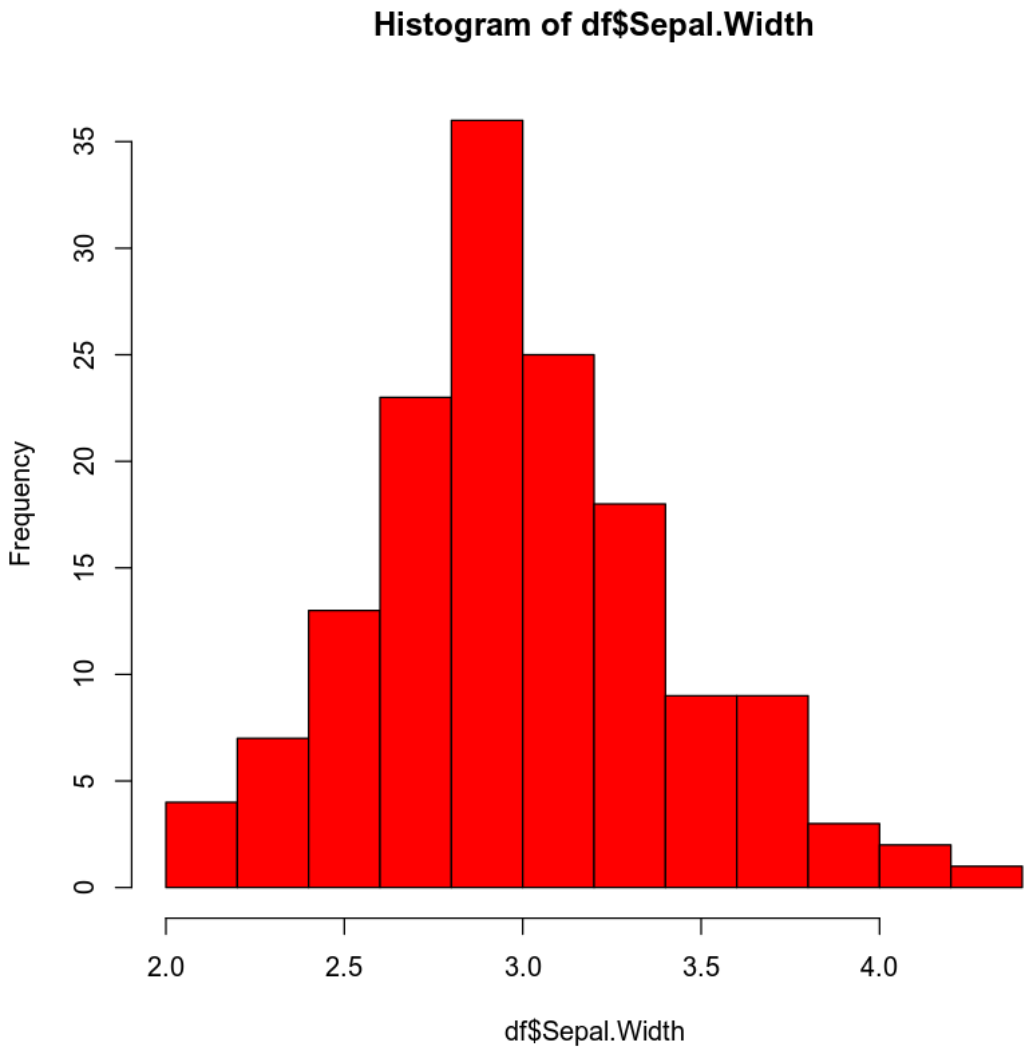
[103]:
```
boxplot(df$Sepal.Width, col = 'red')
```



Baca: https://www.rdocumentation.org/packages/graphics/versions/3.6.2/topics/boxplot.

4.2.4 Histogram

[104]:
```
hist(df$Sepal.Width, col="red")
```



Baca: https://www.rdocumentation.org/packages/graphics/versions/3.6.2/topics/hist.

5 Sekilas tentang dasar pembelajaran mesin

5.1 Pengantar

Pemrograman tradisional:

- Membaca nilai x
- Membaca nilai y
- `jum = x + y`
- `print(jum)`

Jika pada pembelajaran mesin tidak demikian, kita masukan nilai - nilai input dan output yang banyak, sehingga komputer dapat menemukan algoritma pemecahan masalahnya sendiri tanpa harus diperintah.

