



INDONESIA
.NET DEVELOPER COMMUNITY



Seri Belajar
Data Science
Pemrograman R untuk Data Scientist

M Reza Faisal

Kata Pengantar

Puji dan syukur diucapkan kepada Allah SWT atas selesainya ebook sederhana yang berjudul Seri Belajar Data Science: Pemrograman R untuk Data Scientist.

Akhir kata, selamat membaca dan semoga ebook ini bermanfaat bagi para pemula dalam pengolahan data statistik dan machine learning. Kritik dan saran akan sangat berarti dan dapat ditujukan via email.

Banjarmasin, February 2017

M Reza Faisal

(reza.faisal@gmail.com)

Daftar Isi

<i>Kata Pengantar</i>	I
<i>Daftar Isi</i>	II
<i>Daftar Gambar</i>	VIII
1 Pendahuluan	11
Data Science	11
Apakah ini adalah A atau B?	11
Apakah ini aneh?	12
Berapakah ini nanti?	12
Bagaimana hal ini diorganisir?	13
Apa yang harus dilakukan selanjutnya?	14
Definisi Data	14
R Environment	15
Installasi	15
Tool Pemrograman	20
2 Dasar-Dasar Bahasa Pemrograman R	30
Pendahuluan	30
Working Directory	30
R Session	31
Variable	31
Function	33
Tipe Data	34
Skalar	34
Date	34
Time	35
Vector	35
Factor	36
Matrix	37
Array	39
Data Frame	40
List	41
Data Pada RStudio	42

Percabangan	43
Operator	43
Statement if	43
Statement ifelse.....	44
Statement switch	45
Pengulangan.....	45
Statement for.....	45
Statement while.....	46
Statement repeat.....	47
Statement next	47
Statement break.....	47
Penanganan Kesalahan	48
Masih kosong.....	49
Komentar	49
Debug	49
Profilling.....	51
3 Pemrograman Berbasis Obyek	54
Class.....	54
Obyek	55
Inheritance.....	55
Method & Fungsi Generic	56
4 Fungsi Dasar R	58
Dokumen Bantuan.....	58
Package.....	58
Installasi Package	59
Memuat Package	59
GitHub.....	59
Dataset.....	60
Menulis Data Ke File.....	62
write.csv().....	62
write.table().....	63
Membaca File Text.....	63
read.csv()	63
read.delim().....	63

read.table()	64
Membaca File Excel.....	64
Akses Database.....	64
Akses Database MySQL	64
Akses Database SQL Server.....	65
Dimensi Data	66
nrow()	66
ncol()	66
colnames()	66
rownames()	67
Menampilkan Data	68
head()	68
tail()	68
sample().....	69
unique()	69
View()	69
obyek\$rowName	70
Memfilter Data	70
obyek[,col1:col2].....	70
obyek[row1:row2,].....	71
obyek[row1:row2, col1:col2].....	71
obyek[which(),]	71
Sorting Data	71
order.....	71
sort.....	72
Menggabung Data.....	72
rbind()	72
cbind().....	73
rbind.data.frame()	74
cbind.data.frame()	74
paste()	74
paste0()	75
Explorasi Data.....	75
dim().....	75
names()	76

str()	76
table()	76
summary().....	76
Pengulangan.....	77
lapply()	77
sapply().....	78
colSums().....	79
colMeans().....	79
rowSums().....	79
rowMeans()	79
Membuat Data Simulasi	79
rnorm()	80
dnorm().....	80
pnorm().....	80
qnorm()	80
set.seed().....	80
Mengelola Obyek.....	81
assign().....	81
get()	81
ls().....	82
exists().....	82
rm().....	82
save()	83
load()	83
Memeriksa Obyek.....	84
class()	84
is.null().....	84
is.na()	84
is.numeric()	85
Konversi Tipe Data	86
as.character()	86
as.numeric()	86
as.data.frame()	86
5 Visualisasi Data	89
Grafik	90

plot().....	90
scatterplot3()	91
plot3d()	92
hist()	93
density().....	94
boxplot()	94
featurePlot()	95
pie()	96
barplot().....	97
par().....	98
dotchart().....	99
missmap().....	100
corrplot().....	101
pairs()	102
Simbol	103
Export Gambar	104
Visualisasi Dengan Package ggplot.....	106
Masih kosong.....	106
Animasi	106
ani.options()	106
ani.pause().....	106
saveHTML().....	107
saveGIF()	108
Referensi	109
6 Contoh Kasus	110
Pengolahan Data Awal.....	110
Persiapan Data Cross Validation	114
7 Komputasi Paralel.....	118
Pendahuluan	118
Persiapan.....	120
Fungsi Dasar Komputasi Paralel.....	120
detectCore().....	120
makeCluster()	120
stopCluster()	121

parLapply()	121
parSapply()	122
registerDoParallel()	122
Contoh Kasus	124
Referensi	124
8 Reduksi Dimensi Data	125
Principal Component Analysis (PCA)	125
t-Distributed Stochastic Neighbor Embedding (t-SNE)	126
Multidimensional Scaling	126
Referensi	126
9 Referensi	127
10 Penutup	128

Daftar Gambar

Gambar 1. Apakah ini A atau B?	11
Gambar 2. Apakah ini aneh?	12
Gambar 3. Berapa ini nanti?	13
Gambar 4. Bagaimana hal ini diorganisir?	13
Gambar 5. Apa yang harus dilakukan selanjutnya?	14
Gambar 6. Data dalam bentuk tabel.	14
Gambar 7. Window Select Setup Language.....	15
Gambar 8. Window Setup - Welcome.....	16
Gambar 9. Window Setup - Informasi lisensi.....	16
Gambar 10. Windows Setup - Lokasi folder installasi.....	17
Gambar 11. Window Setup - Pemilihan komponen.	17
Gambar 12. Window Setup - Pemilihan opsi startup.	18
Gambar 13. Window Setup - Memilih lokasi folder Start Menu.....	18
Gambar 14. Window Setup - Pemilihan task tambahan.	19
Gambar 15. Windows Setup - Proses installasi.	19
Gambar 16. Window informasi akhir proses installasi.	20
Gambar 17. RGui.	20
Gambar 18. R Console.	21
Gambar 19. Window R Editor.	22
Gambar 20. Menyimpan file script R.	22
Gambar 21. Eksekusi file script R.	23
Gambar 22. Window R Graphics.	23
Gambar 23. Window RStudio Setup - Welcome.	24
Gambar 24. Window RStudio Setup - Lokasi installasi.....	24
Gambar 25. Window RStudio Setup - Memilih lokasi folder Start Menu.	25
Gambar 26. Window RStudio Setup - proses installasi.	25
Gambar 27. Window RStudio Setup - detail proses installasi.	26
Gambar 28. Window RStudio Setup - Proses installasi selesai.	26
Gambar 29. Antarmuka RStudio.	27
Gambar 30. Fitur auto complete kode.	27
Gambar 31. Window Plots untuk menampilkan grafik.	28
Gambar 32. Visual Studio 2017 – R language support.	28

Gambar 33. Visual Studio 2017 – R Project	29
Gambar 34. Visual Studio 2017 – R editor.....	29
Gambar 35. Tiga R session aktif.....	31
Gambar 36. RStudio - Environment.....	32
Gambar 37. RStudio - History.....	32
Gambar 38. RStudio - Editor & Files.....	33
Gambar 39. RStudio - Data.....	42
Gambar 40. RStudio – Memulai proses debug.....	50
Gambar 41. RStudio – Melanjutkan proses debug dengan mengklik tombol Next.....	50
Gambar 42. RStudio – Profilling dimulai.	52
Gambar 43. RStudio – Dokumen bantuan.	58
Gambar 44. RStudio – Installasi package.	59
Gambar 45. RStudio – Daftar dataset.....	61
Gambar 46. View(iris_sample).....	67
Gambar 47. RStudio – View().....	70
Gambar 48. Tipe presentasi (sumber: Dr. Andrew Abela, http://extremepresentation.typepad.com).....	89
Gambar 49. Grafik plot(iris[1], iris[2])	90
Gambar 50. Grafik plot(cars).....	91
Gambar 51. Grafik plot(iris)	91
Gambar 52. Grafik scatterplot3d().....	92
Gambar 53. Grafik plot3d().....	93
Gambar 54. Grafik hist(mtcars[,1]).....	94
Gambar 55. Grafik density.	94
Gambar 56. Grafik fungsi boxplot().....	95
Gambar 57. featurePlot() tipe density untuk dataset iris.	95
Gambar 58. featurePlot() tipe box untuk dataset iris.....	96
Gambar 59. Grafik pie()	96
Gambar 60. Grafik pie dataset iris.....	97
Gambar 61. Grafik barplot vertical.....	97
Gambar 62. Grafik barplot horizontal.....	98
Gambar 63. Fungsi par() untuk membuat 2 grafik dalam 1 kolom.....	98
Gambar 64. Fungsi par() untuk membuat 3 grafik dalam 1 kolom.....	99
Gambar 65. Fungsi par() untuk membuat 4 grafik dalam 2 baris dan 2 kolom.....	99
Gambar 66. Fungsi dotchart() untuk membuat Cleveland dot plot	100

Gambar 67. missmap() dataset iris.....	100
Gambar 68. Missmap dataset Soybean.....	101
Gambar 69. corrplot() dataset iris.....	102
Gambar 70. pairs() dataset iris	102
Gambar 71. pair() dataset iris berdasarkan class.....	103
Gambar 72. Simbol grafik.....	103
Gambar 73. Halaman pertama file kumpulan_chart.pdf.....	104
Gambar 74. Halaman kedua file kumpulan_chart.pdf.....	105
Gambar 75. Halaman ketiga file kumpulan_chart.pdf.....	105
Gambar 76. Fungsi ani.options() & ani.pause().....	107
Gambar 77. Fungi saveHTML().....	108
Gambar 78. Pengolahan data awal – result_all.....	114
Gambar 79. Persiapan data cross validation – dataset iris.....	115
Gambar 80. Persiapan data cross validation – proses I	115
Gambar 81. Persiapan data cross validation – proses II.....	115
Gambar 82. Persiapan data cross validation – proses III	115
Gambar 83. Persiapan data cross validation – proses IV	115
Gambar 84. Persiapan data cross validation – proses V.....	116
Gambar 85. Task Manager pada MS Windows.....	118
Gambar 86. Proses pada Task Manager.	119
Gambar 87. Proses saat menjalankan komputasi parallel.....	122
Gambar 88. Keluaran proses komputasi parallel.....	124

1 Pendahuluan

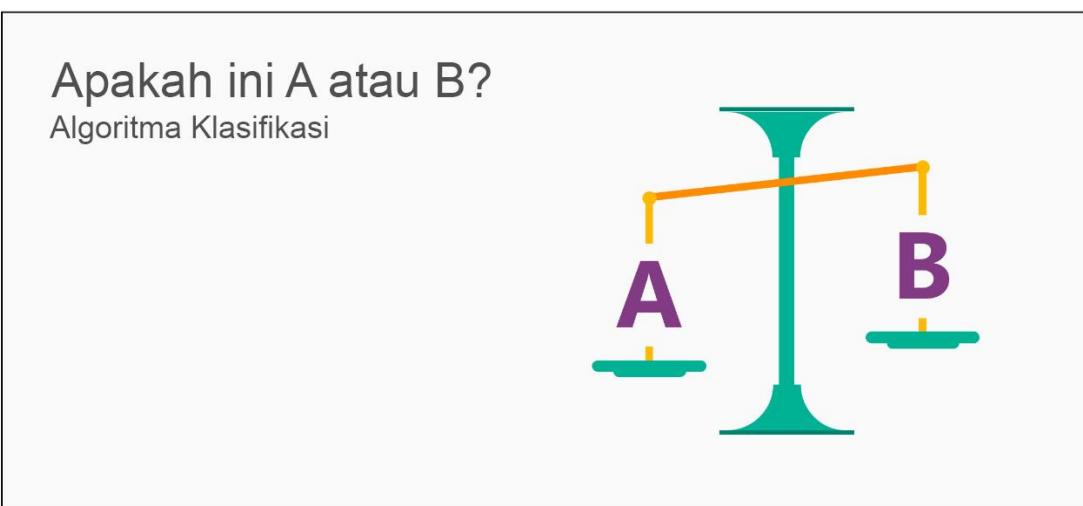
Data Science

Apa yang dipelajari atau dilakukan di bidang Data Science bukanlah bidang baru. Untuk mengetahui apa itu data science dan apa saja yang dipelajari di bidang ini, maka dapat dijawab dengan 5 pertanyaan berikut ini, yaitu:

- Apakah ini adalah A atau B?
- Apakah ini aneh?
- Berapakah nilai ini nanti?
- Bagaimana hal ini diorganisir?
- Apa yang harus dilakukan selanjutnya?

Apakah ini adalah A atau B?

Algoritma klasifikasi bertujuan untuk menentukan kategori atau class dari suatu obyek.



Gambar 1. Apakah ini A atau B?

Sebagai contoh:

- Untuk obyek manusia, maka algoritma klasifikasi dapat digunakan untuk menentukan apakah manusia ini adalah pria atau wanita.
- Untuk kasus kesehatan, maka algoritma klasifikasi dapat digunakan untuk menentukan apakah pasien ini sehat atau sakit.

Contoh kasus di atas adalah disebut klasifikasi 2 class (binary classification), dimana hanya terdapat dua kategori atau class. Contoh lain adalah sebagai berikut:

- Untuk obyek pelajar, maka algoritma klasifikasi dapat digunakan untuk menentukan apakah pelajar ini adalah sedang sekolah di SD, SMP, SMU atau belum sekolah.

- Untuk kasus kesehatan, maka algoritma klasifikasi dapat menentukan apakah pasien bergolongan darah A, B, AB atau O.

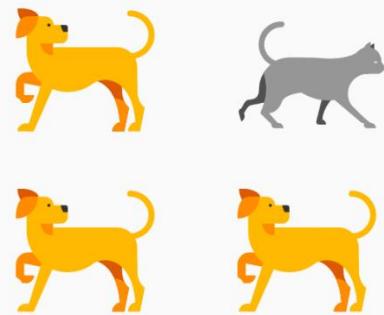
Kasus di atas melibatkan lebih dari 2 kategori atau class. Contoh kasus di atas disebut klasifikasi multiclass (multiclass classification).

Apakah ini aneh?

Algoritma deteksi anomali (anomaly detection) bertujuan untuk menemukan keanehan atau anomali dari sekumpulan data. Deteksi anomali dapat mengetahui peristiwa atau perilaku yang tidak terduga atau tidak biasa.

Apakah ini aneh?

Algoritma Deteksi Anomali



Gambar 2. Apakah ini aneh?

Sebagai contoh:

- Untuk kasus transaksi kartu kredit, algoritma deteksi anomali dapat digunakan untuk menganalisis pola transaksi sehingga dapat diketahui transaksi mana yang aneh yang dapat digolongkan penipuan (fraud).

Berapakah ini nanti?

Algoritma regresi bertujuan untuk melakukan prediksi berupa angka (numerik). Sebagai contoh:

- Untuk kasus prediksi cuaca, algoritma regresi dapat digunakan untuk menentukan suhu atau temperatur pada minggu depan.
- Untuk kasus prediksi keuangan, maka algoritma regresi dapat digunakan untuk menentukan berapa keuntungan perusahaan tahun depan.

Berapa ini nanti?

Algoritma Regresi



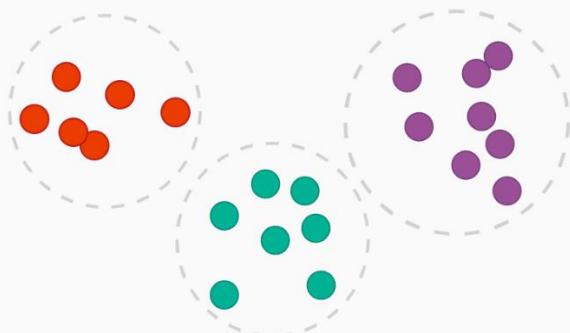
Gambar 3. Berapa ini nanti?

Bagaimana hal ini diorganisir?

Kadang kita menemukan data yang tidak diketahui strukturnya dan bagaimana data tersebut diorganisir. Dan kita tidak memiliki contoh data yang telah memiliki kategori.

Bagaimana hal ini diorganisir?

Algoritma Clustering



Gambar 4. Bagaimana hal ini diorganisir?

Untuk pertanyaan ini digunakan algoritma clustering. Data yang digunakan pada kasus ini tidak memiliki kategori atau class seperti data yang digunakan pada kasus klasifikasi. Algoritma clustering akan mengelompokkan obyek-obyek yang memiliki kemiripan seperti yang terlihat pada contoh di atas.

Contoh implementasinya adalah sebagai berikut:

- Pada kasus biology, data untaian DNA dari hewan dan manusia, algoritma clustering dapat mengelompokkan hewan dan manusia berdasarkan kemiripan data. Hasilnya dari dapat dilihat jika orang hutan, simpanse dan gorila memiliki kemiripan.

Apa yang harus dilakukan selanjutnya?

Algoritma reinforcement learning digunakan untuk menentukan keputusan langkah atau aksi selanjutnya. Algoritma ini berdasarkan bagaimana otak tikus dan manusia menanggapi hukuman (punishment) dan imbalan (reward). Algoritma ini belajar dari hasil, dan memutuskan tindakan selanjutnya.



Gambar 5. Apa yang harus dilakukan selanjutnya?

Implementasi algoritma ini dapat berguna untuk otomasi sistem untuk menentukan keputusan mandiri tanpa bimbingan manusia. Sebagai contoh:

- Untuk kasus mobil yang dapat berjalan sendiri di jalan raya, ketika melihat lampu kuning maka apa yang harus dilakukan, apakah menghentikan mobil atau mempercepat jalannya mobil?

Definisi Data

Secara sederhana data adalah kumpulan obyek atau sample. Dan setiap obyek atau sample tersebut memiliki atribut atau feature. Secara sederhana data dapat digambarkan sebagai tabel di bawah ini.

	Feature 1	Feature 2	...	Feature n
Obyek 1				
Obyek 2				
...				
Obyek n				

Gambar 6. Data dalam bentuk tabel.

Bentuk data dengan struktur sederhana seperti gambar di atas umum digunakan oleh algoritma-algoritma yang telah di bahas di atas. Tetapi seringkali data yang didapat pada kasus nyata belum dalam struktur sederhana atau belum bersih. Sering kali data masih mengandung nilai-nilai yang tidak diperlukan, atau terdapat nilai yang hilang. Sehingga data belum siap untuk diproses oleh algoritma-algoritma di atas.

Oleh karena itu perlu ada tahap pre-processing. Tahapan ini merupakan tahap terpenting dan dapat memakan waktu yang lama.

R Environment

R bukan saja bahasa pemrograman tetapi juga lingkungan/environment untuk komputasi statistik dan grafik. R merupakan project GNU yang dikembangkan oleh Bell Laboratories (sebelumnya AT&T, sekarang Lucent Technologies) oleh John Chamber dan teman-temannya.

R menyediakan berbagai macam tool statistik dari linier dan memodelan non linier, uji statistik klasik, analisis time-series, klasifikasi, clustering dan lain-lain. R juga menyediakan tool teknik grafis yang bertujuan untuk menampilkan data yang telah diolah secara visual dalam bentuk grafik.

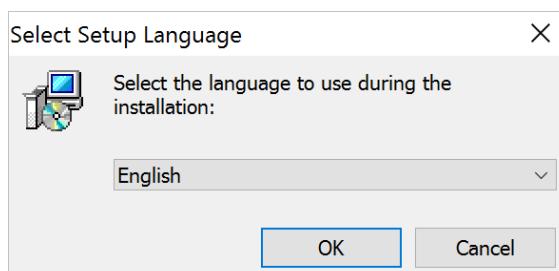
R merupakan project open-source yang memungkinkan banyak pihak untuk memberikan kontribusi dalam proses pengembangan.

Installasi

Installer atau source code R tersedia dalam beberapa platform yaitu Windows, Mac OS X dan Linux. Pada ebook ini digunakan installer yang diunduh dari link berikut ini <https://cran.r-project.org/bin/windows/base/>.

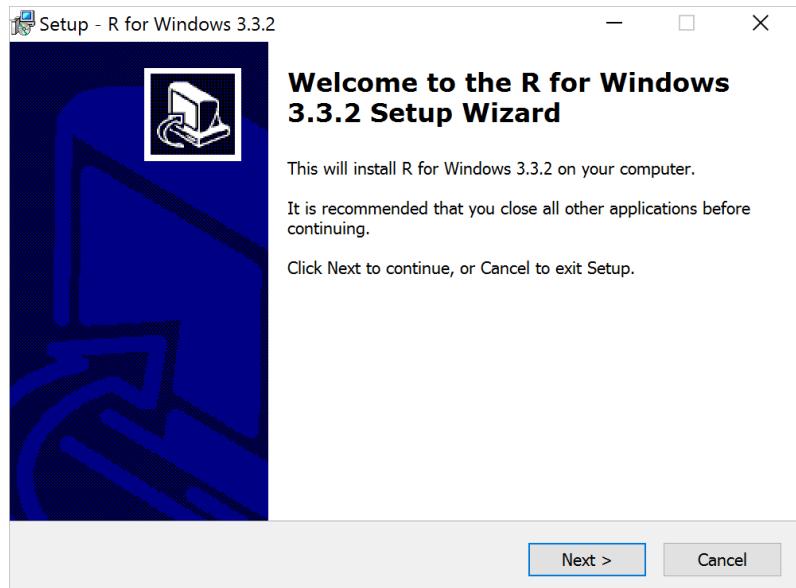
Saat ebook ini ditulis versi R adalah 3.3.2 yang release pada tanggal 31 Oktober 2016, versi ini mempunyai kode Sincere Pumpkin Patch. Nama file installer yang digunakan adalah R-3.3.2-win.exe. Proses installasi sangat mudah seperti proses installer pada platform Windows pada umumnya.

Setelah proses download selesai, klik double pada file R.3.3.2-win.exe. Langkah pertama adalah memilih bahasa yang digunakan.



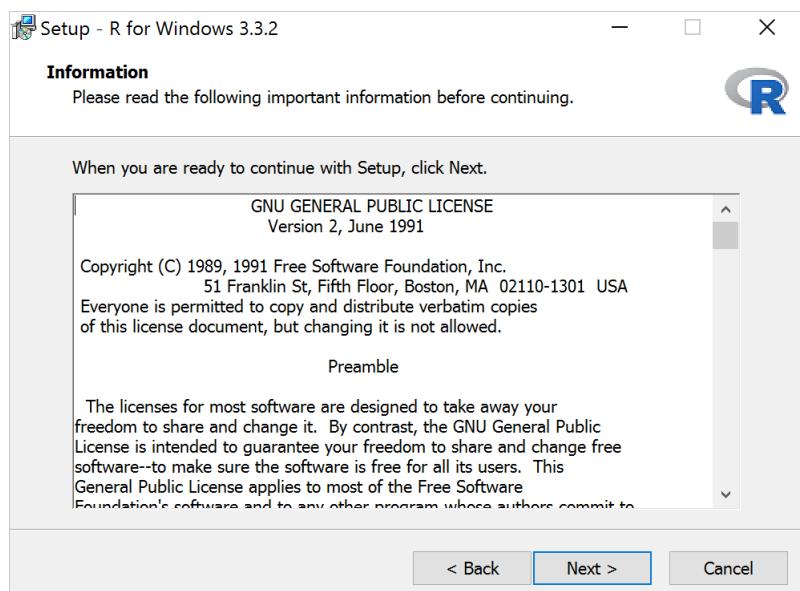
Gambar 7. Window Select Setup Language.

Klik tombol OK, kemudian akan ditampilkan window yang berisi ucapan selamat datang seperti gambar di bawah ini.



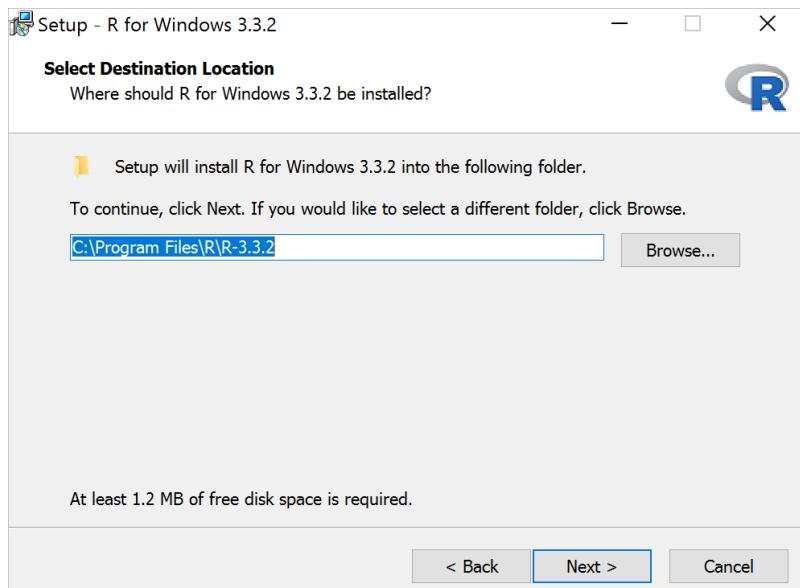
Gambar 8. Window Setup - Welcome.

Klik tombol Next, kemudian akan ditampilkan informasi penting tentang lisensi.



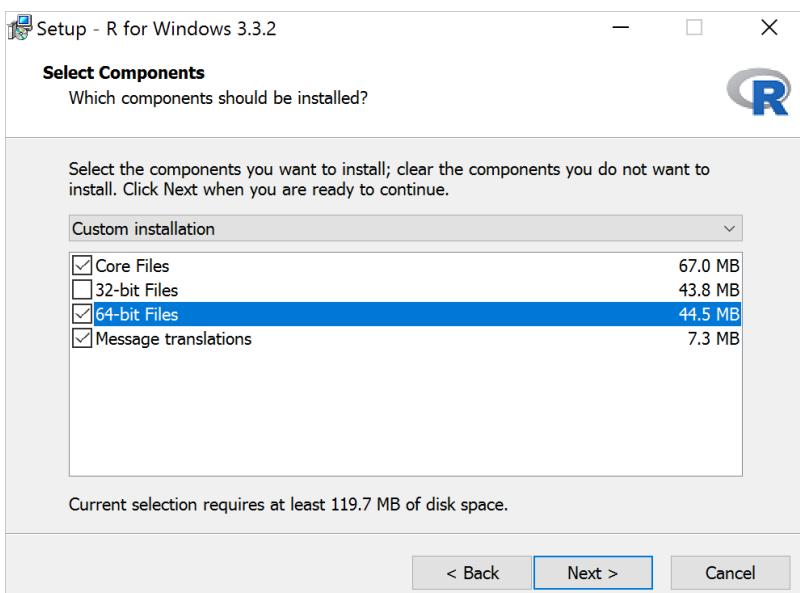
Gambar 9. Window Setup - Informasi lisensi.

Klik tombol Next, kemudian akan ditampilkan window untuk memilih folder installasi. Klik Browse jika ingin mengubah lokasi folder installasi.



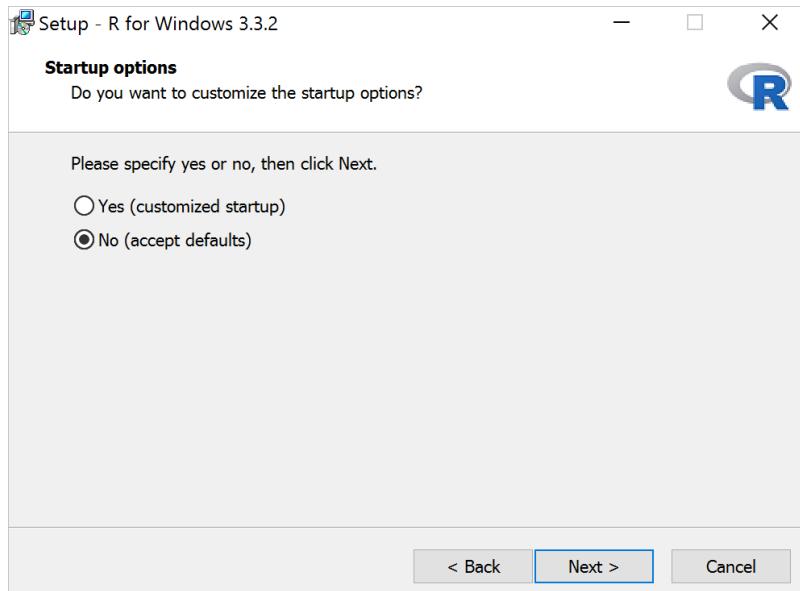
Gambar 10. Windows Setup - Lokasi folder installasi.

Klik Next, kemudian akan window pemilihan komponen untuk diinstall. Centang 32-bit Files jika menggunakan komputer dengan processor 32-bit. Atau centang 64-bit Files jika menggunakan komputer dengan processor 64-bit.



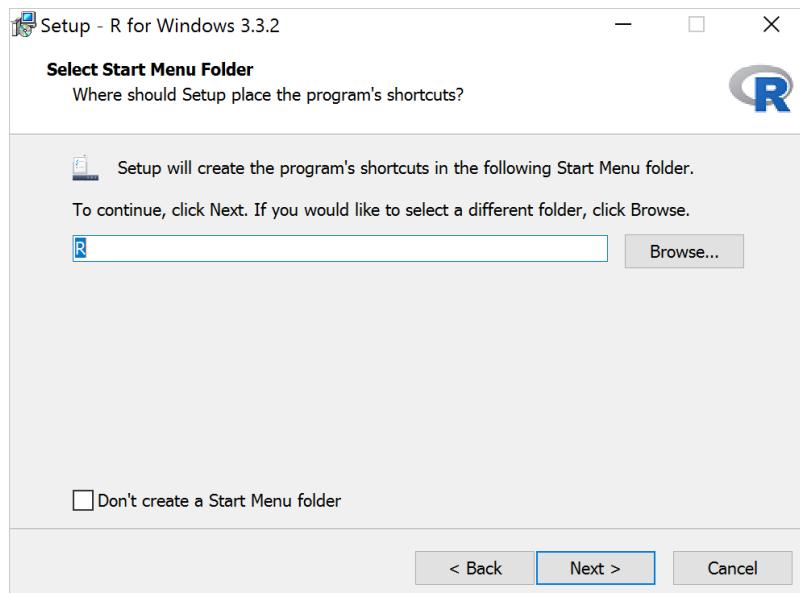
Gambar 11. Window Setup - Pemilihan komponen.

Klik Next, kemudian akan ditampilkan window untuk memilih opsi startup. Untuk memudahkan pilih No agar menggunakan opsi default startup.



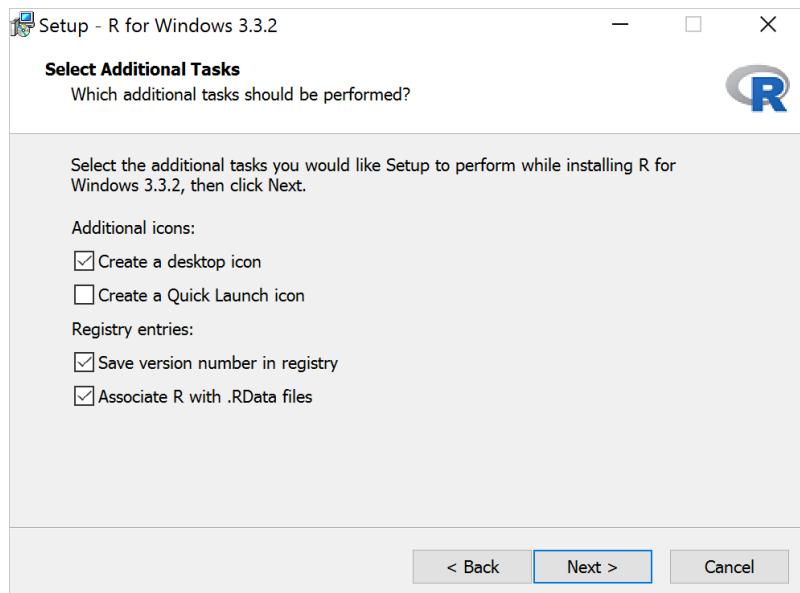
Gambar 12. Window Setup - Pemilihan opsi startup.

Kemudian akan ditampilkan window untuk memilih lokasi folder Start Menu. Klik tombol Browse jika ingin menganti lokasi folder Start Menu.



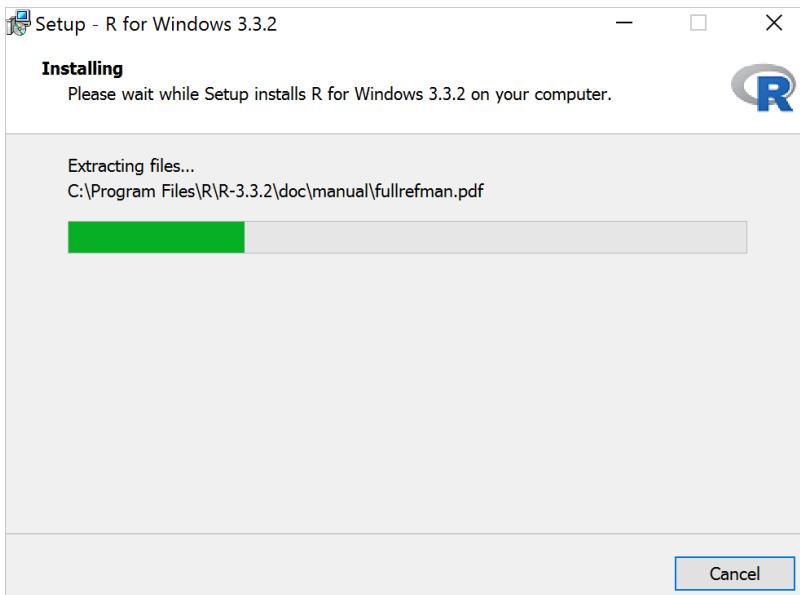
Gambar 13. Window Setup - Memilih lokasi folder Start Menu.

Klik tombol Next, kemudian akan ditampilkan window untuk memilih task tambahan yang akan dilakukan pada proses installasi. Pilih task tambahan sesuai keinginan.



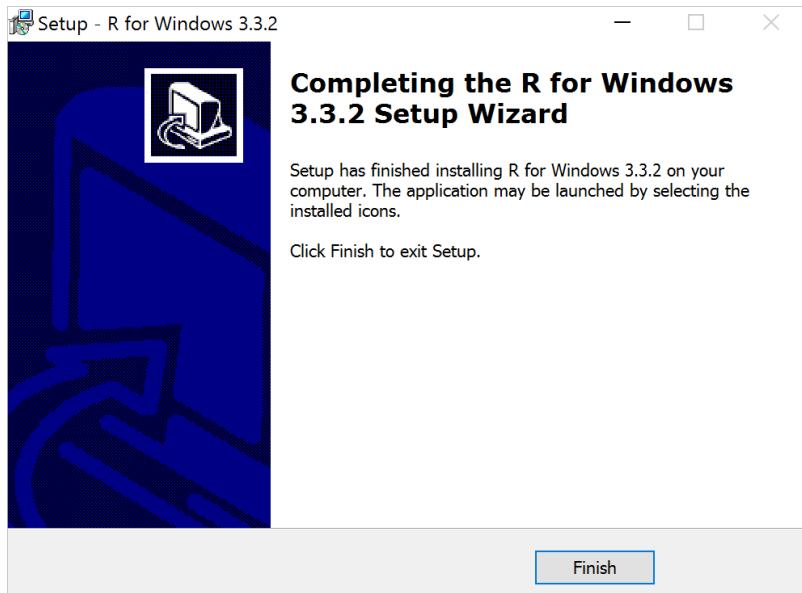
Gambar 14. Window Setup - Pemilihan task tambahan.

Klik tombol Next, kemudian proses installasi akan dilakukan.



Gambar 15. Windows Setup - Proses installasi.

Setelah proses installasi selesai maka akan ditampilkan window seperti gambar di bawah ini.



Gambar 16. Window informasi akhir proses installasi.

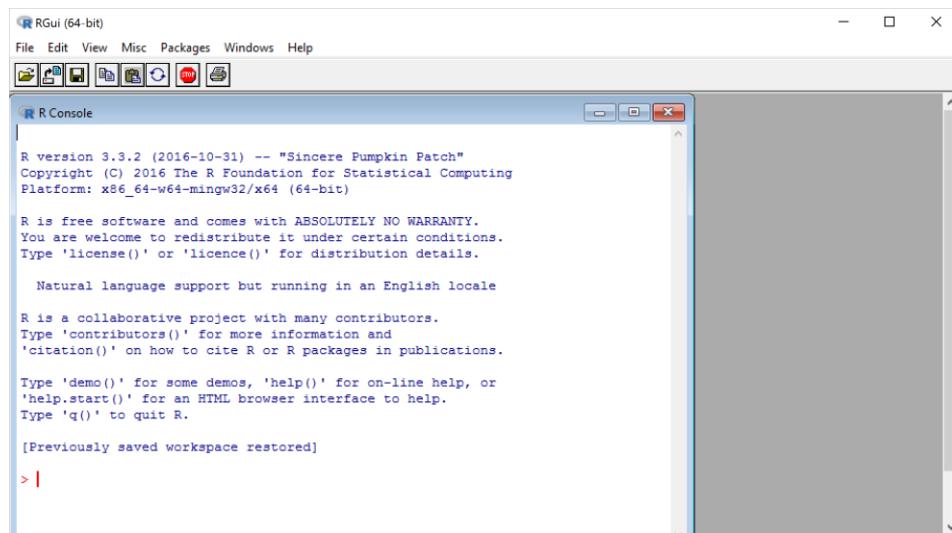
Klik tombol Finish untuk menutup proses installasi.

Tool Pemrograman

RGui

RGui adalah tool pemrograman R. RGui merupakan bagian dari platform R yang telah diinstall pada sub bab sebelumnya. RGui dapat dijalankan dengan memilih icon R x64 3.3.2 pada desktop atau memilih pada start menu.

Antarmuka RGui dapat dilihat pada gambar di bawah ini.



Gambar 17. RGui.

Console

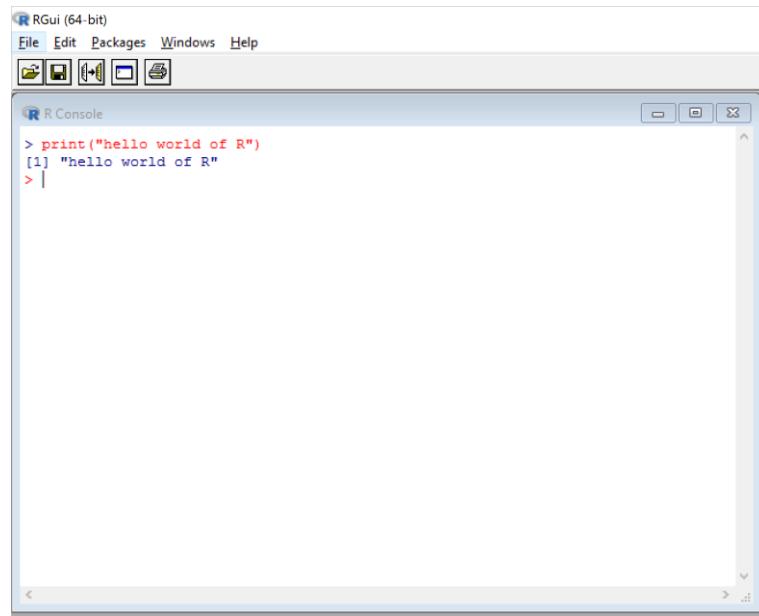
Console atau R Console adalah window yang digunakan untuk mengeksekusi fungsi R. Pada console dapat dieksekusi lebih dari satu fungsi. Console ini juga dapat digunakan untuk

mengeksekusi program yang disimpan di dalam file script R. R Console dapat dilihat pada gambar di atas.

Berikut adalah contoh penggunaan R Console. Ketik kode berikut pada window R Console.

```
print("hello world of R")
```

Kemudian tekan tombol Enter. Maka hasilnya dapat dilihat pada gambar di bawah ini.



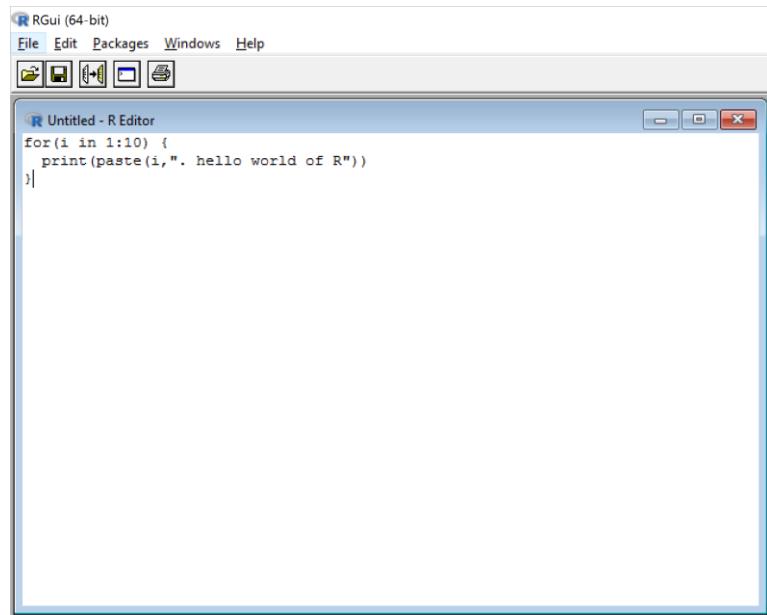
Gambar 18. R Console.

Editor

Editor adalah window untuk mengetik kumpulan fungsi R atau program script R. Window ini tidak dapat digunakan untuk menjalankan fungsi R. Window editor dapat ditampilkan dengan cara memilih menu File > New script atau dengan menekan tombol Ctrl+N. Maka akan ditampilkan window R Editor. Tuliskan kode program seperti berikut pada window R Editor.

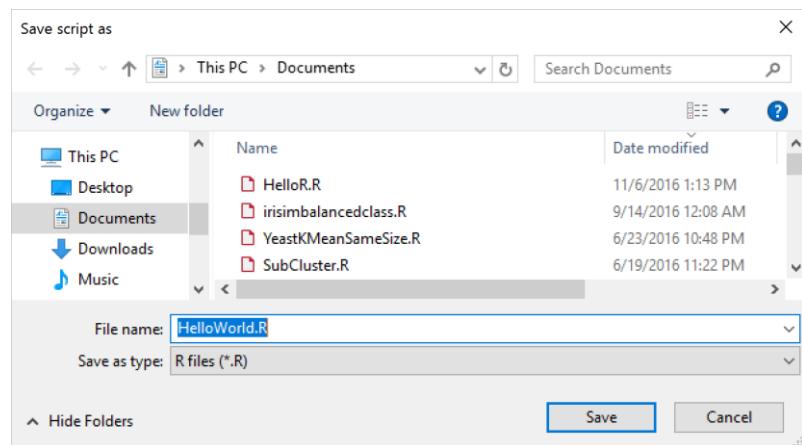
```
for(i in 1:10) {  
  print(paste(i,". hello world of R"))  
}
```

Hasilnya dapat dilihat seperti pada gambar di bawah ini.



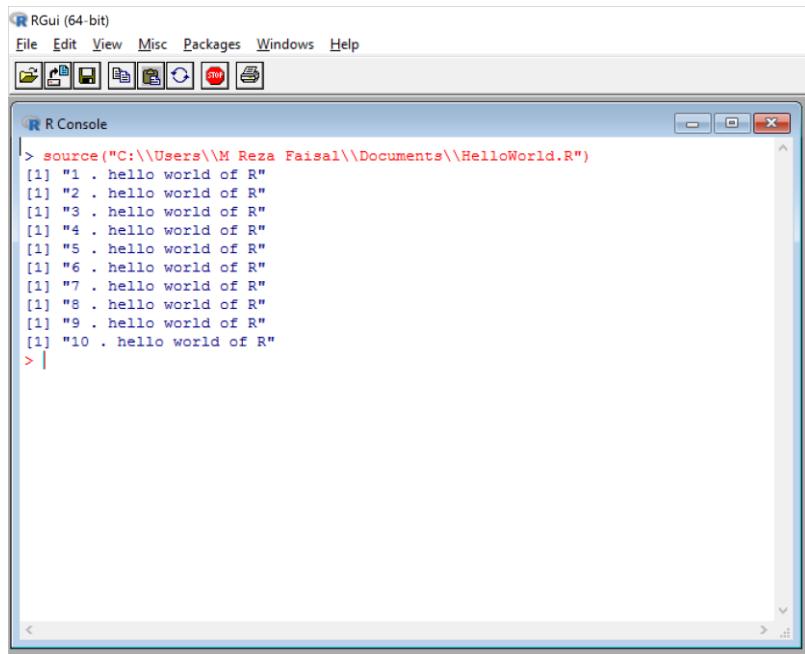
Gambar 19. Window R Editor.

Untuk menyimpan ke dalam file, pilih menu File > Save atau dengan cara menekan tombol Ctrl+S. Kemudian berikan nama file yaitu HelloWorld.R, seperti pada gambar di bawah ini.



Gambar 20. Menyimpan file script R.

Selanjutnya, pilih menu File > Source R Code. Kemudian pilih file HelloWorld.R. Maka akan dapat dilihat hasil seperti pada gambar di bawah ini.



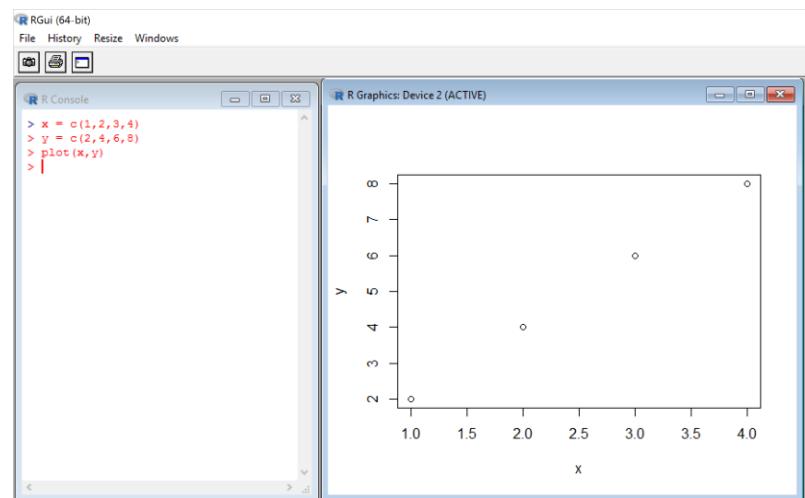
Gambar 21. Eksekusi file script R.

Grafik

RGui memiliki window grafik untuk menampilkan hasil dari fungsi menggambar grafik. Contoh fungsi untuk mengambar grafik adalah plot(). Di bawah ini adalah contoh penggunaan fungsi plot().

```
x = c(1,2,3,4)
y = c(2,4,6,8)
plot(x,y)
```

Eksekusi setiap baris di atas pada window console. Maka akan ditampilkan window grafik seperti berikut ini.



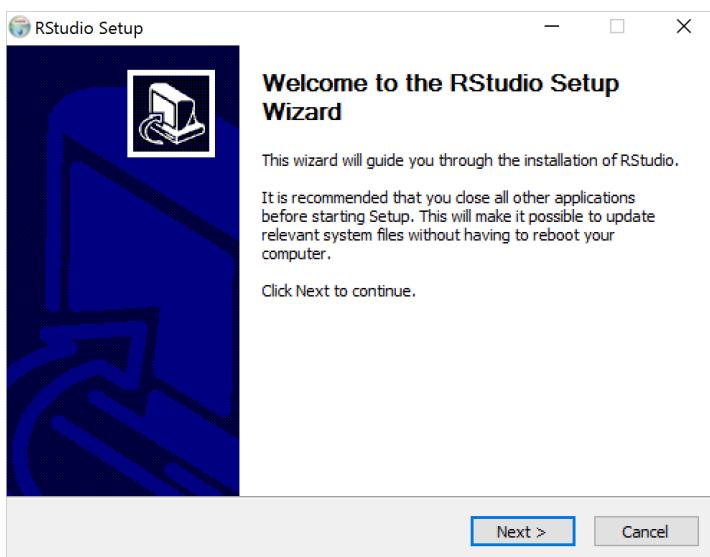
Gambar 22. Window R Graphics.

RStudio

RStudio tool pemrograman atau integrated development environment (IDE) bahasa R yang memiliki antarmuka lebih baik daripada RGui. RStudio memiliki 2 versi lisensi, yaitu Open Source Edition dan Commercial Edition. Installer RStudio dapat didownload di <https://www.rstudio.com/products/rstudio/download/>. RStudio tersedia untuk platform Windows, Mac OS X dan Linux versi Ubuntu dan Fedora.

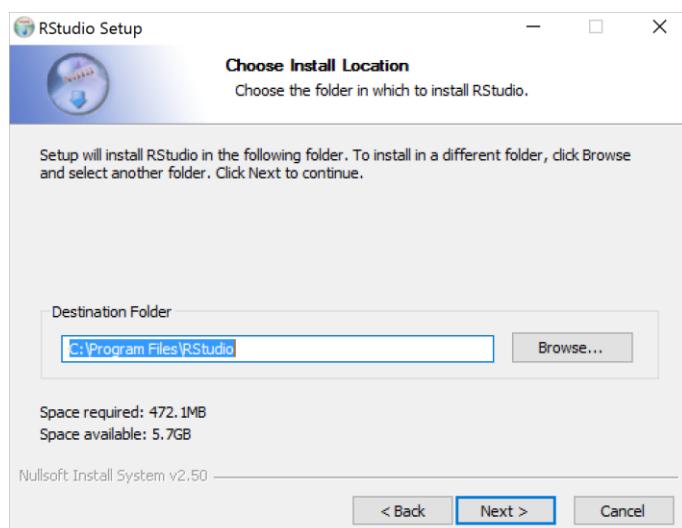
Installasi

Versi RStudio saat buku ini ditulis adalah 1.0.44. Nama file installer untuk versi ini adalah RStudio-1.0.136.exe yang berukuran 82MB. Setelah proses download selesai, klik double pada file installer.