Pemrograman tradisional: Data + Program → Komputer → Output

Pembelajaran mesin: Data + Output → Komputer → Program

5.2 Apa yang hendak kita lakukan pada sesi ini?

Pada sesi ini kita akan membuat model yang dapat memprediksi lebar sepal (`Sepal.Width`) dari dataset Iris berdasarkan panjang sepal (`Sepal.Length`).

5.3 Yuk kita mulai!

5.4 Memuat dataset

[105]:
```
head(df)
```

A data.frame: 6 × 5

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa

5.4.1 Memisahkan data

[106]:
```
Sepalwidth = df[,"Sepal.Width"]
Sepallength = df[,"Sepal.Length"]
```

[107]:
```
print(Sepalwidth)
```

```
  [1] 3.5 3.0 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 3.7 3.4 3.0 3.0 4.0 4.4 3.9 3.5
 [19] 3.8 3.8 3.4 3.7 3.6 3.3 3.4 3.0 3.4 3.5 3.4 3.2 3.1 3.4 4.1 4.2 3.1 3.2
 [37] 3.5 3.1 3.0 3.4 3.5 2.3 3.2 3.5 3.8 3.0 3.8 3.2 3.7 3.3 3.2 3.2 3.1 2.3
 [55] 2.8 2.8 3.3 2.4 2.9 2.7 2.0 3.0 2.2 2.9 2.9 3.1 3.0 2.7 2.2 2.5 3.2 2.8
 [73] 2.5 2.8 2.9 3.0 2.8 3.0 2.9 2.6 2.4 2.4 2.7 2.7 3.0 3.4 3.1 2.3 3.0 2.5
 [91] 2.6 3.0 2.6 2.3 2.7 3.0 2.9 2.9 2.5 2.8 3.3 2.7 3.0 2.9 3.0 3.0 2.5 2.9
[109] 2.5 3.6 3.2 2.7 3.0 2.5 2.8 3.2 3.0 3.8 2.6 2.2 3.2 2.8 2.8 2.7 3.3 3.2
[127] 2.8 3.0 2.8 3.0 2.8 3.8 2.8 2.8 2.6 3.0 3.4 3.1 3.0 3.1 3.1 3.1 2.7 3.2
[145] 3.3 3.0 2.5 3.0 3.4 3.0
```