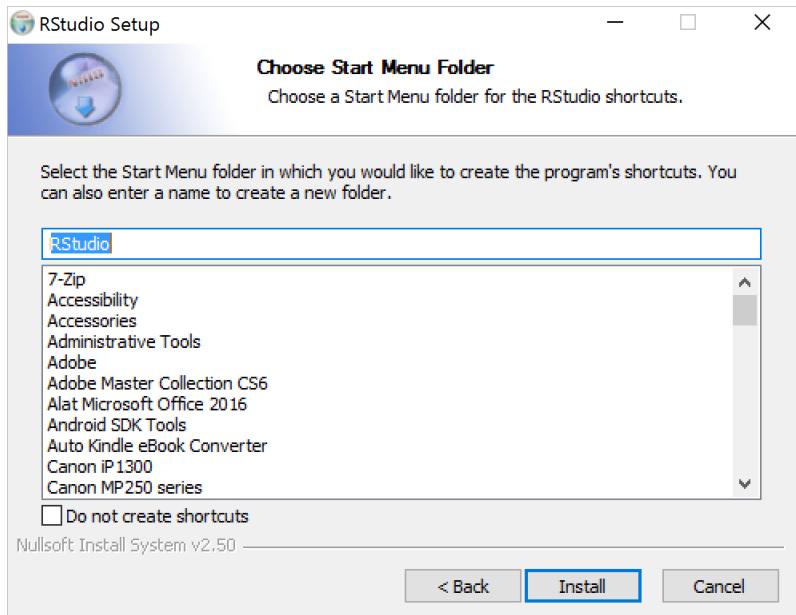
Gambar 23. Window RStudio Setup - Welcome.

Klik tombol Next, kemudian akan ditampilkan window untuk memilih lokasi installasi. Klik tombol Browse jika ingin menganti lokasi folder installasi.



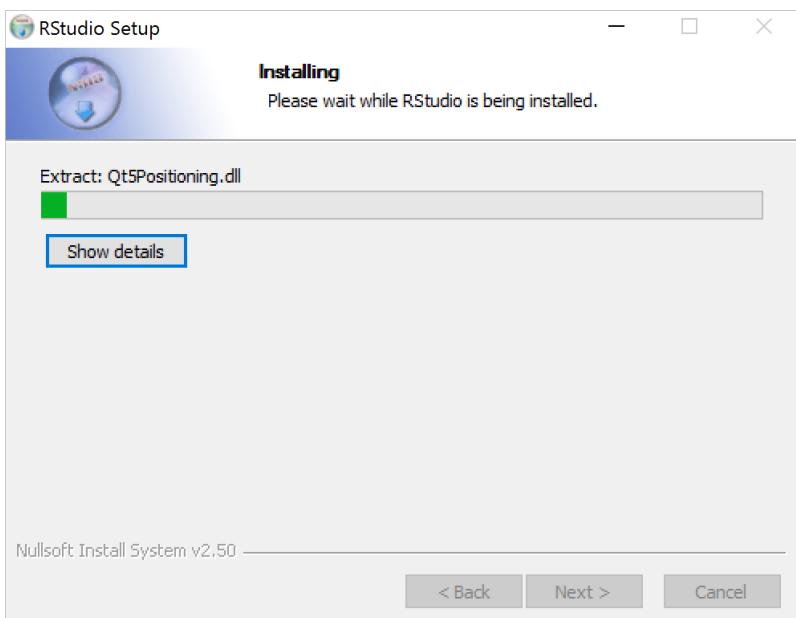
Gambar 24. Window RStudio Setup - Lokasi installasi.

Klik tombol Next, kemudian akan ditampilkan window untuk memilih folder Start Menu.



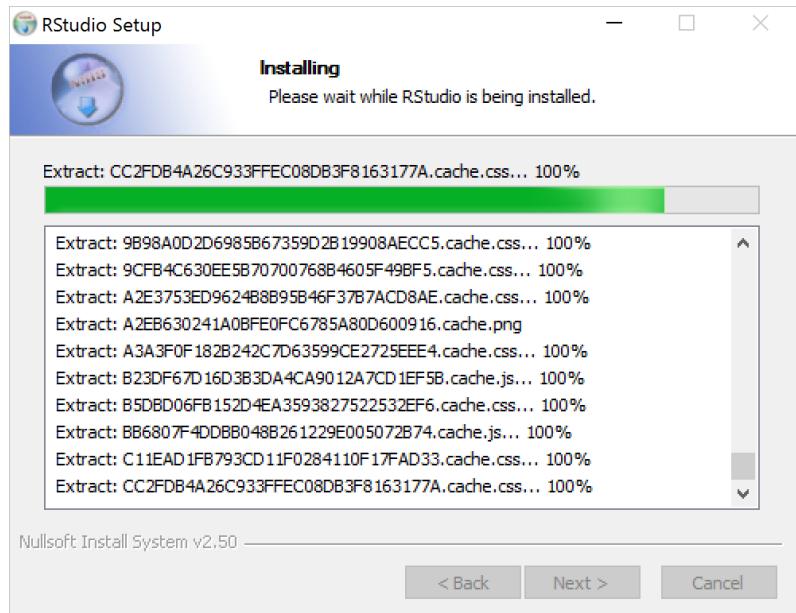
Gambar 25. Window RStudio Setup - Memilih lokasi folder Start Menu.

Klik tombol Install, maka proses installasi dimulai.



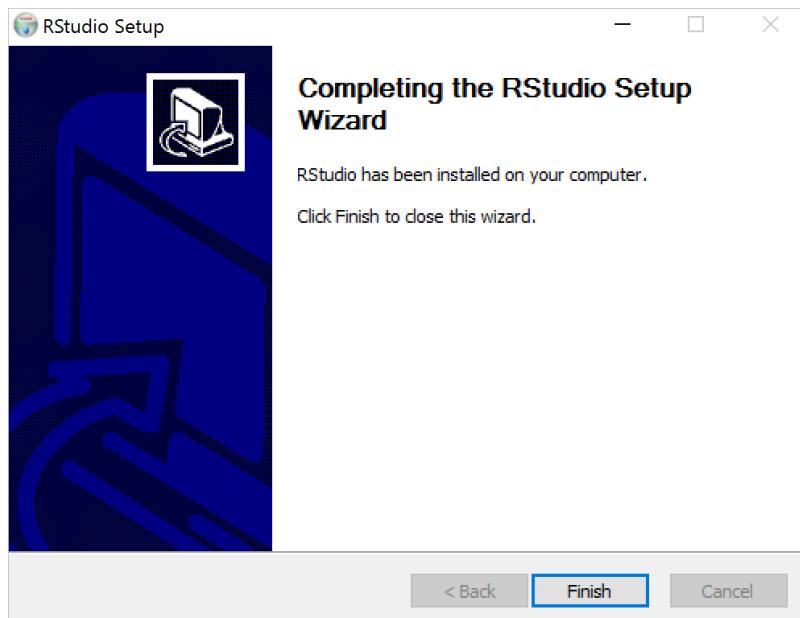
Gambar 26. Window RStudio Setup - proses installasi.

Klik tombol Show details jika ingin melihat daftar file yang diinstall.



Gambar 27. Window RStudio Setup - detail proses installasi.

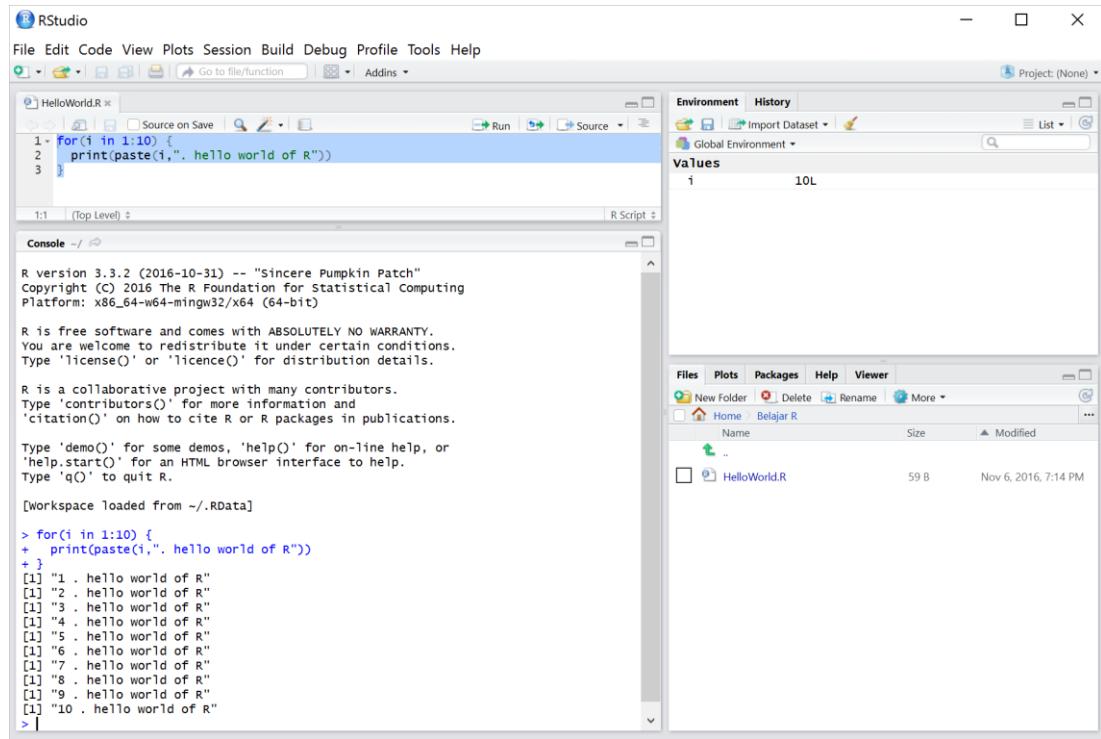
Setelah proses installasi selesai, maka akan ditampilkan window seperti pada gambar di bawah ini. Klik tombol Finish untuk mengakhiri proses installasi.



Gambar 28. Window RStudio Setup - Proses installasi selesai.

Antarmuka

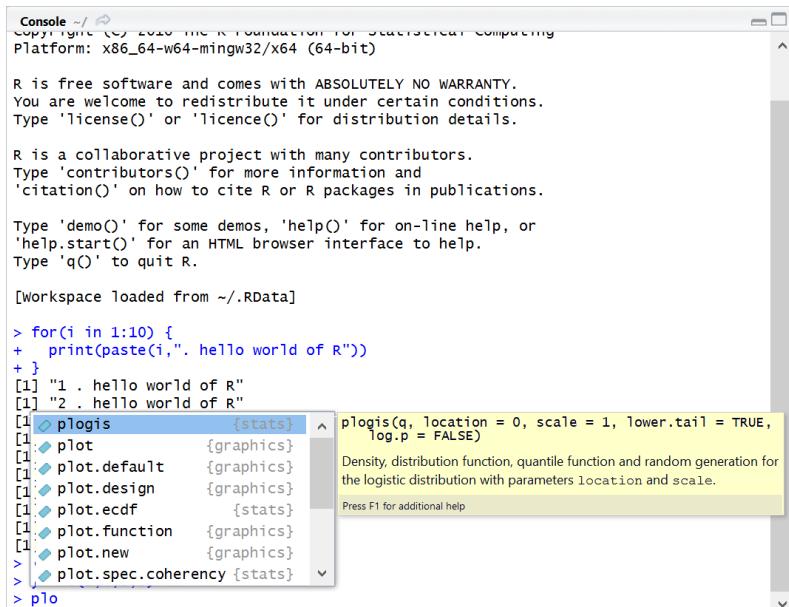
Antarmuka RStudio dapat dilihat pada gambar di bawah ini.



Gambar 29. Antarmuka RStudio.

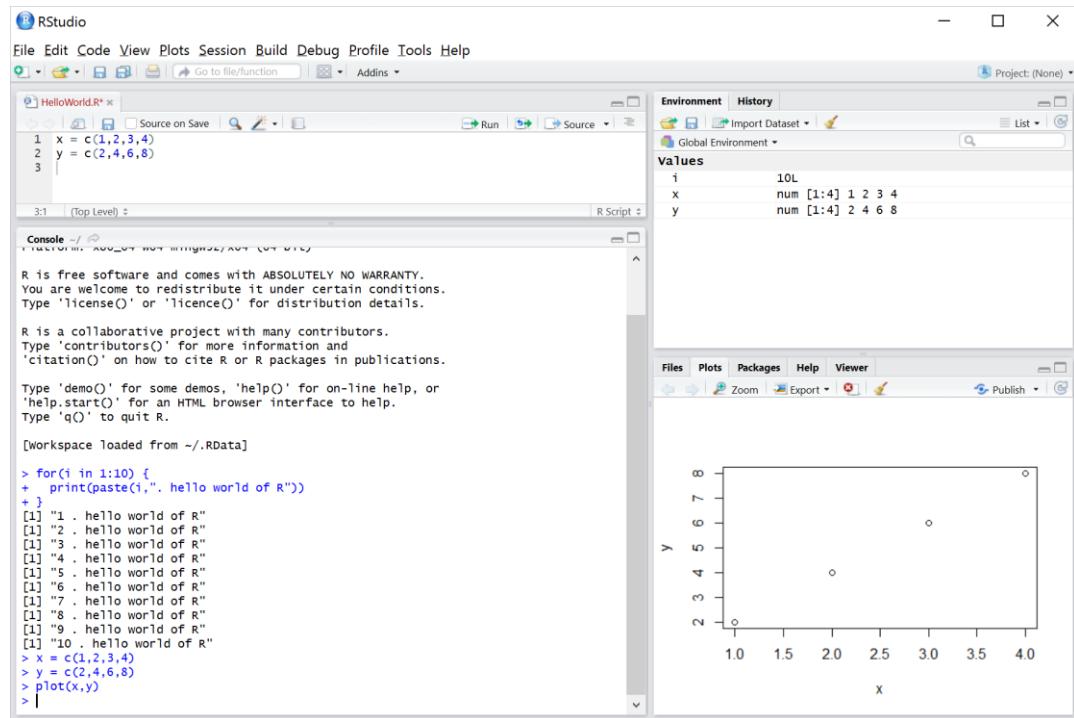
RStudio memiliki window Editor, Console dan Plot (untuk menampilkan grafik) yang terintegrasi. Selain itu RStudio juga memiliki window Environment untuk menampilkan objek atau variable yang telah digunakan. RStudio juga memiliki fitur explorer untuk melihat daftar file dan daftar folder seperti yang terlihat pada tab Files di gambar di atas. Fitur ini dapat digunakan untuk membuat folder atau menghapus folder dan file.

RStudio juga memiliki fitur auto complete kode seperti terlihat pada gambar di bawah ini.



Gambar 30. Fitur auto complete kode.

Berikut adalah contoh bagaimana RStudio menampilkan grafik pada window Plots.

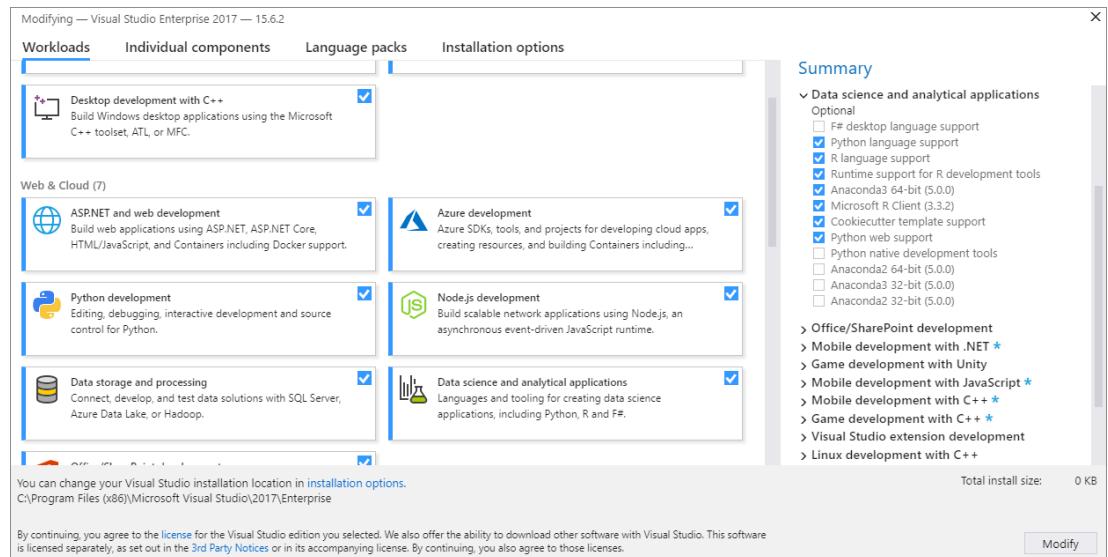


Gambar 31. Window Plots untuk menampilkan grafik.

Visual Studio 2017

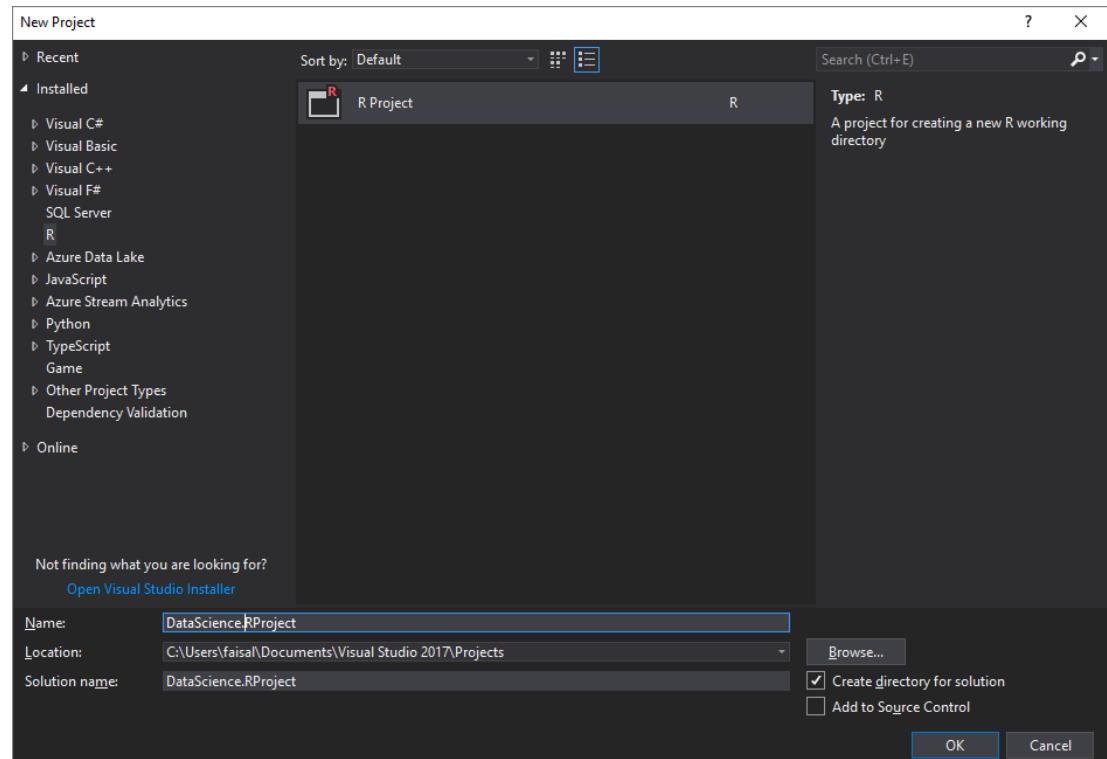
Visual Studio adalah salah satu tool yang dapat digunakan untuk menulis program dengan bahasa R. Dukungan ini telah ada sejak Visual Studio 2015, dengan adanya R Development Tools sebagai tool tambahan. Tetapi R telah menjadi salah satu bahasa utama pada Visual Studio 2017.

Pada window installasi dapat dilihat bahwa dukungan bahasa R ada pada kelompok Data science and analytical applications.



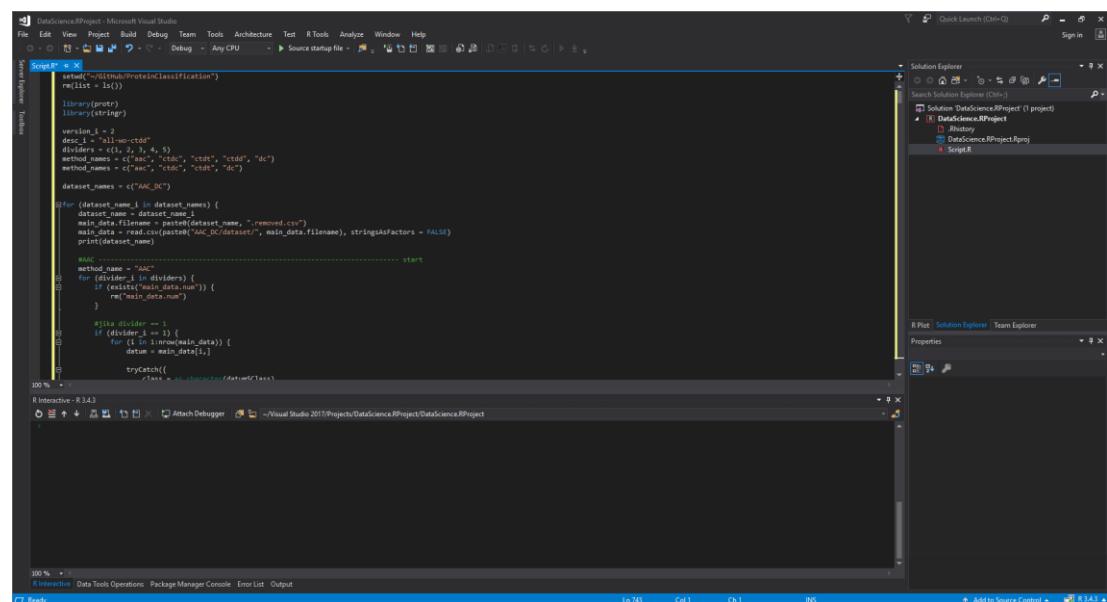
Gambar 32. Visual Studio 2017 – R language support.

Untuk memulai menggunakan bahasa pemrograman R, maka terlebih dahulu membuat project R.



Gambar 33. Visual Studio 2017 – R Project.

Setelah itu akan ditampilkan editor seperti berikut ini.



Gambar 34. Visual Studio 2017 – R editor.

2

Dasar-Dasar Bahasa Pemrograman R

Pendahuluan

Sebagai bahasa pemrograman, R mempunyai gaya penulisan kode pemrograman seperti C#, Java, PHP atau Javascript. Bagi pembaca yang telah mengenal bahasa-bahasa pemrograman tersebut maka akan mudah untuk mempelajari bahasa pemrograman R, karena cara penulisan kode untuk percabangan, pengulangan, block dan komentar yang sama.

Bahasa pemrograman R bukan bahasa pemrograman strong type seperti Java atau C#. Cara penulisan kode program R lebih dekat dengan pemrograman PHP atau Javascript, salah satunya adalah sama-sama tidak perlu melakukan deklarasi tipe data dan kebebasan memanfaatan variable. Hal ini akan dibahas lebih lanjut pada sub bab selanjutnya.

Perbedaan mendasar antara R dengan keempat bahasa di atas adalah fungsi-fungsi R dapat langsung pada R console dan keluarannya juga dapat langsung dilihat pada R console. Contoh-contoh pada bab ini sebagian besar ditulis pada R console agar mempermudah pembaca melihat langsung keluaran dari contoh penggunaan fungsi-fungsi yang diberikan.

Working Directory

Working directory atau direktori kerja adalah direktori/folder aktif pada suatu session. User hanya dapat mengakses file script R dan file data yang berada pada working directory. Jika user ingin mengakses file di luar working directory maka user harus menulis path direktori/folder dan nama file.

Untuk mendapatkan informasi working directory dapat digunakan fungsi getwd(). Di bawah ini adalah contoh penggunaan fungsi getwd() dan outputnya.

```
> getwd()  
[1] "D:/Data/My Projects/Delete/HelloWorldR/HelloWorldR"
```

Untuk menentukan working directory dapat digunakan fungsi setwd(). Sintaks fungsi ini adalah sebagai berikut.

```
setwd("path_absolute")
```

Kode di bawah ini adalah contoh penggunaan fungsi setwd() dan outputnya.

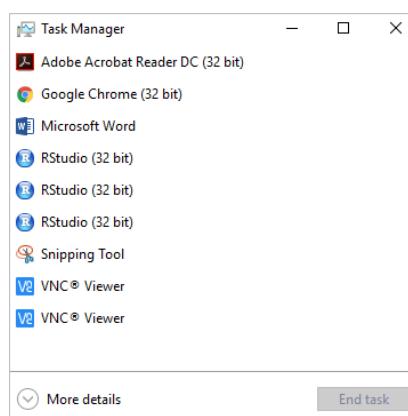
```
> setwd("C:/")  
> getwd()  
[1] "C:/"
```

Setelah working directory ditentukan maka jika ada keperluan menulis atau menyimpan file, file akan disimpan pada working directory yang aktif.

R Session

Setiap RGui atau RStudio dijalankan maka akan digunakan sebuah R session. Begitu juga jika R console dijalankan pada command prompt. Jika ada kebutuhan untuk menjalankan beberapa program sekaligus dengan menggunakan RStudio maka dapat dilakukan cara membuat session baru untuk setiap program.

Untuk membuat session baru pada RStudio dapat dilakukan dengan memilih menu Session > New Session. Misal dibuat 3 session maka dapat akan aktif 3 window RStudio. Selain itu juga terlihat ada 3 proses RStudio pada Task Manager pada sistem operasi Windows.



Gambar 35. Tiga R session aktif.

Variable

Variable pada R tidak perlu melakukan deklarasi tipe data untuk variable yang digunakan. Tipe data variable akan sesuai dengan data yang dimasukkan pada variable tersebut. Untuk membuat atau mendeklarasikan variable pada lingkungan R dapat dilakukan dengan dua cara yaitu dengan menggunakan tanda = atau <- seperti contoh berikut ini.

```
a = 1  
b <- 2
```

Keduanya mempunyai efek yang sama. Tanda panah <- juga dapat dipergunakan secara terbalik, seperti pada contoh di bawah ini.

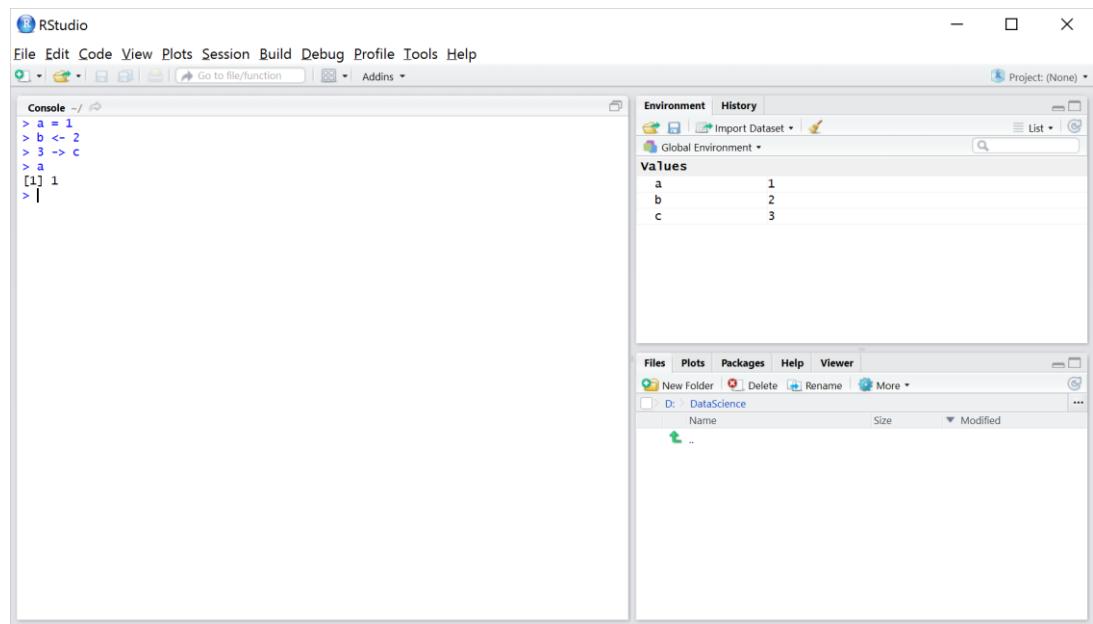
```
3 -> c
```

Cara di atas tidak berlaku untuk tanda =. Selanjutnya variable pada lingkungan R dapat juga disebut sebagai obyek.

Untuk melihat atau mengevaluasi nilai dalam suatu variable atau obyek dapat dilakukan dengan memanggil obyek tersebut.

```
> a  
[1] 1
```

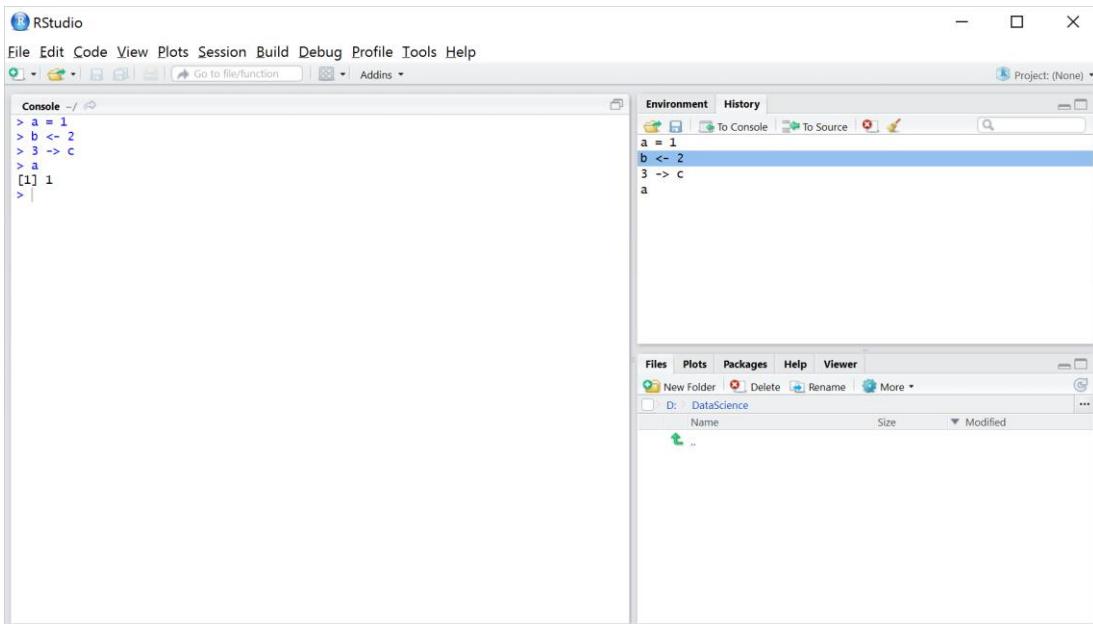
Pada RStudio kode di atas dapat ditulis langsung pada window console seperti pada gambar di bawah ini.



Gambar 36. RStudio - Environment.

Window console juga menjadi tempat untuk menampilkan output dari kode yang dituliskan. Sebagai contoh, menampilkan nilai dari obyek a. RStudio memiliki feature untuk menampilkan daftar obyek yang telah dibuat. Pada gambar di atas, daftar obyek dapat dilihat pada tab Environment.

RStudio juga memiliki feature untuk menampilkan history daftar kode yang telah dieksekusi. Daftar tersebut dapat dilihat pada tab History. Kode pada daftar tersebut dapat dipilih dengan cursor mouse seperti contoh pada gambar di bawah ini. Kode yang dipilih tersebut dapat dieksekusi kembali dengan cara mengklik tombol “To Console”.



Gambar 37. RStudio - History.

Function

Selain menggunakan fungsi yang telah disediakan pada lingkungan R, pengguna dimungkinkan untuk membuat fungsi sendiri. Sintaks yang digunakan untuk membuat fungsi dapat dilihat di bawah ini. Pada bahasa pemrograman terdapat dua jenis fungsi yaitu fungsi yang tidak mengembalikan nilai dan fungsi yang mengembalikan nilai.

Untuk fungsi yang tidak mengembalikan nilai maka digunakan sintaks seperti berikut ini.

```
NamaFungsi <- function(arg1,arg2,...) {  
  ...  
}
```

Atau

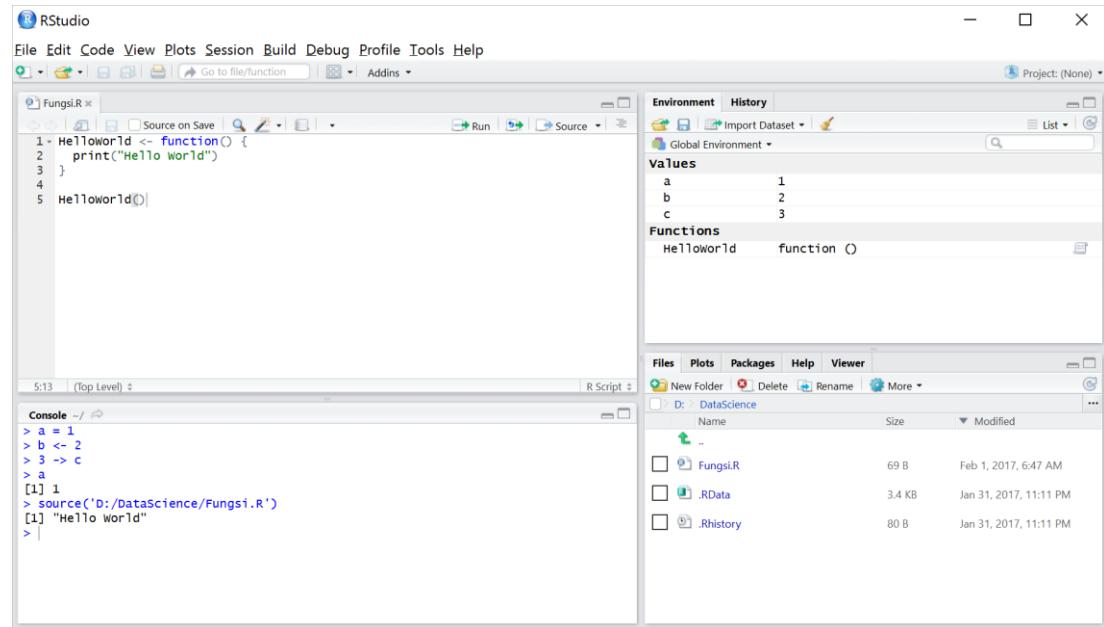
```
NamaFungsi = function(arg1,arg2,...) {  
  ...  
}
```

Fungsi yang dibuat sebaiknya disimpan pada file script R. Untuk membuat file script R pada RStudio dilakukan dengan cara memilih menu File > New File > R Script. Kemudian ketik kode berikut ini pada window editor.

```
HelloWorld <- function() {  
  print("Hello World")  
}  
  
HelloWorld()
```

Pada Fungsi.R, selain ditulis deklarasi fungsi HelloWorld() juga terdapat pemanggilan fungsi tersebut seperti yang dapat dilihat pada baris terakhir.

Selanjutnya simpan file dengan nama Fungsi.R. Pada gambar di bawah ini dapat dilihat tampilan window editor. File Fungsi.R akan terlihat tab Files yang menampilkan daftar file pada working directory.



Gambar 38. RStudio - Editor & Files.

Untuk menjalankan seluruh baris pada file yang sedang aktif dapat dilakukan dengan cara mengklik tombol Source. Tombol Source terletak pada pojok kanan pada window editor. Setelah tombol diklik maka obyek dan fungsi yang digunakan pada file yang sedang aktif

akan dibuat. Pada gambar di atas dapat dilihat terdapat tambahan area Functions pada tab Environtment.

Sedangkan fungsi yang mengembalikan nilai digunakan sintaks seperti berikut ini.

```
NamaFungsi <- function(arg1,arg2,... ) {  
  ...  
  return(object)  
}
```

Atau

```
NamaFungsi = function(arg1,arg2,... ) {  
  ...  
  return(object)  
}
```

Contoh penggunaan sintaks di atas dapat dilihat pada contoh di bawah ini. Contoh kode di bawah ini disimpan ke dalam file Matamatika.R.

```
Tambah = function(a, b) {  
  c = a + b  
  return(c)  
}  
  
hasil = Tambah(10, 3)  
print(hasil)
```

Tipe Data

Walaupun variable atau obyek tidak harus dideklarasikan tipe datanya sebelum digunakan, tetapi lingkungan R tetap mengenal tipe data. Pada lingkungan R tipe data digolongkan menjadi:

1. Skalar.
2. Vector.
3. Matrix.
4. Data Frame.
5. List.
6. Dan lain-lain.

Skalar

Contoh tipe skalar sudah sering digunakan pada contoh-contoh sebelumnya, yaitu:

1. Number, secara default digunakan double.
2. Character, selain single character juga dapat menangani string.
3. Logical, yaitu bernilai TRUE atau FALSE.

Date

Date atau tanggal pada R dikelola dalam nilai angka yaitu jumlah hari sejak 1970-01-01. Obyek bertipe data date dibuat dengan mengubah string tanggal menjadi tanggal dengan menggunakan fungsi `as.Date()`. Berikut adalah contohnya.

```
new_year = as.Date("2017-01-01")
```

Contoh lain adalah sebagai berikut:

```
today = as.Date(Sys.time())
```

Fungsi Sys.time() mengeluarkan output tanggal dan jam waktu server.

Untuk obyek date dapat dilakukan proses matematika seperti pengurangan untuk mengetahui perbedaan hari antara dua tanggal.

```
> today - new_year  
Time difference of 31 days
```

Time

Time atau jam pada R dikelola dalam nilai angka yaitu jumlah detik sejak 1970-01-01. Fungsi yang digunakan untuk membuat obyek time adalah as.POSIXlt(). Berikut adalah contoh penggunaan fungsi tersebut.

```
> after_new_year = as.POSIXct("2017-01-01 11:11:11 EDT")  
  
> after_new_year  
[1] "2017-01-01 11:11:11 JST"
```

Sedangkan untuk membuat obyek yang menyimpan waktu sekarang dapat digunakan cara seperti contoh di bawah ini.

```
> now = as.POSIXct(Sys.time())  
  
> now  
[1] "2017-02-01 11:38:37 JST"
```

Seperti halnya obyek date, obyek time juga dapat diproses untuk mengetahui perbedaan waktu dari dua obyek.

```
> now - after_new_year  
Time difference of 31.01906 days
```

Vector

Untuk membuat vector digunakan bantuan fungsi c(). Vector dapat berisi tipe data skalar seperti number, character atau logical. Berikut ini adalah contoh membuat obyek vector.

```
vec_number <- c(1,2,-5.13,6,-2,4)  
vec_char = c("mohammad", "reza", "faisal")  
vec_bool = c(TRUE, FALSE, TRUE, TRUE)
```

Berikut adalah nilai dari setiap obyek di atas.

```
> vec_number  
[1] 1.00 2.00 -5.13 6.00 -2.00 4.00  
  
> vec_char  
[1] "mohammad" "reza" "faisal"  
  
> vec_bool  
[1] TRUE FALSE TRUE TRUE
```

Jika ingin menampilkan atau mengakses elemen dari vec_number dapat digunakan cara sebagai berikut.

```
> vec_number[0]  
numeric(0)  
  
> vec_number[1]  
[1] 1  
  
> vec_number[2]  
[1] 2  
  
> vec_number[3]  
[1] -5.13
```

```

> vec_number[4]
[1] 6

> vec_number[5]
[1] -2

> vec_number[6]
[1] 4

```

Pada contoh output di atas terdapat kode `vec_number[0]` yang menghasilkan output `numeric(0)`, hal ini menandakan bahwa index tidak dimulai dari 0 tetapi dimulai dari 1.

Selain cara di atas, dapat juga digunakan cara berikut ini untuk mengakses elemen vector.

```

> vec_number[c(1)]
[1] 1

> vec_number[c(1,3)]
[1] 1.00 -5.13

> vec_number[c(1,3,6)]
[1] 1.00 -5.13 4.00

```

Jika ingin mengakses lebih dari satu elemen sekaligus dapat digunakan cara seperti cara di atas ini. Sedangkan jika ingin menganti nilai suatu elemen pada vector dapat digunakan cara seperti contoh di bawah ini.

```
vec_number[1] = -13
```

Factor

Factor digunakan untuk mengelola data kategori. Factor dapat dibuat dengan bantuan fungsi `c()` dan `factor()`. Sebagai contoh misalnya dimiliki nilai ujian akhir mahasiswa pada suatu kelas yang disimpan pada vector berikut.

```
nilai_uas = c("A", "A", "C", "B", "A", "E", "E", "A", "C", "D", "C"
,"D", "A" , "C")
```

Dan berikut adalah output jika obyek `nilai_uas` dipanggil.

```
> nilai_uas
[1] "A" "A" "C" "B" "A" "E" "E" "A" "C" "D" "C" "D" "A" "C"
```

Untuk mengubah obyek tersebut menjadi factor digunakan fungsi `factor()` seperti contoh di bawah ini.

```
nilai_uas = factor(nilai_uas)
```

Dan berikut adalah output ketika obyek `nilai_uas` dipanggil.

```
> nilai_uas
[1] A A C B A E E E A C D C D A C
Levels: A B C D E
```

Dari kedua output di atas dapat dilihat perbedaan, output pertama hanya memberikan nilai-nilai yang disimpan oleh obyek `nilai_uas`. Setelah obyek `nilai_uas` diubah menjadi factor maka outputnya tidak hanya memberikan nilai-nilai yang disimpan oleh obyek tetapi juga menampilkan keterangan `Levels` yang mengkategorikan isi dari obyek.

Untuk mengubah nilai pada factor harus sesuai dengan nilai yang telah dikategorikan sesuai nilai yang terlihat pada `Levels`. Sebagai contoh pada gambar di atas nilai `Levels : A B C D E`, artinya tidak dapat mengubah nilai elemen selain nilai tersebut.

Sebagai contoh adalah sebagai berikut.

```
> nilai_uas[1] = "D"
> nilai_uas
[1] D A C B A E E E A C D C D A C
Levels: A B C D E

> nilai_uas[1] = "F"
Warning message:
In `[<-factor`(`*tmp*`, 1, value = "F") :
  invalid factor level, NA generated
```

Pada contoh pertama dapat dilihat nilai elemen 1 diubah nilainya menjadi D, dan berhasil. Karena nilai D telah dikategorikan. Sedangkan ketika nilai elemen 1 diubah menjadi F maka akan ditampilkan pesan kesalahan. Hal ini karena nilai F tidak termasuk di dalam kategori.

Matrix

Matrix adalah vector yang memiliki atribut dimensi. Untuk membuat matrix dapat mengikuti sintaks seperti berikut ini.

```
var_matrix<- matrix(data, nrow=r, ncol=c, byrow=FALSE, dimnames=list(char_
vector_rownames, char_vector_colnames))
```

Keterangan:

1. data: adalah isi dari matrix. Nilai dapat diisi dari vector.
2. nrow: jumlah baris.
3. ncol: jumlah kolom.
4. byrow: jika nilainya FALSE maka matrix diisi berdasarkan kolom, sebaliknya akan diisi berdasarkan row.
5. dimnames: digunakan untuk menentukan nama kolom dan baris.

Dalam membuat matrix harus diperhatikan bahwa:

1. Seluruh kolom pada matrix harus berisi tipe data yang sama (numeric, char, boolean).
2. Seluruh kolom harus memiliki panjang yang sama.

Berikut ini adalah contoh membuat matrix. Pada contoh di bawah ini akan dibuat dua buah matrix sederhana tetapi dengan menggunakan nilai parameter byrow yang berbeda. Secara default nilai parameter byrow adalah FALSE.

```
> matrix_y = matrix(1:20, nrow = 5, ncol = 4)
> matrix_y
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20

> matrix_x = matrix(1:20, nrow = 5, ncol = 4, byrow = TRUE)
> matrix_x
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
[4,]   13   14   15   16
[5,]   17   18   19   20
```

Dari contoh di atas nilai yang diberikan pada matrix adalah angka 1 sampai 20 yang cukup ditulis dengan cara 1:20. Pada contoh di atas dibuat dua buah obyek yaitu matrix_y yang pengisian nilainya berdasarkan kolom dan matrix_x yang pengisiannya nilainya berdasarkan baris. Obyek matrix_y diisi berdasarkan baris karena mengisikan nilai pada parameter byrow.

Contoh pembuatan matrix yang lain dapat dilihat pada contoh di bawah ini.

```
data = c(1,26,24,68)
nama_baris = c("R1", "R2")
nama_kolom = c("C1", "C2")
matrix_z = matrix(data, nrow=2, ncol=2, byrow=TRUE,
dimnames=list(nama_baris, nama_kolom))
```

Baris pertama adalah membuat obyek data yang menampung data dalam bentuk vector. Baris kedua adalah membuat obyek nama_baris yang akan menjadi nama baris sedangkan pada baris ketiga adalah membuat obyek nama_kolom. Selanjutnya adalah membuat obyek matrix_z. hasilnya dapat dilihat seperti output di bawah ini.

```
> matrix_z
   C1 C2
R1  1 26
R2 24 68
```

Selanjutnya adalah cara untuk mengakses matrix. dari contoh di atas dapat dilihat untuk menampilkan isi seluruh matrix cukup dengan memanggil nama obyek matrix yang diinginkan yaitu x atau x atau z.

Jika ingin mengakses seluruh baris kedua pada matrix z maka dapat digunakan perintah berikut ini.

```
> matrix_z[2,]
C1 C2
24 68

> matrix_z["R2",]
C1 C2
24 68
```

Sedangkan jika ingin mengakses seluruh kolom pertama pada matrix z maka digunakan perintah berikut ini.

```
> matrix_z[,1]
R1 R2
1 24

> matrix_z[, "C1"]
R1 R2
1 24
```

Sedangkan jika ingin mengakses nilai elemen pada elemen baris ke-2 dan kolom ke-2 digunakan perintah berikut ini.

```
> matrix_z[2,2]
[1] 68

> matrix_z["R2", "C2"]
[1] 68
```

Selain itu dapat juga mengakses matrix dengan keluaran berupa matrix seperti contoh di bawah ini.

```
> matrix_x[2:4,1:3]
      [,1] [,2] [,3]
[1,]     5     6     7
[2,]     9    10    11
[3,]    13    14    15
```

Maksud dari perintah itu adalah mengakses elemen matrix baris 2 sampai dengan 4 dan kolom 1 sampai dengan 3.

Sedangkan jika ingin memodifikasi nilai elemen pada matrix dapat dilakukan dengan cara seperti contoh berikut ini.

```
> matrix_x[1,1] = 20
> matrix_x
 [,1] [,2] [,3] [,4]
[1,] 20 2 3 4
[2,] 5 6 7 8
[3,] 9 10 11 12
[4,] 13 14 15 16
[5,] 17 18 19 20

> matrix_x[1,] = 20
> matrix_x
 [,1] [,2] [,3] [,4]
[1,] 20 20 20 20
[2,] 5 6 7 8
[3,] 9 10 11 12
[4,] 13 14 15 16
[5,] 17 18 19 20
```

Baris pertama bertujuan untuk memberikan nilai baru pada elemen matrix baris 1 dan kolom 1. Sedangkan kode pada baris kedua bertujuan untuk mengubah nilai seluruh elemen pada baris pertama.

Array

Array, tipe data ini, mempunyai karakteristik seperti matrix bedanya adalah dimensi yang dimiliki array dapat lebih dari 2, seperti diketahui bahwa matrix hanya dapat memiliki dimensi $N \times N$. Array memungkinkan untuk memiliki dimensi:

1. $N \times N \times N$.
2. $N \times N \times N \times N$.
3. Dan seterusnya.

Berikut ini adalah contoh membuat obyek bertipe array.

```
number_arr = array(1:27, dim=c(3,3,3))
```

Dan berikut adalah isi dari obyek number_arr.

```
> number_arr
, , 1
 [,1] [,2] [,3]
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9

, , 2
 [,1] [,2] [,3]
[1,] 10 13 16
[2,] 11 14 17
[3,] 12 15 18

, , 3
 [,1] [,2] [,3]
[1,] 19 22 25
```

```
[2,] 20 23 26
[3,] 21 24 27
```

Untuk mengakses obyek array berdimensi 3x3x3 ini dapat dilakukan dengan cara berikut ini, yang akan menghasilkan keluaran yaitu 9.

```
> number_arr [3,3,1]
[1] 9
```

Cara lain yang lebih rumit untuk membuat array dapat dilihat dari contoh di bawah ini

```
tiga_d_array = array(
  1:24,
  dim = c(4, 3, 2),
  dimnames = list(
    c("satu", "dua", "tiga", "empat"),
    c("siji", "loro", "telu"),
    c("hiji", "dua")
  )
)
```

Contoh di atas adalah membuat array dengan memberikan nama untuk kolom dan baris dimensinya.

```
> tiga_d_array
, , hiji

      siji  loro  telu
satu     1     5     9
dua      2     6    10
tiga     3     7    11
empat    4     8    12

, , dua

      siji  loro  telu
satu    13    17    21
dua     14    18    22
tiga    15    19    23
empat   16    20    24
```

Data Frame

Data frame lebih general jika dibandingkan dengan matrix. data frame digunakan untuk menyimpan data tabular. Data frame dapat memiliki tipe data skalar yang berbeda, tetapi jumlah elemen seluruh kolom tetap harus sama.