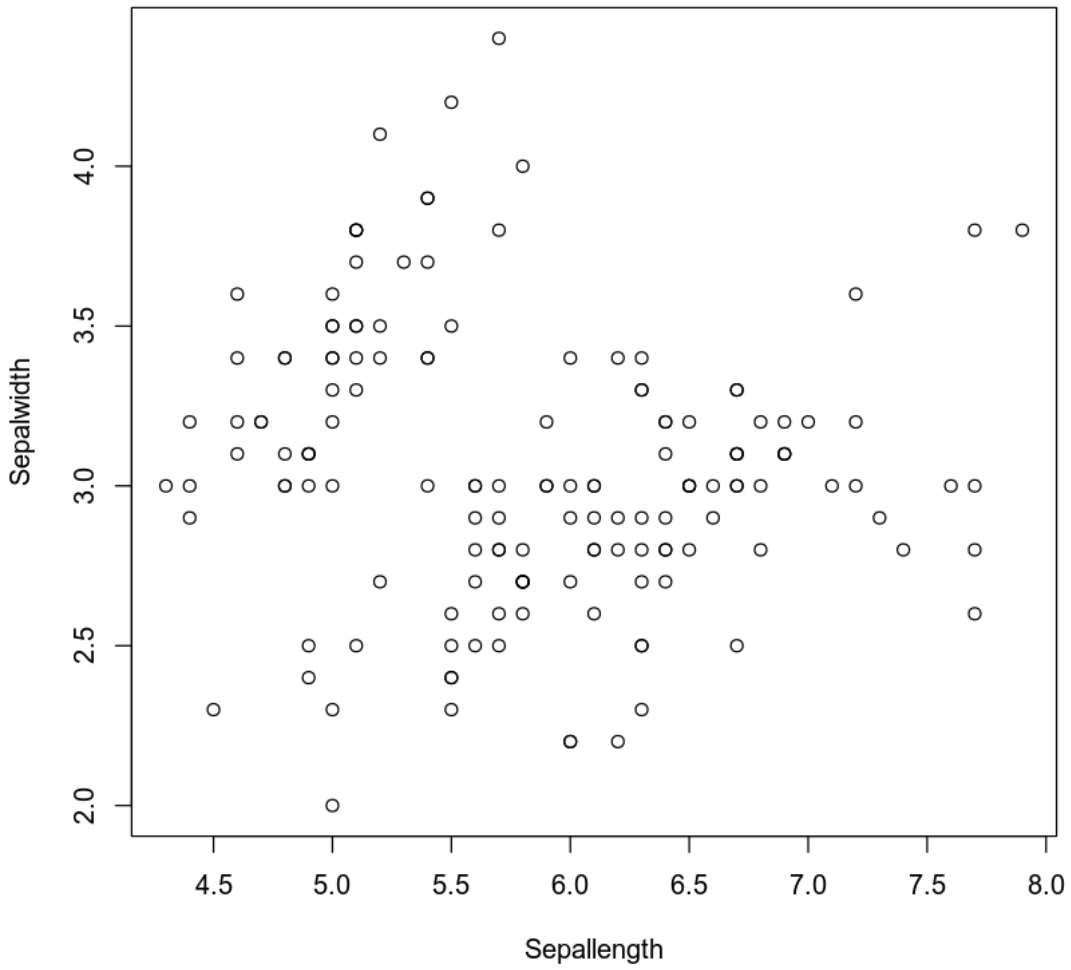
[108]:
```
print(Sepallength)
```

```
  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4 5.1
 [19] 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5 4.9 5.0
 [37] 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0 6.4 6.9 5.5
```

```
 [55] 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8 6.2 5.6 5.9 6.1
 [73] 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4 6.0 6.7 6.3 5.6 5.5
 [91] 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8 7.1 6.3 6.5 7.6 4.9 7.3
[109] 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7 6.0 6.9 5.6 7.7 6.3 6.7 7.2
[127] 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8
[145] 6.7 6.7 6.3 6.5 6.2 5.9
```

5.4.2 Visualisasi data

[109]:
```
plot(Sepallength, Sepalwidth)
```



5.4.3 Membangun model pembelajaran mesin

Karena hanya untuk pengenalan dasar, kita akan menggunakan algoritma pembelajaran mesin paling sederhana, yakni regresi linier.

x	y
2	4
4	8
5	10
6	12
7	14
8	16

Bisa kah kalian lihat pola hubungan antara x dan y?

$$y = 2x$$

Maka dengan demikian kita dapat menggunakan persamaan tersebut untuk memprediksi nilai y untuk setiap x.

Bentuk umum:

$$y = a + bx$$

$$y = 2x \rightarrow A = 0, B = 2$$

Regresi linier digunakan untuk mencari nilai a dan b yang paling mendekati persamaan analitik.

$$Y = a + bX$$

,

di mana:

$$a = \bar{y} - b\bar{x}$$

dan

$$b = \text{Cov}(x, y)/\text{Var}(x)$$

, di mana

$$\text{Cov}(x, y) = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{n}$$

dan

$$\sigma^2 = \frac{\Sigma(x_i - \bar{x})}{N}$$

Untuk penjelasan yang lebih mendetail tentang matematika di baliknya, disarankan untuk melihat video Khan Academy berikut ini:

https://www.youtube.com/watch?v=ualmyZiPs9w

[110]:
```
# regresi linier = lm(y ~ x)
model = lm(Sepalwidth ~ Sepallength)
model
```

```
Call:
lm(formula = Sepalwidth ~ Sepallength)

Coefficients:
(Intercept)  Sepallength
    3.38864     -0.05727
```

$$y = 3.38864 - 0.05727x$$

,

di mana:

y : Sepalwidth

x : Sepallength

5.4.4 Memprediksi berdasarkan model yang telah kita bangun

[111]:
```
predict(model, data.frame(Sepallength=15))
```

1: 2.52961387253063

Mustinya kita harus memisahkan data menjadi *train - test sets*, tetapi karena webinar ini berfokus pada perkenalan R secara sederhana maka hal tersbut tidak akan dibahas di sini.

TERIMAKASIH