Berikut ini adalah sintaks yang dapat digunakan.

```
data.frame(..., row.names = NULL, check.rows = FALSE,
           check.names = TRUE,
           stringsAsFactors = default.stringsAsFactors())
```

Dan di bawah ini adalah contoh penggunaan fungsi ini.

```
a = c(1,2,3,4)
b = c("green", "white", "red", "blue")
c = c(TRUE,TRUE,TRUE,FALSE)
data_frame = data.frame(a,b,c, stringsAsFactors = FALSE)
names(data_frame) = c("ID","Color","Passed")
```

Baris pertama sampai dengan ketiga bertujuan untuk membuat obyek a,b dan c yang berisi vector dengan tipe data skalar yang berbeda. Sedangkan baris keempat adalah langkah untuk membuat obyek dengan nama data_frame sebagai data frame dengan isi vector a, b dan c. terakhir adalah memberikan nama kolom, kolom pertama akan memiliki nama ID, kolom kedua bernama Color dan kolom ketiga memiliki nama Passed. Berikut ini adalah isi dari data frame data_frame.

```
> data_frame
#> # A tibble: 4 x 3
#>   ID Color Passed
#>   <dbl> <chr>   <lgl>
#> 1     1 green    TRUE
#> 2     2 white    TRUE
#> 3     3 red     FALSE
#> 4     4 blue    FALSE
```

Selanjutnya adalah akan dicontohkan bagaimana mengakses elemen pada data frame. Jika ingin mengakses elemen baris ke 4 dan kolom pertama maka bisa dilakukan dengan dua cara yaitu:

```
> data_frame[4,1]
[1] 4

> data_frame[4,"ID"]
[1] 4
```

Sedangkan jika ingin mengakses seluruh nilai elemen kolom kedua dapat dilakukan dengan kedua cara berikut ini.

```
> data_frame[,2]
[1] "green" "white" "red"    "blue"

> data_frame[,"Color"]
[1] "green" "white" "red"    "blue"
```

Sedangkan untuk mengubah nilai suatu elemen, misal elemen baris ke-1 dan kolom ke-1, maka dapat dilakukan dengan cara seperti berikut.

```
data_frame[1,1] = 5
data_frame[1,"ID"] = 5
```

List

Fungsi list() dapat digunakan untuk membuat obyek yang di dalamnya berisi banyak objek dengan tipe data yang beragam. Sebagai contoh di dalam sebuah obyek yang dibentuk dari fungsi list() dapat berisi atas tipe data skalar, vector, matrix dan lain-lain. Berikut adalah contoh dari obyek yang dibentuk dari fungsi list().

```
mahasiswa = list(name="Adi", mynumbers=a, mymatrix=y, umur=21)
```

Sebagaimana yang telah dibahas pada bagian sebelumnya, pembahasan tentang data frame, telah dibuat vector a. Dan pada pembahasan bagian tentang matrix telah dibuat matrix y. dari contoh kode di atas dapat diketahui bahwa list() dapat digunakan untuk menampung beragam tipe data.

Di bawah ini dapat dilihat isi dari obyek mahasiswa.

```
> mahasiswa
$name
[1] "Adi"

$mynumbers
[1] 1 2 3 4

$mymatrix
     [,1] [,2] [,3] [,4]
[1,]     1     6    11    16
[2,]     2     7    12    17
[3,]     3     8    13    18
[4,]     4     9    14    19
[5,]     5    10    15    20

$umur
[1] 21
```

Sedangkan cara untuk mengakses isi dari data yang disimpan di dalam list dapat digunakan dengan sintaks seperti berikut ini.

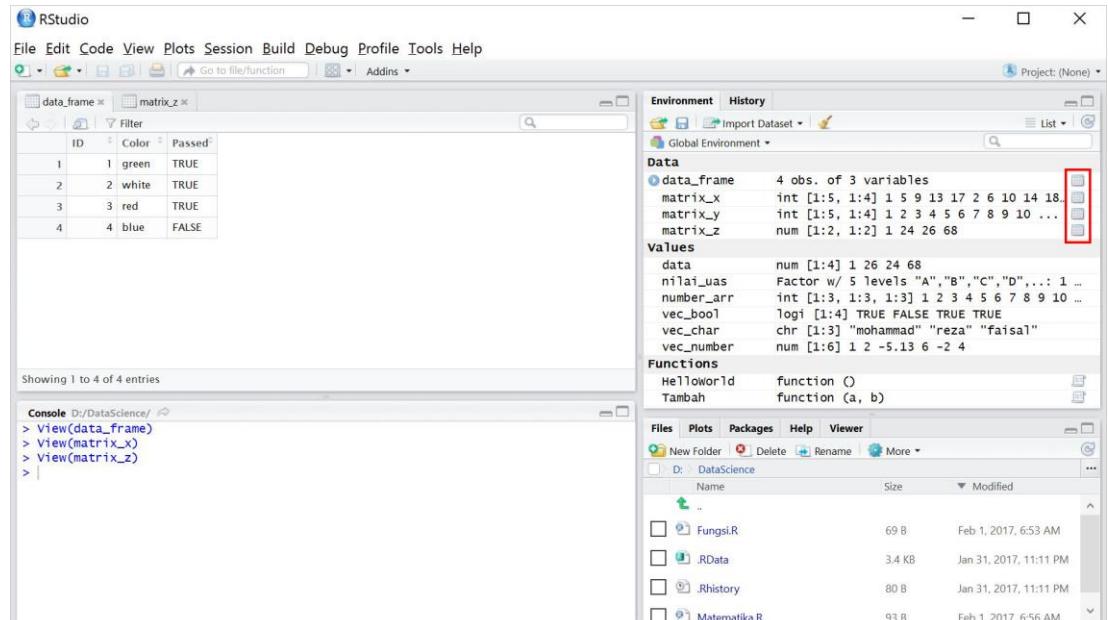
```
nama_objek[[index]]
```

Jika ingin mengakses nilai pertama pada obyek mahasiswa maka digunakan kode seperti berikut.

```
> mahasiswa[[1]]  
[1] "Adi"
```

Data Pada RStudio

Dari tipe data yang telah disebutkan di atas, hanya matrix dan data frame yang bisa dikategorikan sebagai data. Pada gambar di bawah dapat dilihat obyek bertipe data matrix yaitu matrix_x, matrix_y dan matrix_z berada di dalam kategori Data pada tab Environment. Begitu juga obyek bertipe data data frame yaitu data_frame.



Gambar 39. RStudio - Data.

R memiliki fungsi yang dapat digunakan untuk melihat data yaitu View(). Pada gambar di atas dapat dilihat contoh penggunaan fungsi View(). Pada RStudio memiliki kemudahan untuk memanggil fungsi ini, yaitu dengan mengklik tombol pada kotak merah. Tombol tersedia di setiap data. Misal diklik tombol pada obyek data_frame, maka secara otomatis akan dijalankan fungsi berikut.

```
View(data_frame)
```

Hasilnya akan ditampilkan oleh RStudio berupa tabel seperti pada gambar di atas.

Percabangan

Pada bagian ini akan diberikan penjelasan tentang percabangan pada bahasa R.

Operator

Untuk melakukan percabangan pada bahasa R perlu digunakan operator pembanding dan operator logika. Operator ini merupakan operator umum yang dijumpai pada bahasa pemrograman lain.

Berikut ini adalah daftar operator pembanding:

Tanda	Fungsi
==	Sama dengan
!=	Tidak sama dengan
>	Lebih besar
>=	Lebih besar atau sama dengan
<	Lebih kecil
<=	Lebih kecil atau sama dengan

Sedangkan operator logika dapat dilihat pada tabel di bawah ini.

Tanda	Fungsi
&	and
	or
!	negasi

Statement if

if adalah statement yang umum digunakan pada percabangan. Berikut ini adalah sintaks dari statement ini.

```
if(kondisi) {  
    ...  
}
```

Jika "kondisi" bernilai benar maka "fungsi" akan dijalankan, jika "kondisi" salah maka "fungsi" tidak akan dijalankan.

Sebagai contoh:

```
> if(5 > 4){print("5 lebih besar daripada 4")}  
[1] "5 lebih besar daripada 4"
```

Karena kondisi $5 > 4$ adalah benar maka fungsi print() di dalam tanda {} dieksekusi.

Sintaks penggunaan statement ini yang lebih rumit dapat dilihat di bawah ini.

```
if(kondisi) {  
    ...  
} else {  
    ...  
}
```

Atau sintaks berikut ini.

```
if(kondisi 1) {  
    ...  
} else if(kondisi 2) {  
    ...  
} else {  
    ...  
}
```

Untuk mencoba sintaks di atas maka dibuat fungsi sederhana yang disimpan pada file HitungNilai.R dengan kode di bawah ini.

```
HitungNilai.R
HitungNilai <- function(nilai) {
  if(nilai > 100 | nilai < 0) {
    sprintf("Nilai harus 0-100")
  } else {
    if(nilai >= 80) {
      sprintf("A")
    } else if (nilai >= 70) {
      sprintf("B")
    } else if (nilai >= 60) {
      sprintf("C")
    } else if (nilai >= 50) {
      sprintf("D")
    } else {
      sprintf("E")
    }
  }
}
```

Hasilnya dapat dilihat di bawah ini.

```
> HitungNilai(110)
[1] "Nilai harus 0-100"

> HitungNilai(-1)
[1] "Nilai harus 0-100"

> HitungNilai(80)
[1] "A"

> HitungNilai(70)
[1] "B"

> HitungNilai(60)
[1] "C"

> HitungNilai(50)
[1] "D"

> HitungNilai(30)
[1] "E"
```

Pada contoh kode HitungNilai.R di atas telah digunakan operator logika dan operator pembanding.

Statement ifelse

Statement ini merupakan percabangan yang merupakan bentuk ringkas dari sintaks if-else di bawah ini.

```
if(kondisi) {
  ...
} else {
  ...
}
```

Dengan statement ifelse maka sintaksnya menjadi seperti berikut.

```
if(kondisi, dijalankan jika benar, dijalankan jika salah)
```

Contoh penggunaan perintah ini adalah sebagai berikut.

```
> ifelse(TRUE, sprintf("benar"), sprintf("salah"))
[1] "benar"
```

```
> ifelse(FALSE, sprintf("benar"), sprintf("salah"))
[1] "salah"
```

Statement switch

Sintaks untuk statement switch adalah sebagai berikut.

```
switch (statement, list)
```

Parameter statement dapat berupa tipe data numerik, string atau karakter. Sedangkan parameter list berisi nilai-nilai yang akan ditampilkan sesuai dengan nilai yang diberikan pada parameter statement.

Untuk parameter statement berupa numerik dapat dilihat contohnya di bawah ini.

```
> switch(2,"merah","kuning","biru")
[1] "kuning"
```

Pada contoh di atas, nilai untuk parameter statement adalah 2. Sedangkan parameter list berisi nilai-nilai sebagai berikut:

1. merah.
2. kuning.
3. biru.

Karena diberikan nilai 2 sebagai nilai parameter maka output dari contoh di atas adalah "kuning".

Contoh berikutnya adalah jika tipe data parameter statement adalah karakter.

```
> switch("B","B" = "Benar","S" = "Salah")
[1] "Benar"
```

Output dari contoh di atas adalah "Benar".

Sedangkan contoh yang lain dapat dilihat di bawah ini.

```
> switch("panjang", "warna"="merah", "bentuk"="persegi", "panjang"=5)
[1] 5
```

Output dari contoh di atas adalah 5.

Pengulangan

Pada bagian ini akan dijelaskan penggunaan statement for dan while untuk melakukan pengulangan.

Statement for

Berikut ini adalah sintaks dari statement for.

```
for(obyek in sequence) {
  ...
}
```

Jika dilihat dari sintaks di atas maka sintaks tersebut lebih mirip dengan statement foreach yang ada pada bahasa pemrograman lain seperti C#.

Berikut adalah contoh penggunaan statement ini.

```
> for(i in 1:5) {print(i)}
[1] 1
[1] 2
```

```
[1] 3  
[1] 4  
[1] 5
```

Dari kode di atas dilakukan pengulangan 1:5 (1 sampai 5) dan nilai itu akan disimpan pada obyek i setiap langkah pengulangan dilakukan.

Contoh lain adalah menggunakan pengulangan untuk membaca isi dari vector yang dibuat dengan cara di bawah ini.

```
> data = 1:10  
> for(i in data) {print(data[i])}  
[1] 1  
[1] 2  
[1] 3  
[1] 4  
[1] 5  
[1] 6  
[1] 7  
[1] 8  
[1] 9  
[1] 10
```

Statement while

Cara lain untuk melakukan pengulangan adalah dengan menggunakan statement while. Sintaks dari statement ini adalah sebagai berikut.

```
inisialisasi nilai obyek  
while (kondisi) {  
  ...  
  mengubah nilai obyek  
}
```

Contoh penggunaan statement ini adalah seperti dicontohkan pada kode di bawah ini.

```
i = 1;  
while(i <= 5){  
  print(i);  
  i = i+1;  
}
```

Hasil dari kode di atas dapat dilihat di bawah ini.

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

Berikut ini adalah contoh penggunaan pengulangan while() untuk mengakses data pada vector.

```
data_mahasiswa = c("mohammad", "reza", "faisal", "budi", "iwan", "wati")  
  
i=1  
  
while(i <= 6){  
  print(data_mahasiswa[i])  
  i = i+1  
}
```

Dan berikut adalah hasil dari kode di atas.

```
[1] "mohammad"  
[1] "reza"  
[1] "faisal"  
[1] "budi"
```

```
[1] "iwan"  
[1] "wati"
```

Statement repeat

Sintaks untuk statement repeat adalah sebagai berikut.

```
repeat {  
  ...  
}
```

Berbeda dengan bahasa pemrograman seperti Pascal, statement repeat pada R tidak memiliki kondisi. Untuk itu perlu dibuat kondisi di dalam badan tanda { }. Berikut ini adalah contoh dari penggunaan statement repeat.

```
i = 1  
repeat{  
  print(i)  
  if(i > 20) {break()  
  i = i + 3  
}
```

Dan hasilnya dapat dilihat di bawah ini.

```
[1] 1  
[1] 4  
[1] 7  
[1] 10  
[1] 13  
[1] 16  
[1] 19  
[1] 22
```

Statement next

Statement next berfungsi untuk melewati sebuah iterasi pada suatu pengulangan. Berikut adalah contoh penggunaan statement next.

```
for(i in 1:10) {  
  if(i <= 5) {  
    next  
  }  
  
  print(i)  
}
```

Hasilnya dapat dilihat di bawah ini.

```
[1] 6  
[1] 7  
[1] 8  
[1] 9  
[1] 10
```

Dari contoh di atas dapat dilihat, jika nilai i lebih kecil dan sama dengan 5 maka proses iterasi akan dilewati sehingga fungsi print(i) tidak dieksekusi. Setelah nilai i = 6 maka proses iterasi berjalan normal kembali.

Statement break

Statement break untuk menghentikan proses pengulangan jika suatu kondisi dipenuhi. Berikut adalah contoh penggunaan statement break.

```
for(i in 1:10) {  
  if(i >5) {  
    break
```

```
    }
    print(i)
}
```

Hasilnya dapat dilihat di bawah ini.

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

Dari contoh di atas dapat dilihat, jika kondisi i lebih besar dari 5 maka proses pengulangan dihentikan.

Penanganan Kesalahan

Penanganan kesalahan atau lebih dikenal dengan istilah exception handling. Hal ini digunakan agar ketika terjadi kesalahan pada program ketika dijalankan tidak menghentikan jalannya program tetapi kesalahan akan ditangani dan program tetap dijalankan sampai selesai.

Fungsi yang digunakan untuk proses ini adalah tryCatch(). Berikut adalah sintaks fungsi ini.

```
tryCatch({
  # kode program
}, warning = function(w) {
  # kode untuk menangani ketika mendapatkan warning
}, error = function(e) {
  # kode untuk menangani ketika mendapatkan error
}, finally = {
  # kode yang akan dijalankan setelah seluruh kode di atas di jalankan
})
```

Dari sintaks di atas dapat dilihat terdapat 4 blok, yaitu:

- Kode program.
- Warning.
- Error.
- Finally.

Dalam implementasinya keempat blok ini tidak perlu digunakan semuanya cukup menggunakan 2 blok saja yaitu:

- Kode program.
- Error.

Sehingga dapat ditulis sintaksnya sebagai berikut.

```
tryCatch({
  # kode program
}, error = function(e) {
  # kode untuk menangani ketika mendapatkan error
})
```

Dan berikut ini adalah contoh sederhana dari penggunaan fungsi tryCatch().

```
tryCatch({
  print("kode program")
}, warning = function(w) {
  print("ada warning")
}, error = function(e) {
  print("ada error")
}, finally = {
```

```
    print("kode final dijalankan")
})
```

Hasil dari contoh kode di atas adalah sebagai berikut.

```
[1] "kode program"
[1] "hello world"
```

Dari contoh di atas dapat dilihat walau tidak terjadi warning atau error, kode yang berada pada blok finally akan tetap dieksekusi.

Masih kosong

<http://mazamascience.com/WorkingWithData/?p=912>

Komentar

Komentar adalah baris yang tidak dianggap dijalankan sebagai kode program atau fungsi. Komentar sering digunakan untuk memberikan keterangan terhadap apa yang ditulis pada program. Selain itu komentar juga dapat digunakan untuk “menon-aktifkan” suatu baris agar tidak dieksekusi.

Untuk membuat komentar cukup gunakan karakter # pada posisi awal baris, seperti yang terlihat pada contoh di bawah ini.

```
# Fungsi tanpa mengembalikan keluaran
HelloRepeat = function(jumlah) {
  data = 1:jumlah
  for(i in data) {
    print(data[i])
  }
}

# Fungsi yang mengembalikan keluaran
Tambah = function(bilangan1, bilangan2) {
  hasil = bilangan1 + bilangan2;
  return(hasil);
}

HelloRepeat(6)

#hasil_jumlah = Tambah(5, 6)
#hasil_jumlah
```

Pada dua baris terakhir dapat dilihat fungsi komentar agar kedua baris tersebut tidak dijalankan.

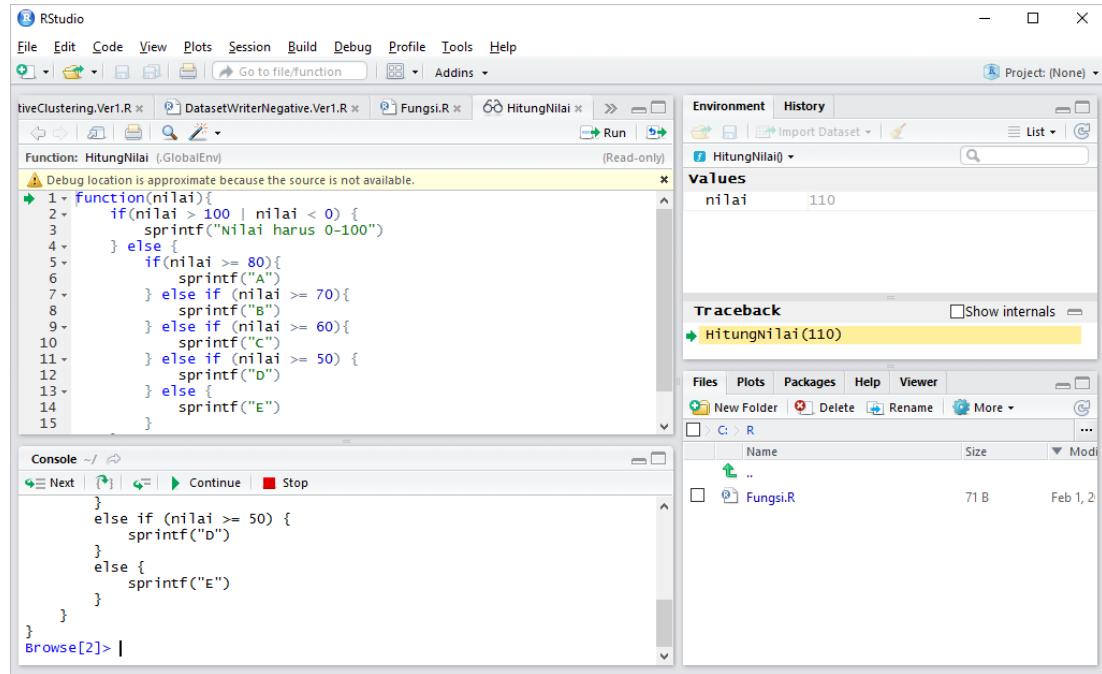
Debug

Proses debug memungkinkan programmer melacak kesalahan dari kode yang sedang dijalankan. Jika baris kode yang ditulis berjumlah banyak, maka proses debug akan membantu menampilkan baris mana yang membuat kesalahan pertama kali. Proses debug pada R dilakukan dengan bantuan fungsi debug().

Misal ingin mendebug fungsi HitungNilai yang telah dibuat pada sub bab Percabangan > Statement if. Untuk memulai debug fungsi HitungNilai digunakan cara sebagai berikut.

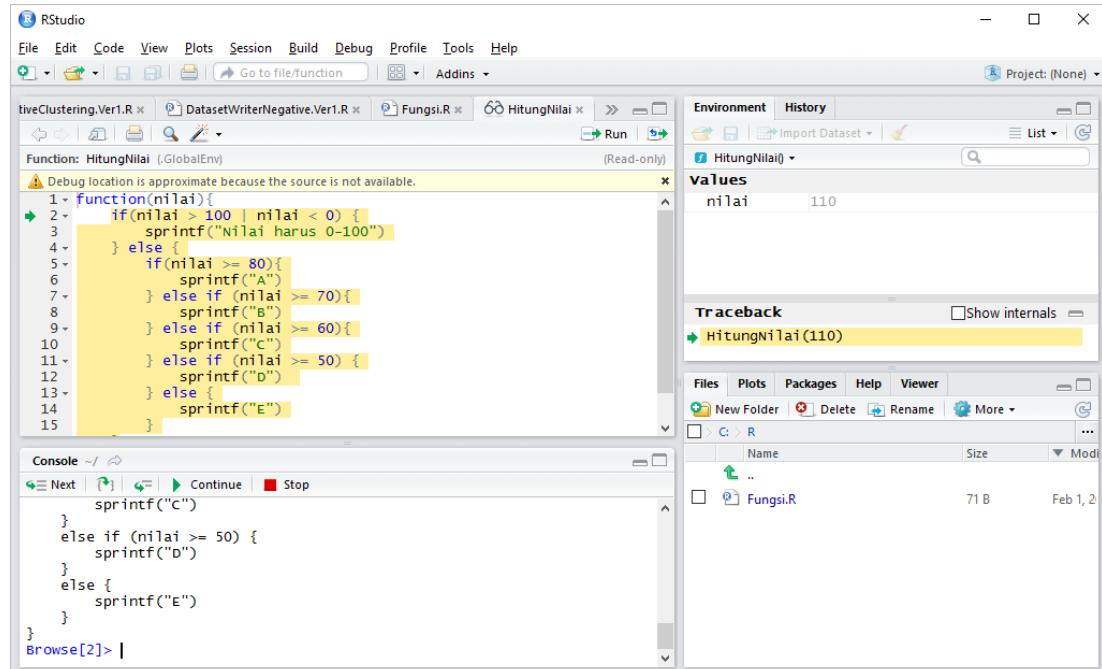
```
debug(HitungNilai)
```

Kemudian dapat dilihat RStudio akan menampilkan bantuan proses debug seperti pada gambar di bawah ini.



Gambar 40. RStudio – Memulai proses debug.

Dengan mengklik tombol Next maka akan dieksekusi baris atau langkah berikutnya sesuai dengan alur algoritma. Pada gambar di bawah ini dapat dilihat dilakukan eksekusi baris kedua.



Gambar 41. RStudio – Melanjutkan proses debug dengan mengklik tombol Next.

Setelah proses debug selesai maka akan kembali ke prompt normal. Untuk menghentikan mode debug maka dapat diketik fungsi berikut ini.

```
undebug(HitungNilai)
```

Jika hal ini tidak dilakukan, maka proses debug terhadap fungsi HitungNilai akan dilakukan kembali saat fungsi ini dieksekusi.

Profiling

Secara sederhana profiling berarti cara untuk mengetahui waktu yang digunakan oleh program. Fungsi yang dapat digunakan untuk mengetahui waktu yang digunakan suatu program adalah system.time().

Sintaks dari fungsi ini adalah sebagai berikut:

```
System.time()  
  #kode ditulis di sini  
)
```

Berikut adalah contoh penggunaan fungsi ini.

```
print(system.time({  
hasil_i = 0  
hasil_j = 0  
  
for(i in 1:1000){  
  for (j in 1:25000) {  
    hasil_j = hasil_j + 1  
  }  
  
  hasil_i = hasil_i + i  
}  
}))
```

Hasilnya adalah sebagai berikut.

```
user  system elapsed  
8.72      0.00     8.73
```

Dari ketiga kolom waktu di atas, waktu total yang digunakan dapat dilihat pada kolom elapsed. Waktu ini juga yang sesuai perhitungan saat program dijalankan sampai berakhir.

Fungsi system.time() juga dapat digunakan untuk melihat waktu yang diperlukan untuk satu iterasi pada pengulangan. Contoh kasus ini dapat dilihat pada kode berikut:

```
print(system.time({  
hasil_i = 0  
hasil_j = 0  
  
#pengulangan utama  
for(i in 1:5){  
  print(system.time({  
    #pengulangan kedua  
    for (j in 1:1000000) {  
      hasil_j = hasil_j + 1  
    }  
  }))  
  
  hasil_i = hasil_i + i  
}  
print("-----")  
}))
```

Output dari contoh di atas adalah sebagai berikut. 5 output pertama adalah waktu yang pergunakan oleh setiap iterasi di dalam blok pengulangan utama.

```
user  system elapsed  
0.34      0.00     0.34  
  
user  system elapsed  
0.36      0.00     0.36
```

```

user    system elapsed
0.35      0.00   0.35

user    system elapsed
0.36      0.00   0.36

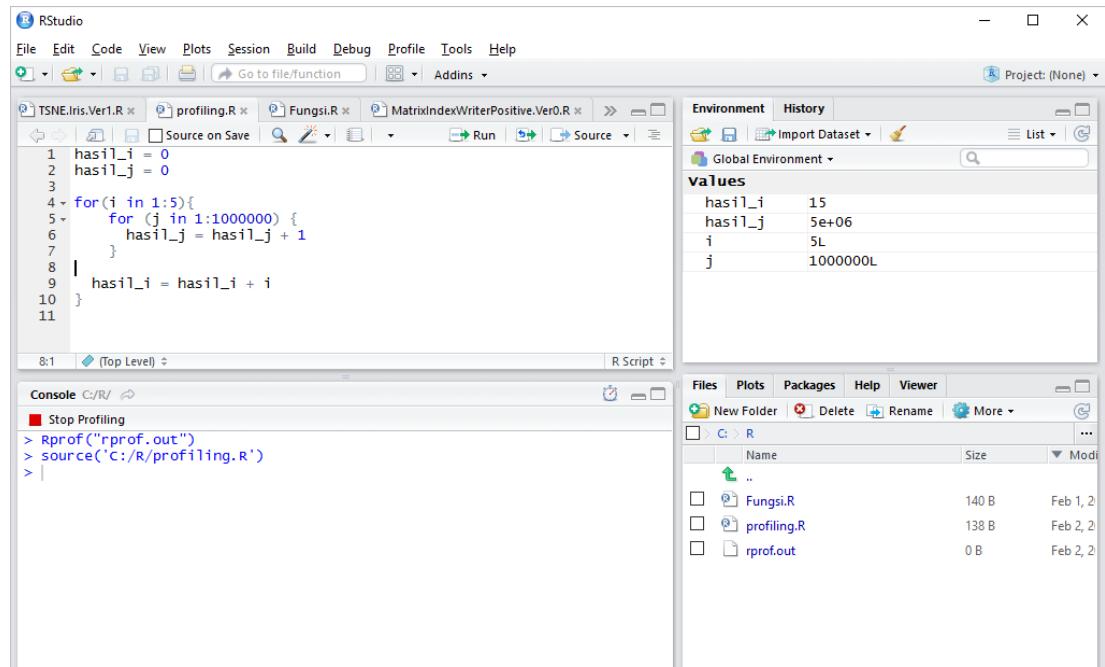
user    system elapsed
0.34      0.00   0.34
[1] "-----"
user    system elapsed
1.95      0.00   1.95

```

Sedangkan output terakhir adalah waktu yang dipergunakan untuk menjalankan seluruh program.

Fungsi lain yang dapat digunakan untuk melakukan profiling adalah Rprof() dan summaryRprof(). Langkah pertama mengetikkan fungsi Rprof() untuk memulai proses profiling seperti pada gambar di bawah ini.

Rprof()



Gambar 42. RStudio – Profilling dimulai.

Kemudian jalankan program. Setelah itu jalankan fungsi berikut ini untuk menghentikan proses profiling.

Rprof(NULL)

Untuk melihat detail laporan profiling dapat digunakan fungsi berikut ini.

```

> summaryRprof("rprof.out")
$by.self
      self.time self.pct total.time total.pct
"eval"     1.76    84.62      2.08   100.00
"+"

|               | total.time | total.pct | self.time | self.pct |
|---------------|------------|-----------|-----------|----------|
| "eval"        | 2.08       | 100.00    | 1.76      | 84.62    |
| "source"      | 2.08       | 100.00    | 0.00      | 0.00     |
| "withVisible" | 2.08       | 100.00    | 0.00      | 0.00     |
| "+"           | 0.32       | 15.38     | 0.32      | 15.38    |


```

```
$sample.interval  
[1] 0.02  
  
$sampling.time  
[1] 2.08
```

3

Pemrograman Berbasis Obyek

Setelah belajar dasar-dasar pemrograman R selanjutnya akan dikenalkan pemrograman berbasis obyek atau Object Oriented Programming (OOP) dengan menggunakan R. Hal ini perlu diperkenalkan agar pembaca dapat mengerti ketika bertemu contoh-contoh kode program yang ditulis dengan R yang menggunakan konsep OOP ini, sebagai contoh adalah jika contoh kode program pada bidang Bioinformatics.

Lingkungan R mendukung dua sistem untuk OOP yaitu S3 dan yang lebih baru adalah S4. Pada sistem S4 telah mendukung multiple inheritance, multiple dispatch dan introspection.

Class

Untuk membuat class pada lingkungan R digunakan fungsi `setClass()`. Seperti pada fungsi pada umumnya, didalamnya dapat berisi parameter-parameter. Berikut adalah contoh penggunaan fungsi tersebut.

```
y = matrix(1:50, 10, 5)

setClass(Class="ProsesMatrix",
         representation=representation(a="ANY"),
         prototype=prototype(a=y[1:2,]),
         validity=function(object) {
           if(class(object@a)!="matrix") {
             return(paste("nilai yang digunakan harus matrix, data yg anda gunakan
adalah", class(object@a)))
           } else {
             return(TRUE)
           }
         }
       )
```

Keterangan parameter-parameter yang dimiliki oleh fungsi `setClass()`:

1. Class: nama class.
2. representation: slot yang harus dimiliki oleh class baru atau class turunannya.
3. prototype: sebuah obyek menyediakan data default untuk slot.
4. contains: class-class induk.
5. validity, access, version: control argument.
6. where: lingkungan untuk menggunakan, menyimpan atau menghapus definisi sebagai meta data.

Untuk membuat inisialisasi method yang akan dipanggil saat class diinstansiasi dapat digunakan dengan lain seperti berikut ini.

```
setMethod("initialize", "ProsesMatrix", function(.Object, a) {
  .Object@a <- a/a
  .Object
})
```

Obyek

Untuk membuat objek dari class yang telah dibuat digunakan fungsi new(). Berikut ini adalah contoh untuk membuat objek dari class ProsesMatrix yang telah dibuat pada bagian sebelumnya.

```
prosesMatrixObj = new("ProsesMatrix", a=y)
```

Jika objek prosesMatrixObj dipanggil akan dapat dilihat output seperti berikut.

```
> prosesMatrixObj
An object of class "ProsesMatrix"
Slot "a":
 [,1] [,2] [,3] [,4] [,5]
 [1,]    1   11   21   31   41
 [2,]    2   12   22   32   42
 [3,]    3   13   23   33   43
 [4,]    4   14   24   34   44
 [5,]    5   15   25   35   45
 [6,]    6   16   26   36   46
 [7,]    7   17   27   37   47
 [8,]    8   18   28   38   48
 [9,]    9   19   29   39   49
[10,]   10   20   30   40   50
```

Berikut ini akan dicontohkan jika class diberikan nilai dengan tipe bukan matrix. Pada contoh di bawah ini nilai yang diberikan bertipe data frame.

```
> prosesIris = new("ProsesMatrix", a=iris)
Error in validObject(.Object) :
  invalid class "ProsesMatrix" object: nilai yang digunakan harus matrix,
data yg anda gunakan adalah data.frame
```

Inheritance

Inheritance atau pewarisan adalah konsep pemrograman dimana sebuah class dapat menurunkan sifat-sifat (biasanya berupa pewarisan property dan method) yang dimilikinya kepada class lain. Pada lingkungan R, untuk mewarisi suatu class induk dapat dilakukan dengan menggunakan pengisian parameter contain pada fungsi setClass().

Misal dibuat dua buah class yaitu ClassInduk1 dan ClassInduk2 seperti berikut.

```
setClass("ClassInduk1", representation(a = "character", b = "character"))
setClass("ClassInduk2", representation(c = "numeric", d = "numeric"))
```

Jika ingin membuat class yang mewarisi sifat-sifat dari ClassInduk1 dan ClassInduk2 maka dapat dibuat class dengan cara seperti berikut ini.

```
setClass("ClassAnak", contains=c("ClassInduk1", "ClassInduk2"))
```

Untuk melihat sifat-sifat ClassInduk1 dan ClassInduk2 pada ClassAnak maka dapat dibuat objek dari ClassAnak sebagai berikut.

```
> new("ClassAnak", a=letters[1:4], b=letters[1:4], c=1:4, d=4:1)
An object of class "ClassAnak"
```

```

Slot "a":
[1] "a" "b" "c" "d"

Slot "b":
[1] "a" "b" "c" "d"

Slot "c":
[1] 1 2 3 4

Slot "d":
[1] 4 3 2 1

```

Dan jika dilihat obyek yang dibuat dari class ClassInduk1 maka akan dapat dilihat output seperti berikut.

```

> getClass("ClassInduk1")
Class "ClassInduk1" [in ".GlobalEnv"]

Slots:

Name:           a          b
Class: character character

Known Subclasses: "ClassAnak"

```

Sedangkan output dari obyek yang dibuat dari class ClassInduk2 adalah sebagai berikut.

```

> getClass("ClassInduk2")
Class "ClassInduk2" [in ".GlobalEnv"]

Slots:

Name:           c          d
Class: numeric numeric

Known Subclasses: "ClassAnak"

```

Pada contoh kode di atas diperkenalkan fungsi untuk memeriksa class, yaitu getClass(). Selain fungsi itu juga dapat digunakan fungsi lain untuk memeriksa class yaitu:

1. getSlots().
2. slotNames().
3. extends().

Method & Fungsi Generic

Untuk membuat method atau fungsi generic pada lingkungan R digunakan fungsi:

1. setMethod()
2. setGeneric().

Untuk membuat fungsi generic dapat dilakukan dengan dua cara. Yang pertama adalah dengan mengubah fungsi yang sudah ada menjadi fungsi generic dengan cara seperti berikut ini.

```

fungsiBiasa = function(object) 0
fungsiGeneric = setGeneric("fungsiBiasa")

```

Cara kedua adalah membuat fungsi generic sendiri dengan cara seperti berikut, misal dimiliki class seperti berikut.

```

setClass("Shape")
setClass("Polygon", representation(sides = "integer"), contains = "Shape")

```

Maka untuk membuat method untuk class tersebut dapat dilakukan dengan cara seperti berikut.

```
setMethod("sides", signature(object = "Polygon"), function(object) {  
  object@sides  
})
```

Jika diperiksa dengan fungsi showMethods() seperti berikut di bawah ini, maka akan dapat dilihat obyek yang menggunakan method ini adalah obyek dari class Polygon.

```
> showMethods("sides")  
Function: sides (package .GlobalEnv)  
object="ANY"  
object="Polygon"
```

Fungsi-fungsi yang dapat digunakan untuk memeriksa method adalah sebagai berikut:

1. showMethods().
2. findMethods().
3. getMethod().
4. existsMethod().

4

Fungsi Dasar R

Dokumen Bantuan

Dokumen bantuan atau help adalah dokumen yang memberikan informasi detail tentang suatu fungsi atau hal-hal lain pada lingkungan R. Untuk informasi detail suatu fungsi dapat dilihat sintaks dari fungsi dan contoh-contoh cara penggunaannya.

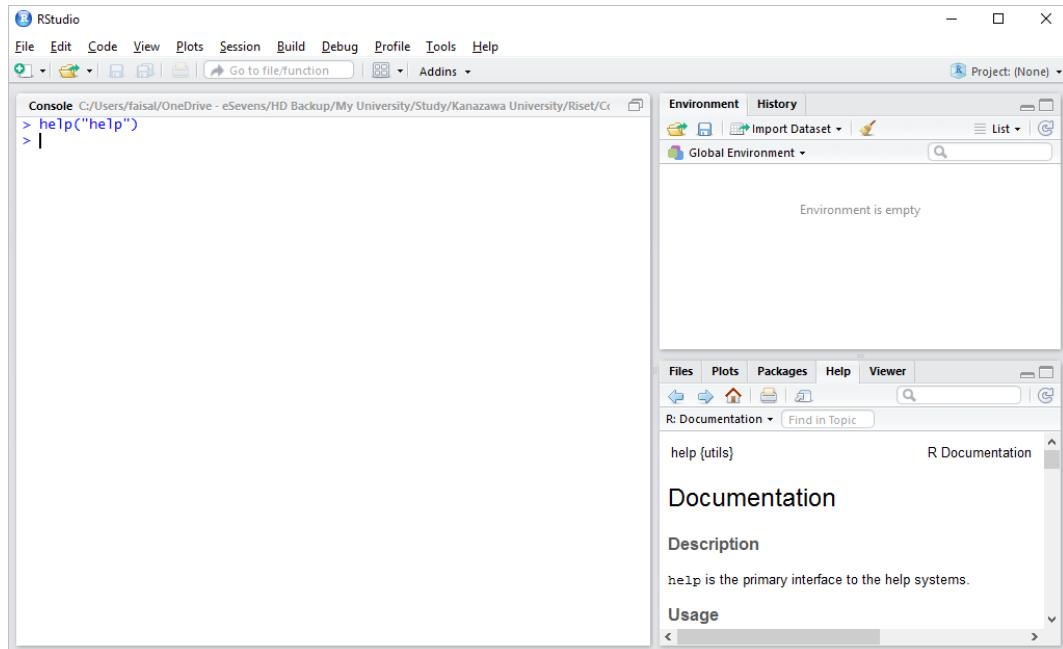
Untuk melihat dokumen bantuan digunakan fungsi `help()`. Sintaks fungsi ini adalah sebagai berikut.

```
help("nama fungsi")
```

Sebagai contoh untuk melihat dokumen bantuan fungsi `help()` dapat digunakan perintah berikut.

```
help("help")
```

Hasilnya dapat dilihat pada gambar di bawah ini. Dokumen akan ditampilkan pada tab Help.



Gambar 43. RStudio – Dokumen bantuan.

Package

Default lingkungan R telah memiliki banyak fungsi-fungsi yang dapat digunakan untuk berbagai keperluan. Lingkungan R dapat ditambahkan fungsi-fungsi baru. Fungsi-fungsi baru tersebut biasanya dikemas dalam bentuk package.

Installasi Package

Package-package tersebut disimpan pada server yang dapat diakses secara online. Proses installasi package hanya dapat dilakukan jika ada koneksi internet.

Fungsi `install.package()` adalah fungsi untuk mengunduh dan menginstall package. Sintaks fungsi ini adalah sebagai berikut.

```
install.packages("NamaPackage")
```

Kode di bawah ini adalah contoh untuk menginstall package “`kernlab`”.

```
install.packages("kernlab")
```

Gambar di bawah ini adalah informasi yang dilihat setelah proses installasi package selesai.

```
Console ~ / 
> install.packages("kernlab")
Installing package into 'C:/Users/M Reza Faisal/Documents/R/win-library/3.3'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.3/kernlab_0.9-25.zip'
Content type 'application/zip' length 2205869 bytes (2.1 MB)
downloaded 2.1 MB

package 'kernlab' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
  C:\Users\M Reza Faisal\AppData\Local\Temp\RtmpIhvzQT\downloaded_packages
> |
```

Gambar 44. RStudio – Installasi package.

Memuat Package

Fungsi-fungsi package hanya dapat digunakan jika package telah dimuat. Package harus dimuat setiap saat session baru dibuat atau setiap saat baru menjalankan tool pemrograman R.

Fungsi untuk memuat package adalah `library()`. Sintaks fungsi ini adalah sebagai berikut.

```
library(NamaPackage)

atau

library("NamaPackage")
```

Berikut adalah contoh kode untuk memuat package “`kernlab`”.

```
library("kernlab")
```

GitHub

Selain menggunakan fungsi-fungsi yang telah tersedia pada repository seperti CRAN (The Comprehensive R Archive Network), dapat juga digunakan package yang tersedia pada repository lain seperti GitHub.

Untuk menggunakan package yang tersedia pada GitHub terlebih dahulu telah terinstall package devtools. Untuk menginstall package ini digunakan perintah berikut.

```
install.packages("devtools")
```

Kemudian dapat digunakan perintah berikut ini.

```
library(devtools)
install_github('alamat_tujuan_pada_github')
```

Sebagai contoh sebagai berikut.

```
library(devtools)
install_github('ramhiser/datamicroarray')
```

Keterangan:

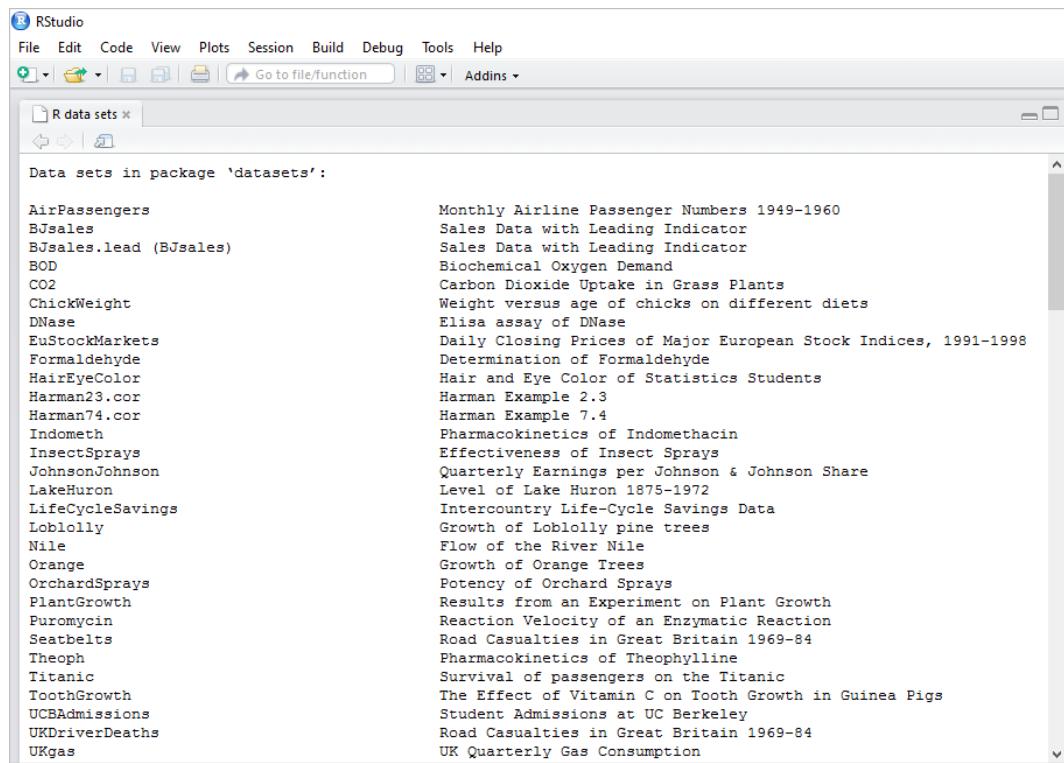
- ramhiser adalah nama user pada GitHub.
- datamicroarray adalah nama repository yang dimiliki oleh user tersebut. Nama ini juga bisa menjadi nama package.

Setelah proses installasi package berhasil, maka dapat dipanggil nama package tersebut dengan fungsi library(), sebagai contoh sebagai berikut ini.

```
library(datamicroarray)
```

Dataset

Platform R telah memiliki lebih 100 dataset yang dapat digunakan untuk latihan. Fungsi data() digunakan untuk melihat daftar dataset tersebut. Ketik fungsi data() pada window Console pada RStudio maka hasilnya dapat dilihat pada gambar di bawah ini.



Gambar 45. RStudio – Daftar dataset.

Salah satu dataset yang umum dan banyak digunakan untuk latihan adalah iris. Dataset ini menyimpan ukuran sepal dan petal dari bunga iris. Dataset ini terdiri atas 150 sample yang terbagi atas 3 class yaitu:

- 50 sample class setosa.
- 50 sample class versicolor.
- 50 sample class virginica.

Untuk melihat data iris dapat memanggil obyek iris pada R console.

```
iris
Atau:
data(iris)
```

Selain data di atas, dapat juga ditambahkan dataset lain dari package mlbench. Untuk menginstall package ini digunakan fungsi di bawah ini.

```
install.packages("mlbench")
```

Setelah proses installasi berhasil, maka untuk memuat package ini digunakan fungsi berikut.

```
library(mlbench)
```

Berikut adalah beberapa dataset dari package ini, yaitu:

- Ionosphere.
- LetterRecognition.
- Ozone.
- PimaIndiansDiabetes.

- Soybean.
- Vowel.
- Dan lain-lain.

Data di atas umum digunakan sebagai data benchmark pada kasus-kasus analitik data dan machine learning. Data tersebut juga dapat ditemui pada website berikut <https://archive.ics.uci.edu/ml/datasets.html>.

Menulis Data Ke File

Iris dataset akan digunakan pada contoh-contoh di sub bab ini.

write.csv()

Fungsi write.csv() untuk menulis file yang berisi nilai-nilai yang dipisahkan oleh koma. Format file yang dihasilkan adalah file text.

Sintaks fungsi ini adalah sebagai berikut.

```
write.csv(ObjectName, "FileName")
```

Untuk menyimpan dataset iris ke file data.csv maka digunakan perintah berikut ini.

```
write.csv(iris, "data.csv")
```

Perintah di atas akan menghasilkan file dengan isi sebagai berikut.

```
","", "Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width", "Species"
"1", 5.1, 3.5, 1.4, 0.2, "setosa"
"2", 4.9, 3, 1.4, 0.2, "setosa"
"3", 4.7, 3.2, 1.3, 0.2, "setosa"
```

Terdapat kolom yang tidak penting, yaitu kolom pertama yang berisi nomor row. Jika ingin menghilangkan kolom ini dapat digunakan tambahan opsi seperti berikut.

```
write.csv(iris, "data.csv", row.names = FALSE)
```

Hasilnya adalah sebagai berikut.

```
"Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width", "Species"
5.1, 3.5, 1.4, 0.2, "setosa"
4.9, 3, 1.4, 0.2, "setosa"
4.7, 3.2, 1.3, 0.2, "setosa"
```

Pada hasil di atas, nilai pada baris pertama dan kolom Species pada baris berikutnya menggunakan tanda quote karena kolom ini bernilai bukan angka. Jika ingin menyimpan seluruh nilai tanpa menggunakan tanda quote dapat digunakan opsi berikut ini.

```
write.csv(iris, "data.csv", row.names = FALSE, quote = FALSE)
```

Hasilnya adalah sebagai berikut. Dapat dilihat seluruh nilai yang disimpan pada file tidak menggunakan tanda quote.

```
Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, Species
5.1, 3.5, 1.4, 0.2, setosa
4.9, 3, 1.4, 0.2, setosa
4.7, 3.2, 1.3, 0.2, setosa
4.6, 3.1, 1.5, 0.2, setosa
```

write.table()

Fungsi write.table() untuk menulis file yang berisi nilai-nilai yang dipisahkan oleh tab. Sintaks fungsi ini adalah sebagai berikut.

```
write.table(ObjectName, "FileName")
```

Untuk menyimpan dataset iris ke file data.txt maka digunakan perintah berikut ini.

```
write.table(iris, "data.txt")
```

Seperti fungsi write.csv(), fungsi ini juga memiliki opsi untuk menghilangkan penggunaan tanda quote dan nomor baris.

```
write.table(iris, "data.txt", row.names = FALSE, quote = FALSE)
```

Membaca File Text

Format file yang umum digunakan digunakan menyimpan data adalah file text yang berisi nilai-nilai yang dipisahkan oleh tanda koma atau tab.

read.csv()

Jika file text berisi nilai-nilai yang dipisahkan oleh tanda koma maka dapat digunakan fungsi read.csv(). Sintaks fungsi ini adalah sebagai berikut.

```
ObjectName = read.csv("FileName")
```

Sebagai contoh, untuk membaca file data.csv digunakan kode berikut. Obyek data_csv akan menyimpan data file data_penelitian.csv yang dibaca oleh fungsi ini.

```
data_csv = read.csv("data.csv")
```

Fungsi ini juga dapat membaca file yang disimpan di komputer lain. Sebagai contoh file yang disimpan pada suatu server internet, maka dapat dilakukan pembacaan file dengan cara berikut ini.

```
data = read.csv("https://servername/filename.csv")
```

read.delim()

Fungsi read.delim() dapat digunakan membaca file text yang berisi nilai-nilai yang dipisahkan suatu karakter.

```
ObjectName = read.delim("FileName", sep = "char")
```

Sebagai contoh jika dimiliki file dengan nama data.txt dengan isi sebagai berikut.

```
5#1#1#1#2#1#3#1#1#2  
5#4#4#5#7#10#3#2#1#2
```

```
3#1#1#1#2#2#3#1#1#2  
6#8#8#1#3#4#3#7#1#2  
4#1#1#3#2#1#3#1#1#2
```

Maka untuk membaca file ini digunakan perintah sebagai berikut.

```
data_delim = read.delim("test.txt", sep = "#")
```

read.table()

Jika file text berisi nilai-nilai yang dipisahkan tab maka dapat digunakan fungsi `read.table()`. Sintaks fungsi ini adalah sebagai berikut.

```
ObjectName = read.table("FileName")
```

Sebagai contoh, untuk membaca file `data.txt` digunakan kode di bawah ini.

```
data_txt = read.table("data.txt")
```

Obyek `data_txt` akan menyimpan data file `data_penelitian.txt` yang dibaca oleh fungsi `read.table()`.

Membaca File Excel

File Excel juga umum digunakan untuk menyimpan data. Untuk membaca file Excel diperlukan package “`xlsx`”. Install package ini dengan perintah berikut.

```
install.packages("xlsx")
```

Setelah proses installasi package “`xlsx`” selesai, muat library ini dengan perintah di bawah ini.

```
library(xlsx)
```

Fungsi yang digunakan untuk membaca file Excel adalah sebagai berikut.

```
data_xlsx = read.xlsx("FileName", sheetName = "SheetName")
```

Sebagai contoh, untuk membaca file `data.xlsx` digunakan perintah berikut ini.

```
data_xlsx = read.xlsx("data.xlsx", sheetName = "Sheet1")
```

Obyek `data_xlsx` akan menyimpan data file `data.xlsx`.

Akses Database

Platform R juga dapat membaca data dari database server seperti MySQL dan SQL Server.

Akses Database MySQL

Untuk mengakses database MySQL pada platform R diperlukan package tambahan. Package yang dapat digunakan adalah `RMySQL`.

Untuk menginstall package RMySQL digunakan perintah berikut.

```
install.packages("RMySQL")
```

Kemudian muat library RMySQL dengan perintah berikut.

```
library(RMySQL)
```

Langkah pertama sebelum melakukan operasi database adalah melakukan koneksi ke database dengan fungsi dbConnect(). Untuk melakukan koneksi ke database digunakan sintaks berikut.

```
ObjectName = dbConnect(MySQL(), user='UserName', password='Password',
dbname='DatabaseName', host='HostName')
```

Sebagai contoh nama database adalah belajar_r, nama host adalah localhost, nama user adalah root dan password user adalah Rahasia. Maka perintah untuk melakukan koneksi adalah sebagai berikut.

```
mydb = dbConnect(MySQL(), user='root', password='Rahasia',
dbname='belajar_r', host='localhost')
```

Untuk melihat daftar tabel pada database belajar_r digunakan fungsi dbListTables(). Berikut adalah contoh untuk menampilkan daftar tabel.

```
> dbListTables(mydb)
[1] "iris"
```

Dari output perintah di atas, terdapat 1 tabel yaitu iris.

Untuk mengeksekusi SQL query dapat digunakan fungsi dbSendQuery(). Sebagai contoh, untuk menampilkan data dari tabel iris dapat digunakan kode di bawah ini.

```
rs = dbSendQuery(mydb, 'select * from iris')
data_iris = fetch(rs, n=-1)
```

Data dari tabel iris akan disimpan pada obyek data_iris. Untuk menampilkan isi data ke layar dapat dilakukan dengan mengetik obyek data_iris pada R console.

Akses Database SQL Server

Package tambahan yang digunakan untuk mengakses database SQL Server adalah RODBC. Kode di bawah ini untuk menginstall package RODBC.

```
install.packages("RODBC")
```

Untuk memuat library RODBC digunakan kode di bawah ini.

```
library(RODBC)
```

Untuk melakukan koneksi ke database digunakan fungsi odbcDriverConnect(). Sintaks fungsi odbcDriverConnect() adalah sebagai berikut.

```
ObjectName = odbcDriverConnect('driver={SQL
Server};server=HostName;database=DatabaseName;trusted connection=true')
```

Jika database server berada pada komputer yang sama dan nama database adalah BelajarR, maka digunakan kode berikut ini.

```
myodbc = odbcDriverConnect('driver={SQL
Server};server=.;database=BelajarR;trusted connection=true')
```

Untuk membaca data pada tabel iris pada database BelajarR digunakan fungsi sqlQuery(). Contoh kode yang dapat digunakan adalah sebagai berikut.

```
data_iris <- sqlQuery(myodbc, 'select * from iris')
```

Ketik obyek data_iris pada R console untuk melihat isi data tabel iris.

Dimensi Data

Umumnya data disimpan dalam bentuk tabel yang memiliki kolom dan baris. Jika suatu obyek digunakan untuk menyimpan data seperti itu maka obyek tersebut dapat ditulis dengan rumus seperti berikut.

```
Object[row, column]
```

Keterangan:

- Object adalah nama obyek.
- row adalah nomor baris.
- column adalah nomor kolom.

Selanjutnya jika pada buku ini menyebut data, maka yang dimaksud adalah obyek yang menyimpan data dua dimensi seperti penjelasan di atas.

nrow()

Fungsi nrow() digunakan untuk menghitung jumlah baris yang dimiliki oleh obyek yang menyimpan data. Sintaks fungsi ini adalah sebagai berikut.

```
nrow(object)
```

Sebagai contoh untuk menghitung jumlah baris pada obyek yang menyimpan data iris digunakan perintah berikut.

```
> nrow(iris)  
[1] 150
```

ncol()

Fungsi ncol() digunakan untuk menghitung jumlah kolom yang dimiliki oleh obyek yang menyimpan data. sintaks fungsi ini adalah sebagai berikut.

```
nrow(object)
```

Sebagai contoh untuk menghitung jumlah kolom pada obyek yang menyimpan data iris digunakan perintah berikut.

```
> nrow(iris)  
[1] 5
```

colnames()

Fungsi colnames() digunakan untuk melihat nama atau header pada suatu data. Misal dimiliki data sebagai berikut.

```
> iris_sample = iris[1:5,]  
> View(iris_sample)
```

Maka akan dapat ditampilkan data seperti pada gambar di bawah ini.

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

Gambar 46. View(iris_sample).

Dengan menggunakan fungsi colnames() akan menampilkan keluaran sebagai berikut.

```
> colnames(iris_sample)
[1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"
```

Jika ingin melihat nama kolom tertentu dapat digunakan cara berikut ini.

```
> colnames(iris_sample) [1]
[1] "Sepal.Length"

> colnames(iris_sample) [3]
[1] "Petal.Length"

> colnames(iris_sample) [5]
[1] "Species"
```

Fungsi colnames() juga dapat digunakan untuk menganti nama kolom tertentu. Sebagai contoh untuk mengganti nama kolom kelima digunakan perintah berikut.

```
> colnames(iris_sample) [5] = "Class"
> colnames(iris_sample)
[1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Class"
```

rownames()

Fungsi rownames() digunakan untuk melihat nama baris yang digunakan pada data. Berikut adalah contoh penggunaan fungsi ini.

```
> rownames(iris_sample)
[1] "1" "2" "3" "4" "5"
```

Sedangkan untuk melihat nama pada suatu baris digunakan cara berikut ini.

```
> rownames(iris_sample) [3]
[1] "3"
```

Umumnya nama baris menggunakan nomer urut. Untuk mengganti nama baris dapat dilakukan dengan cara sebagai berikut.

```
> rownames(iris_sample) [1] = "sample1"
> rownames(iris_sample) [2] = "sample2"

> rownames(iris_sample)
[1] "sample1" "sample2" "3" "4" "5"
```

Terkadang ada kemungkinan dimiliki data dengan nama baris yang sama. Kasus ini dapat terjadi ketika mengumpulkan data yang dilakukan oleh program yang dibuat. Jika menemui kasus tersebut dan ingin membuat nama baris menjadi urutan angka maka dapat digunakan kode pada contoh di bawah ini.

```
> rownames(iris_sample) = seq(nrow(iris_sample))

> rownames(iris_sample)
[1] "1" "2" "3" "4" "5"
```

Menampilkan Data

Cara untuk menampilkan data adalah memanggil obyek yang digunakan untuk menyimpan data. Sebagai contoh, cara ini adalah untuk menampilkan data yang disimpan pada obyek `data_txt`.

```
> data_txt = read.table("data.txt")
> data_txt
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
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
7          4.6         3.4         1.4         0.3   setosa
8          5.0         3.4         1.5         0.2   setosa
9          4.4         2.9         1.4         0.2   setosa
...
...
```

head()

Untuk menampilkan sejumlah data dari awal digunakan fungsi `head()`. Sintaks fungsi ini adalah sebagai berikut.

```
head(ObjekName, RowNumber)
```

Sebagai contoh, untuk menampilkan data dari obyek `data_txt` sebanyak 5 row digunakan perintah ini.

```
> head(iris, 5)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
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
```

tail()

Untuk menampilkan sejumlah data dari akhir digunakan fungsi `tail()`. Sintaks fungsi ini adalah sebagai berikut.

```
tail(ObjekName, RowNumber)
```

Sebagai contoh, untuk menampilkan data dari obyek `data_txt` sebanyak 5 row digunakan perintah ini.

```
> tail(iris, 5)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
146          6.7         3.0         5.2         2.3 virginica
147          6.3         2.5         5.0         1.9 virginica
148          6.5         3.0         5.2         2.0 virginica
149          6.2         3.4         5.4         2.3 virginica
150          5.9         3.0         5.1         1.8 virginica
```

sample()

Dataset iris yang digunakan pada contoh-contoh di atas terlihat tersusun berdasarkan kategori yaitu setosa, virginica dan versicolor. Jika diambil 10 data teratas maka hanya akan didapat 10 data class setosa. Jika ingin mengambil 10 data secara acak maka dapat digunakan bantuan fungsi sample(). Sintaks fungsi ini adalah sebagai berikut.

```
sample(x, size)
```

Keterangan:

- x adalah jumlah total data.
- size adalah jumlah sample.

Maka untuk mengambil 10 data iris secara acak dilakukan dengan cara sebagai berikut.

```
iris_sample = iris[sample(nrow(iris), 10), ]
```

Hasilnya dapat dilihat di bawah ini.

```
> iris_sample
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
79          6.0        2.9       4.5      1.5 versicolor
90          5.5        2.5       4.0      1.3 versicolor
141         6.7        3.1       5.6      2.4  virginica
56          5.7        2.8       4.5      1.3 versicolor
55          6.5        2.8       4.6      1.5 versicolor
11          5.4        3.7       1.5      0.2   setosa
49          5.3        3.7       1.5      0.2   setosa
26          5.0        3.0       1.6      0.2   setosa
47          5.1        3.8       1.6      0.2   setosa
95          5.6        2.7       4.2      1.3 versicolor
```

unique()

Fungsi ini digunakan untuk menghilangkan data yang sama atau data duplikat. Sebagai contoh jika dimiliki data sebagai berikut.

```
> x <- round(rnorm(20, 10, 5))
> x
[1]  6  8 10  2  8  8 11  9  7 11  8 18 16 12  7  9  7 10 15  8
```

Maka dengan fungsi unique() akan dihasilkan data seperti berikut.

```
> unique(x)
[1]  6  8 10  2 11  9  7 18 16 12 15
```

Contoh lain adalah jika menggunakan data iris. Data iris terdiri 150 sample.

```
iris_unique = unique(iris)
```

Hasilnya dapat dilihat terdapat 149 sample unik.

View()

Fungsi View() digunakan untuk menampilkan data dalam bentuk tabel. Untuk menampilkan data dari obyek data_txt, gunakan perintah di bawah ini.

```
View(data_txt)
```

Hasilnya dapat dilihat pada gambar-gambar di bawah ini.

The screenshot shows the RStudio interface with the 'View()' function open. The left pane displays a data frame 'data_txt' with 150 observations and 5 variables: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, and Species. The right pane shows the 'Environment' tab with variables 'hasil_i', 'hasil_j', 'i', and 'j'. Below the environment is a file browser showing files like 'Fungsi.R', 'profiling.R', 'rprof.out', and 'data.txt'. The bottom pane is the 'Console' where the command 'view(data_txt)' was run.

Gambar 47. RStudio – View().

obyek\$rowName

Pada gambar di atas dapat dilihat obyek data_txt memiliki 5 kolom. Nama kelima kolom itu adalah Sepal.Length, Sepal.Width, Petal.Length, Petal.Width dan Species. Cara untuk melihat seluruh nilai pada kolom Species dapat dilakukan dengan contoh di bawah ini.

```
data_txt$Species
```

Untuk menampilkan seluruh nilai pada kolom Sepal.Width digunakan contoh di bawah ini.

```
data_txt$Sepal.Width
```

Memfilter Data

obyek[,col1:col2]

Jika data pada obyek terdiri atas 5 kolom, maka untuk menampilkan kolom 1 sampai 4 saja digunakan perintah di bawah ini.

```
iris[,1:4]
```

Jika ingin menampilkan kolom 1, kolom 3 dan kolom 5 saja dapat digunakan perintah berikut ini.

```
iris[,c(1,3,5)]
```

obyek[row1:row2,]

Jika ingin menampilkan data pada obyek berdasarkan baris yang diinginkan, misal baris 13 sampai baris 23 maka digunakan perintah ini.

```
iris[13:23, ]
```

Jika ingin menampilkan baris 1, baris 3 dan baris 5 saja maka digunakan perintah berikut.

```
iris[c(1,3,5), ]
```

obyek[row1:row2, col1:col2]

Jika ingin menampilkan data pada obyek baris 13 sampai 23 saja dan kolom yang ditampilkan hanya kolom 1 sampai 4 saja maka digunakan perintah berikut ini.

```
iris[13:23, 1:4]
```

Jika ingin menampilkan data pada baris 1, 3 dan 5 saja dan kolom 1, 3 dan 5 saja maka digunakan perintah berikut ini.

```
iris[c(1,3,5), c(1,3,5)]
```

obyek[which(),]

Untuk memfilter data berdasarkan suatu nilai pada sebuah kolom dapat digunakan fungsi which(). Berikut ini adalah contoh menampilkan data jika kolom Species bernilai setosa.

```
iris[which(iris$Species == "setosa"), ]
```

Sorting Data

order

Untuk mengurutkan data berdasarkan suatu kolom dapat digunakan fungsi order(). Penggunaan fungsi ini mirip dengan penggunaan fungsi which() yang telah dibahas di sub bab sebelumnya. Data dapat diurutkan dari nilai terkecil sampai terbesar dan sebaliknya.

Untuk mengurutkan data dari nilai kolom Sepal.Length terkecil dapat dilakukan dengan cara berikut ini.

```
order(iris$Sepal.Length)
```

Outputnya adalah nomor baris seperti berikut ini.

[1]	14	9	39	43	42	4	7	23	48	3	30	12	13	25
[15]	31	46	2	10	35	38	58	107	5	8	26	27	36	41
[29]	44	50	61	94	1	18	20	22	24	40	45	47	99	28
[43]	29	33	60	49	6	11	17	21	32	85	34	37	54	81
[57]	82	90	91	65	67	70	89	95	122	16	19	56	80	96
[71]	97	100	114	15	68	83	93	102	115	143	62	71	150	63

```
[85] 79 84 86 120 139 64 72 74 92 128 135 69 98 127
[99] 149 57 73 88 101 104 124 134 137 147 52 75 112 116
[113] 129 133 138 55 105 111 117 148 59 76 66 78 87 109
[127] 125 141 145 146 77 113 144 53 121 140 142 51 103 110
[141] 126 130 108 131 106 118 119 123 136 132
```

Dengan memanfaatkan nomor baris ini maka dapat ditampilkan datanya dengan cara meletakkan fungsi `order()` di atas pada bagian baris, seperti pada kode di bawah ini.

```
iris[order(iris$Sepal.Length), ]
```

Hasilnya dapat dilihat pada output di bawah ini.

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
14	4.3	3.0	1.1	0.1	setosa
9	4.4	2.9	1.4	0.2	setosa
39	4.4	3.0	1.3	0.2	setosa
43	4.4	3.2	1.3	0.2	setosa
...					

Untuk mendapatkan urutan dari nilai terbesar cukup dengan menambahkan tanda – seperti pada contoh berikut ini.

```
order(-iris$Sepal.Length)
```

Sehingga jika kode di atas digunakan untuk menampilkan data iris yang terurut berdasarkan nilai kolom Sepal.Length terbesar maka kode yang digunakan adalah sebagai berikut.

```
head(iris[order(-iris$Sepal.Length), ])
```

Hasilnya adalah sebagai berikut.

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
132	7.9	3.8	6.4	2.0	virginica
118	7.7	3.8	6.7	2.2	virginica
119	7.7	2.6	6.9	2.3	virginica
123	7.7	2.8	6.7	2.0	virginica
...					

sort

Menggabung Data

rbind()

Fungsi `rbind()` berfungsi untuk menggabung data dari dua obyek atau lebih. Sintaks fungsi ini adalah sebagai berikut.

```
ObjectName = rbind(object1, object2, . . . , objectN)
```

Obyek-obyek yang akan digabung harus memiliki jumlah kolom yang sama. Misal obyek data1 adalah sebagai berikut.

```
> data1 = iris[5:10,]
> data1
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
5          5.0       3.6        1.4       0.2    setosa
6          5.4       3.9        1.7       0.4    setosa
7          4.6       3.4        1.4       0.3    setosa
```

8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

Dan misal obyek data2 adalah sebagai berikut.

> data2 = iris[130:135,]
> data2
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
130 7.2 3.0 5.8 1.6 virginica
131 7.4 2.8 6.1 1.9 virginica
132 7.9 3.8 6.4 2.0 virginica
133 6.4 2.8 5.6 2.2 virginica
134 6.3 2.8 5.1 1.5 virginica
135 6.1 2.6 5.6 1.4 virginica

Gabungan obyek data1 dan obyek data2 digabung dengan fungsi rbind(), dan akan disimpan pada obyek data_all.

> data_all = rbind(data1, data2)
> data_all
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
5 5.0 3.6 1.4 0.2 setosa
6 5.4 3.9 1.7 0.4 setosa
7 4.6 3.4 1.4 0.3 setosa
8 5.0 3.4 1.5 0.2 setosa
9 4.4 2.9 1.4 0.2 setosa
10 4.9 3.1 1.5 0.1 setosa
130 7.2 3.0 5.8 1.6 virginica
131 7.4 2.8 6.1 1.9 virginica
132 7.9 3.8 6.4 2.0 virginica
133 6.4 2.8 5.6 2.2 virginica
134 6.3 2.8 5.1 1.5 virginica
135 6.1 2.6 5.6 1.4 virginica

Secara default fungsi rbind() menghasilkan object bertipe matrix atau sesuai dengan tipe data asalnya, jika object-object yang digabung memiliki tipe data yang sama.

cbind()

Fungsi cbind() untuk mengabungkan data pada obyek-obyek yang memiliki jumlah baris yang sama. Misal obyek data1 berisi data seperti berikut.

> data1 = iris[1:5, 1]
> data1
[1] 5.1 4.9 4.7 4.6 5.0

Dan data2 berisi data sebagai berikut.

> data2 = iris[131:135, 1]
> data2
[1] 7.4 7.9 6.4 6.3 6.1

Obyek data_all berisi data gabungan data1 dan data2 yang digabung dengan fungsi cbind().

> data_all = cbind(data1, data2)
> data_all
data1 data2
[1,] 5.1 7.4
[2,] 4.9 7.9
[3,] 4.7 6.4
[4,] 4.6 6.3
[5,] 5.0 6.1

Selanjut dimiliki data3 sebagai berikut.

> data3 = iris[1:5, 5]
> data3
[1] setosa setosa setosa setosa setosa

Levels: setosa versicolor virginica

Dari output data3 di atas dapat dilihat jika data3 berisi nilai bertipe factor. Kemudian data3 digabung dengan data_all dengan menggunakan fungsi cbind().

```
> data_all = cbind(data_all, data3)
> data_all
  data1 data2 data3
[1,] 5.1 7.4 1
[2,] 4.9 7.9 1
[3,] 4.7 6.4 1
[4,] 4.6 6.3 1
[5,] 5.0 6.1 1
```

Hasilnya dapat dilihat jika nilai data3 pada kolom ketiga dikonversi menjadi numerik. Oleh karena itu tidak disarankan untuk menggunakan fungsi cbind() untuk menggabung data yang bertipe factor.

rbind.data.frame()

Fungsi rbind.data.frame() memiliki kegunaan yang sama dengan fungsi rbind(), bedanya fungsi ini menghasilkan object bertipe data frame. Berikut adalah contoh penggunaan fungsi ini.

```
> data1 = iris[5:10,]
> data2 = iris[130:135,]
> data_all = rbind.data.frame(data1, data2)
> data_all
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
5          5.0       3.6        1.4       0.2    setosa
6          5.4       3.9        1.7       0.4    setosa
7          4.6       3.4        1.4       0.3    setosa
8          5.0       3.4        1.5       0.2    setosa
9          4.4       2.9        1.4       0.2    setosa
10         4.9       3.1        1.5       0.1    setosa
130        7.2       3.0        5.8       1.6  virginica
131        7.4       2.8        6.1       1.9  virginica
132        7.9       3.8        6.4       2.0  virginica
133        6.4       2.8        5.6       2.2  virginica
134        6.3       2.8        5.1       1.5  virginica
135        6.1       2.6        5.6       1.4  virginica
```

cbind.data.frame()

Fungsi cbind.data.frame() mempunyai kegunaan yang mirip dengan fungsi cbind(), bedanya adalah fungsi ini tidak akan mengubah data yang bertipe factor menjadi numerik. Berikut adalah contoh penggunaan dan contoh outputnya.

```
> data_all = cbind.data.frame(data_all, data3)
> data_all
  data1 data2 data3
1  5.1 7.4 setosa
2  4.9 7.9 setosa
3  4.7 6.4 setosa
4  4.6 6.3 setosa
5  5.0 6.1 setosa
```

paste()

Fungsi paste() digunakan untuk mengabungkan string. Sebagai contoh dapat dilihat pada kode di bawah ini. Antara string-string yang digabungkan akan terdapat spasi.

```
> hasil = 13 + 666
```

```
> paste("Hasil penjumlahan:", hasil)
[1] "Hasil penjumlahan: 679"
```

Contoh yang lain adalah sebagai berikut.

```
> paste("Bilangan", c(1:5))
[1] "Bilangan 1" "Bilangan 2" "Bilangan 3" "Bilangan 4" "Bilangan 5"
```

paste0()

Fungsi paste0() mempunyai cara kerja yang sama dengan fungsi paste(), bedanya adalah tidak ada spasi antara string-string yang digabungkan. Berikut adalah contohnya.

```
> paste0("Bilangan", c(1:5))
[1] "Bilangan1" "Bilangan2" "Bilangan3" "Bilangan4" "Bilangan5"
```

Contoh yang lain adalah implementasi fungsi ini untuk mengganti nama kolom pada suatu data.

```
> colnames(iris) = paste0("V", c(1:ncol(iris)))

> head(iris)
  V1  V2  V3  V4      V5
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
```

Dapat dilihat nama kolom pada data di atas sesuai dengan nilai yang diberikan yaitu V1 sampai dengan V5. Contoh yang lain adalah mengganti nama baris pada data.

```
> rownames(iris) = paste0("C", c(1:nrow(iris)))

> head(iris)
  V1  V2  V3  V4      V5
C1 5.1 3.5 1.4 0.2 setosa
C2 4.9 3.0 1.4 0.2 setosa
C3 4.7 3.2 1.3 0.2 setosa
C4 4.6 3.1 1.5 0.2 setosa
C5 5.0 3.6 1.4 0.2 setosa
C6 5.4 3.9 1.7 0.4 setosa
```

Explorasi Data

Mengetahui informasi data yang akan diolah sangat diperlukan. Informasi itu dapat digunakan untuk menentukan strategi yang dipilih untuk mengolah data tersebut. Berikut ini adalah beberapa fungsi untuk mengetahui ukuran, struktur, atribut, summary dan lain-lain

dim()

Fungsi dim() dapat memberikan informasi ukuran dimensi data. Contoh fungsi dim() dapat dilihat di bawah ini.

```
> dim(iris)
[1] 150   5
```

Dari informasi di atas dapa diketahui iris dataset terdiri atas 150 baris dan 5 kolom.

names()

Fungsi names() memberikan informasi name kolom data. Contoh fungsi names() dapat dilihat di bawah ini.

```
> names(iris)
[1] "Sepal.Length" "Sepal.Width"   "Petal.Length" "Petal.Width"   "Species"
```

str()

Fungsi str() memberikan informasi struktur data. Kode di bawah ini adalah contoh penggunaan fungsi ini.

```
> str(iris)
'data.frame': 150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species      : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 ...
```

Output fungsi ini memberikan informasi yang lebih lengkap dibandingkan fungsi dim() dan names(). Fungsi ini memberikan informasi tambahan berupa tipe data yang digunakan pada setiap kolom.

table()

Fungsi table() mempunyai banyak manfaat. Salah satunya berguna untuk melihat jumlah sample atau frekuensi pada masing-masing target variable. Sintaks untuk menggunakan fungsi ini adalah sebagai berikut.

```
table(ObjectName[,ColumnNumber])
atau
table(ObjectName$ColumnName)
```

Contoh di bawah ini untuk melihat jumlah sample berdasarkan kolom Species.

```
> table(iris[,5])
  setosa versicolor  virginica
      50        50        50
```

Atau dapat juga digunakan seperti contoh berikut ini.

```
> table(iris$Species)
  setosa versicolor  virginica
      50        50        50
```

summary()

Fungsi summary() memberikan informasi statistik ringkas dari data yang dimiliki sebuah obyek. Di bawah ini adalah contoh penggunaan fungsi ini dan outputnya.

```
> summary(iris)
  Sepal.Length   Sepal.Width    Petal.Length   Petal.Width      Species
  Min.    :4.300  Min.    :2.000   Min.    :1.000   Min.    :0.100  setosa
  :50
  1st Qu.:5.100  1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  versicolor:50
  Median :5.800  Median :3.000   Median :4.350   Median :1.300  virginica
  :50
  Mean    :5.843  Mean    :3.057   Mean    :3.758   Mean    :1.199
  3rd Qu.:6.400  3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
  Max.    :7.900  Max.    :4.400   Max.    :6.900   Max.    :2.500
```

Fungsi ini memberikan informasi statistik untuk setiap feature seperti nilai terkecil, nilai terbesar, median, mean dan kuartil.

Pengulangan

Pada bab sebelumnya telah diberikan penjelasan untuk melakukan pengulangan dengan for, while dan lain-lain. Kemudahan lain pada R adalah berupa fungsi untuk melakukan pengulangan.

lapply()

Fungsi lapply() akan melakukan pengulangan untuk data bertipe list atau vector. Sintaks fungsi ini adalah sebagai berikut.

```
lapply(X, FUNCTION, ...)
```

Keterangan:

- X adalah data bertipe list atau vector.
- FUNCTION adalah fungsi yang akan dijalankan untuk memproses setiap elemen data X.

Misal dimiliki obyek sebagai berikut.

```
a = 1:10  
b = 11:20  
c = 50:100
```

Kemudian ketiga obyek tersebut disimpan dalam obyek bertipe list.

```
data_list = list(a, b, c)
```

Dapat dilihat isi obyek data_list sebagai berikut.

```
> data_list  
[[1]]  
[1] 1 2 3 4 5 6 7 8 9 10  
  
[[2]]  
[1] 11 12 13 14 15 16 17 18 19 20  
  
[[3]]  
[1] 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66  
[6] 67 68 69 70 71 72 73 74 75 76  
[28] 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93  
[34] 94 95 96 97 98 99 100
```

Selanjutnya akan digunakan fungsi lapply() untuk menghitung rata-rata dengan fungsi mean() untuk setiap data a, b dan c yang berada di dalam list.

```
> lapply(data_list, mean)  
[[1]]  
[1] 5.5  
  
[[2]]  
[1] 15.5  
  
[[3]]  
[1] 75
```

Dari contoh di atas dapat dilihat hasil perhitungan rata-rata untuk masing-masing obyek.

- Rata-rata obyek a = 5.5.
- Rata-rata obyek b = 15.5.
- Rata-rata obyek c = 75.

Selain menggunakan fungsi R, juga dapat digunakan fungsi yang dibuat oleh user. Sebagai contoh dibuat fungsi sebagai berikut.

```
hitungNilaiAkhir = function(nilai_ujian) {  
  if(mean(nilai_ujian) > 60) {  
    return("lulus")  
  } else {  
    return("tidak lulus")  
  }  
}
```

Kemudian dimiliki data hasil ujian dari siswa-siswa seperti berikut.

```
adi = c(70, 80, 90)  
ida = c(50, 60, 70)  
data_siswa = list(adi, ida)
```

Selanjutnya fungsi lapply() digunakan seperti contoh kode di bawah ini.

```
> lapply(data_siswa, hitungNilaiAkhir)  
[[1]]  
[1] "lulus"  
  
[[2]]  
[1] "tidak lulus"
```

Contoh berikut adalah menggunakan data yang lebih sederhana.

```
plusTwo = function(x) {  
  x = x + 2  
}  
  
lapply(iris[1:5,1], plusTwo)
```

Hasilnya adalah menambahkan nilai 5 nilai dari kolom 1 data iris dengan 2. Sehingga didapat hasil seperti berikut ini.

```
> lapply(iris[1:5,1], plusTwo)  
[[1]]  
[1] 7.1  
  
[[2]]  
[1] 6.9  
  
[[3]]  
[1] 6.7  
  
[[4]]  
[1] 6.6  
  
[[5]]  
[1] 7
```

sapply()

Fungsi sapply() sama seperti fungsi lapply() dengan output yang lebih sederhana. Sintaks fungsi ini adalah sebagai berikut.

```
sapply(X, FUNCTION, ...)
```

Dan berikut adalah contoh penggunaan fungsi ini.

```
> sapply(data_list, mean)
[1] 5.5 15.5 75.0

> sapply(data_siswa, hitungNilaiAkhir)
[1] "lulus"
[1] "tidak lulus"
```

Contoh lain adalah sebagai berikut.

```
> sapply(iris[1:5,], plusTwo)
[1] 7.1 6.9 6.7 6.6 7.0
```

colSums()

Fungsi colSums() digunakan untuk menjumlahkan nilai dataset berdasarkan kolom. Sebagai contoh adalah sebagai berikut.

```
> colSums(iris[,1:4])
Sepal.Length Sepal.Width Petal.Length Petal.Width
876.5       458.6      563.7      179.9
```

Dari hasil di atas dapat dilihat hasil penjumlahan nilai untuk setiap kolom dataset iris.

colMeans()

Fungsi colMeans() digunakan untuk menghitung rata-rata nilai dataset berdasarkan kolom. Berikut adalah contoh penggunaan fungsi ini.

```
> colMeans(iris[,1:4])
Sepal.Length Sepal.Width Petal.Length Petal.Width
5.843333   3.057333   3.758000   1.199333
```

rowSums()

Fungsi rowSums() digunakan untuk menghitung jumlah nilai dataset berdasarkan row. Berikut adalah contoh penggunaan fungsi ini.

```
rowSums(iris[,1:4])
```

rowMeans()

Fungsi rowMeans() digunakan untuk menghitung rata-rata nilai dataset berdasarkan row. Berikut adalah contoh penggunaan fungsi ini.

```
rowMeans(iris[,1:4])
```

Membuat Data Simulasi

Dalam suatu percobaan yang kerap digunakan data buatan. Untuk membuat data yang akan digunakan sebagai simulasi maka R menyediakan beberapa fungsi untuk digunakan.

rnorm()

Fungsi rnorm() digunakan untuk membuat data acak yang terdistribusi normal. Sintaks fungsi ini adalah sebagai berikut:

```
rnorm(n, mean = 0, sd = 1)
```

Keterangan:

- n adalah jumlah bilangan yang akan dihasilkan.
- mean adalah rata-rata, secara default akan digunakan nilai 0.
- sd adalah deviasi standar, secara default akan digunakan nilai 1.

Berikut adalah contoh penggunaan fungsi ini.

```
> rnorm(5)  
[1] 0.1718101 0.1910118 1.0932906 0.1499124 0.8744652
```

dnorm()

Fungsi dnorm() membuat nilai distribusi probabilitas pada setiap titik untuk nilai mean dan deviasi standar yang diberikan.

Berikut ini adalah contoh data yang digunakan.

```
x = seq(-10, 10, by = .1)
```

Kemudian menentukan nilai y dengan menggunakan fungsi dnorm() seperti berikut.

```
y = dnorm(x, mean = 2.5, sd = 0.5)
```

pnorm()

Fungsi pnorm() memberikan informasi probabilitas dari distribusi normal bilangan acak kemudian dilakukan evaluasi pada nilai x untuk melihat apakah probabilitas nilai y akan mendapatkan nilai lebih kecil atau sama dengan x.

Data yang digunakan adalah sebagai berikut.

```
x = seq(-10,10,by = .2)
```

Sedangkan untuk data pada sumbu y didapat dengan menggunakan fungsi pnorm().

```
y = pnorm(x, mean = 2.5, sd = 2)
```

qnorm()

Fungsi ini mengambil nilai probabilitas dan memberikan bilangan yang nilai komultaif sesuai dengan nilai probabilitas.

Berikut ini adalah kode yang menggunakan fungsi qnorm().

```
x = seq(0, 1, by = 0.02)  
y = qnorm(x, mean = 2, sd = 1)
```

set.seed()

Untuk mengerti gunanya fungsi set.seed() maka terlebih dahulu akan dicontohkan keluaran yang dihasilkan oleh fungsi rnorm() di bawah ini.

```
> rnorm(5)  
[1] 0.7079547 -0.2396980 1.9844739 -0.1387870 0.4176508
```

```
> rnorm(5)
[1] 0.9817528 -0.3926954 -1.0396690 1.7822290 -2.3110691
```

Pada contoh di atas dapat dilihat keluaran dari fungsi `rnorm()` yang dilakukan sebanyak 2 kali. Keduanya memberikan keluaran yang berbeda. Jika ingin mengatur keluaran fungsi random maka fungsi `set.seed()` dapat digunakan untuk hal tersebut.

Berikut adalah pengaruh fungsi `set.seed()` terhadap keluaran fungsi `rnorm()`.

```
> set.seed(1)
> rnorm(5)
[1] -0.6264538 0.1836433 -0.8356286 1.5952808 0.3295078

> set.seed(1)
> rnorm(5)
[1] -0.6264538 0.1836433 -0.8356286 1.5952808 0.3295078
```

Contoh yang lain adalah sebagai berikut.

```
> set.seed(10)
> rnorm(5)
[1] 0.01874617 -0.18425254 -1.37133055 -0.59916772 0.29454513

> set.seed(10)
> rnorm(5)
[1] 0.01874617 -0.18425254 -1.37133055 -0.59916772 0.29454513
```

Dari kedua contoh di atas maka dapat dilihat pengaruh fungsi `set.seed()`.

Mengelola Obyek

Dari contoh-contoh di atas pembuatan obyek dapat dilakukan dengan mudah. Sebagai contoh adalah seperti di bawah ini.

```
kota = "banjarmasin"
random data = iris[sample(nrow(iris), 5), ]
```

assign()

Selain dengan cara di atas, pembuatan obyek dapat dilakukan dengan menggunakan fungsi `assign()`. Dengan sintaks sebagai berikut.

```
assign("ObjectName", value)
```

Keterangan:

- `ObjectName` adalah string dari nama obyek.
- `value` adalah nilai dari obyek.

Berikut adalah dua contoh penggunaan fungsi ini.

```
assign("kota", "banjarmasin")
assign("random data", iris[sample(nrow(iris), 5), ])
```

get()

Fungsi `get()` dapat digunakan untuk mengambil nilai suatu object dengan cara memasukkan string nama object sebagai input pada fungsi `get()`. Berikut adalah contoh penggunaan fungsi ini.

```
> bilangan1 = 5
> get("bilangan1")
[1] 5

> assign("bilangan2", 13)
> get("bilangan2")
[1] 13
```

ls()

Fungsi ls() digunakan untuk melihat daftar obyek yang ada pada session yang sedang aktif. Berikut adalah contoh penggunaan fungsi ini dan keluarannya.

```
> ls()
[1] "kota"           "random_data"
```

exists()

Fungsi exists() digunakan untuk memeriksa keberadaan sebuah obyek atau lebih. Sebagai contoh penggunaan fungsi ini.

```
> exists("kota")
[1] TRUE

> exists("desa")
[1] FALSE
```

Fungsi ini juga dapat memeriksa lebih dari satu obyek dengan memanfaatkan fungsi c() seperti contoh di bawah ini.

```
> exists(c("kota", "random_data"))
[1] TRUE

> exists(c("kota", "random_data", "desa"))
[1] TRUE

> exists(c("dusun", "desa"))
[1] FALSE
```

rm()

Fungsi rm() digunakan untuk menghapus obyek dari session yang sedang aktif. Sintaks fungsi ini adalah sebagai berikut.

```
rm(ObjectName) atau rm("ObjectName")
```

Sebagai contoh untuk menghapus obyek kota digunakan perintah sebagai berikut.

```
> rm(kota) #atau rm("kota")

> exists("kota")
[1] FALSE
```

Setelah obyek kota dihapus, maka jika dicek keberadaannya dengan fungsi exists() dapat dilihat obyek kota sudah tidak ada lagi.

Jika ingin menghapus seluruh obyek pada session yang sedang aktif dapat digunakan gabungan fungsi rm() dan ls() seperti berikut ini.

```
rm(list = ls())
```

Parameter list dapat berisi kumpulan nama obyek dalam bentuk karakter. Pada contoh di atas parameter list diisi oleh daftar seluruh obyek yang didapat dari keluaran fungsi ls().

save()

Fungsi save() dapat digunakan untuk menyimpan obyek R ke dalam file dengan format binary. Untuk memperlihatkan cara kerja fungsi ini maka dapat dikuti langkah-langkah berikut.

Pertama akan dibuat obyek yang menyimpan sebagian data iris.

```
iris_sample = iris[1:5, ]
```

Selanjutnya akan digunakan fungsi save() untuk menyimpan obyek iris_sample ke dalam file sample.RData dengan kode sebagai berikut.

```
save(iris_sample, file = "sample.RData")
```

Hasilnya dapat dilihat pada working directory, terdapat tambahan file sample.RData. Fungsi save() tidak hanya dapat digunakan untuk menyimpan sebuah obyek, tapi juga dapat digunakan untuk menyimpan lebih dari satu obyek. Berikut adalah contohnya.

```
x <- stats:::runif(23)
y <- list(a = 1, b = TRUE, c = "banjarmasin")
save(x, y, file = "xy.RData")
```

Selain untuk menyimpan obyek data seperti contoh-contoh di atas, juga dapat menyimpan obyek fungsi. Sebagai contoh jika dimiliki fungsi berikut.

```
HelloWorld = function() {
  print("Hello world...")
}
```

Kemudian fungsi tersebut dimuat ke session dengan menjalankan kode di atas pada R Console. Selanjutnya obyek fungsi tersebut dapat disimpan dengan cara berikut.

```
save(HelloWorld, file = "function.RData")
```

Seluruh jenis obyek pada lingkungan R dapat disimpan ke dalam file, sehingga jika dimiliki obyek dari hasil pembelajaran algoritma klasifikasi maka obyek itu pun dapat disimpan ke dalam file .RData. Sehingga obyek tersebut dapat dimuat kembali jika akan digunakan untuk melakukan prediksi kembali class dari sample baru.

Selanjutnya untuk memuat file .RData akan digunakan fungsi load() yang dijelaskan pada sub bab selanjutnya.

load()

Fungsi load() digunakan untuk memuat obyek-obyek yang disimpan dalam sebuah .RData. Untuk memperlihatkan obyek benar dapat dimuat ke session yang aktif. Maka untuk percobaan, terlebih dahulu semua obyek pada session akan dihapus.

```
> rm(list = ls())
> ls()
character(0)
```

Dengan melihat keluaran fungsi ls() di atas, dapat dilihat sudah tidak ada lagi obyek pada session. Selanjutnya akan digunakan fungsi load() untuk memuat file sample.RData yang telah dibuat sebelumnya.

```
> load("sample.RData")
> load("function.RData")
```

```
> ls()
[1] "HelloWorld"  "iris sample"
```

Dari keluar di atas dapat dilihat kedua obyek yang disimpan dari file sample.RData dan function.RData dimuat kembali ke session yang sedang aktif.

Memeriksa Obyek

class()

Fungsi class() dapat digunakan untuk memeriksa tipe data dari suatu obyek. Sintaks dari fungsi ini adalah sebagai berikut.

```
class(ObjectName)
```

Sebagai contoh untuk memeriksa tipe data dari obyek yang menyimpan dataset iris dapat digunakan perintah berikut ini. Dari keluarannya dapat dilihat bahwa tipe data obyek iris adalah data.frame.

```
> class(iris)
[1] "data.frame"
```

Contoh lain kode berikut ini.

```
> kota = "banjarmasin"
> class(kota)
[1] "character"
```

is.null()

Fungsi is.null() digunakan untuk memeriksa apakah suatu obyek berisi suatu nilai atau NULL. Sebagai contoh dibuat obyek seperti contoh di bawah ini.

```
> bilangan = NULL
> is.null(bilangan)
[1] TRUE
```

Dapat dilihat ketika diperiksa dengan fungsi ini didapat keluaran adalah TRUE, artinya obyek memang NULL. Dan ketika bilangan diberikan nilai.

```
> bilangan = 1
> is.null(bilangan)
[1] FALSE
```

is.na()

NA adalah Not Available. Jika suatu obyek memiliki nilai ini maka obyek tersebut dianggap kehilangan elemen. Fungsi is.na() digunakan untuk memeriksa hal tersebut. Obyek yang kehilangan elemen, akan memberikan efek yang tidak bagus pada suatu proses analisis data. Dengan mengetahui informasi ini maka dapat dilakukan langkah untuk mengganti nilai yang hilang pada obyek tersebut. Atau menghapus obyek tersebut jika obyek tersebut terlalu banyak kehilangan elemen.

Sebagai contoh data digunakan data sebagian data iris dan akan diberikan NA pada beberapa elemen pada obyek tersebut.

```

> iris_sample = iris[1:5,]
> iris_sample[1,1] = NA

> iris_sample[1,1]
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          NA         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

```

Sekarang dapat dilihat elemen pada baris 1 dan kolom 1 kehilangan nilai. Penggunaan fungsi `is.na()` dapat dilakukan dengan cara sebagai berikut.

```

> is.na(iris_sample[1,1])
[1] TRUE

```

Fungsi ini juga dapat memeriksa seluruh nilai pada elemen pada obyek `iris_sample` dengan cara berikut ini.

```

> is.na(iris_sample)
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1      TRUE      FALSE      FALSE      FALSE      FALSE
2      FALSE      FALSE      FALSE      FALSE      FALSE
3      FALSE      FALSE      FALSE      FALSE      FALSE
4      FALSE      FALSE      FALSE      FALSE      FALSE
5      FALSE      FALSE      FALSE      FALSE      FALSE

```

Untuk mendapatkan rangkuman informasi di atas dapat digunakan fungsi `table()` seperti berikut ini.

```

> table(is.na(iris_sample))
FALSE  TRUE
24     1

```

is.numeric()

Fungsi `is.numeric()` digunakan untuk memeriksa apakah suatu obyek atau elemen pada obyek bernilai numerik. Berikut adalah contoh penggunaan fungsi ini.

```

> bilangan1 = 1
> bilangan2 = "2"

> is.numeric(bilangan1)
[1] TRUE

> is.numeric(bilangan2)
[1] FALSE

```

Selain ketiga fungsi di atas, terdapat fungsi-fungsi lain untuk memeriksa tipe data dari suatu obyek, yaitu:

- `is.character()`.
- `is.logical()`.
- `is.data.frame()`.
- `is.matrix()`.
- `is.array()`.
- `is.list()`.
- Dan lain-lain.

Konversi Tipe Data

Konversi tipe data digunakan untuk mengubah tipe data dari suatu obyek.

as.character()

Fungsi as.character() digunakan untuk mengubah tipe data pada suatu obyek menjadi karakter atau string. Berikut ini adalah contoh penggunaan fungsi ini.

```
> bilangan = 6666
> bilangan
[1] 6666

> bilangan = as.character(bilangan)
> bilangan
[1] "6666"
```

Dari contoh di atas dapat dilihat konversi bilangan ke karakter dilakukan dengan menambahkan tanda quote.

Jika fungsi ini digunakan untuk mengkonversi obyek yang menyimpan data iris, maka proses konversi akan dilakukan pada nilai-nilai pada setiap kolom dan baris. Sebagai contoh akan dikonversi baris pertama data iris.

```
> iris[1,]
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5         1.4         0.2   setosa
```

Berikut adalah contoh keluaran setelah data di atas dikonversi dengan fungsi ini.

```
> as.character(iris[1,])
[1] "5.1" "3.5" "1.4" "0.2" "1"
```

Seperti halnya contoh sebelumnya, nilai-nilai ditambahkan dengan tanda quote. Kemudian karena tipe data kolom Species adalah factor, maka ketika dikonversi maka yang digunakan adalah angka factornya.

as.numeric()

Fungsi as.numeric() digunakan untuk mengkonversi tipe data suatu obyek menjadi numerik.

Berikut adalah contoh penggunaan fungsi ini.

```
> bil1 = "1"
> bil2 = "2"
> hasil = bil1 + bil2
Error in bil1 + bil2 : non-numeric argument to binary operator
```

Karena obyek bil1 dan bila2 menyimpan data karakter, maka terlebih dahulu obyek tersebut dikonversi menjadi numerik jika ingin melakukan operasi matematika.

```
> bil1 = as.numeric(bil1)
> bil2 = as.numeric(bil2)
> hasil = bil1 + bil2
> hasil
[1] 3
```

as.data.frame()

Fungsi as.data.frame() digunakan untuk mengubah nilai pada suatu obyek menjadi tipe data frame. Sebagai contoh akan digunakan data Titanic. Tapi terlebih dahulu dapat diperiksa tipe data dan dimensi dari obyek yang menyimpan obyek Titanic dengan kode di bawah ini.

```
> class(Titanic)
```

```

[1] "table"

> dim(Titanic)
[1] 4 2 2 2

> Titanic
, , Age = Child, Survived = No

      Sex
Class Male Female
  1st    0     0
  2nd    0     0
  3rd   35    17
Crew    0     0

, , Age = Adult, Survived = No

      Sex
Class Male Female
  1st  118      4
  2nd  154     13
  3rd  387     89
Crew  670      3

, , Age = Child, Survived = Yes

      Sex
Class Male Female
  1st    5     1
  2nd   11    13
  3rd   13    14
Crew    0     0

, , Age = Adult, Survived = Yes

      Sex
Class Male Female
  1st   57   140
  2nd   14    80
  3rd   75    76
Crew  192   20

```

Dan berikut adalah contoh penggunaan fungsi `as.data.frame()`.

```

> titanic_data_frame = as.data.frame(Titanic)

> titanic_data_frame
  Class   Sex   Age Survived Freq
1  1st  Male Child     No     0
2  2nd  Male Child     No     0
3  3rd  Male Child     No    35
4 Crew  Male Child     No     0
5  1st Female Child   No     0
6  2nd Female Child   No     0
7  3rd Female Child   No    17
8 Crew  Female Child  No     0
9  1st  Male Adult   No   118
10 2nd  Male Adult   No   154
11 3rd  Male Adult   No   387
12 Crew  Male Adult   No   670
13 1st Female Adult  No     4
14 2nd Female Adult  No    13
15 3rd Female Adult  No    89
16 Crew  Female Adult No     3
17 1st  Male Child   Yes     5
18 2nd  Male Child   Yes    11
19 3rd  Male Child   Yes    13

```

20	Crew	Male	Child	Yes	0
21	1st	Female	Child	Yes	1
22	2nd	Female	Child	Yes	13
23	3rd	Female	Child	Yes	14
24	Crew	Female	Child	Yes	0
25	1st	Male	Adult	Yes	57
26	2nd	Male	Adult	Yes	14
27	3rd	Male	Adult	Yes	75
28	Crew	Male	Adult	Yes	192
29	1st	Female	Adult	Yes	140
30	2nd	Female	Adult	Yes	80
31	3rd	Female	Adult	Yes	76
32	Crew	Female	Adult	Yes	20

Dengan mengubah menjadi data frame, maka data Titanic menjadi lebih mudah untuk dibaca.

Selain ketiga fungsi di atas, masih banyak lagi fungsi lain untuk melakukan konversi tipe data, yaitu:

- as.list().
- as.matrix().
- as.logical().
- as.factor().
- as.vector().
- as.array().
- Dan lain-lain.

5

Visualisasi Data

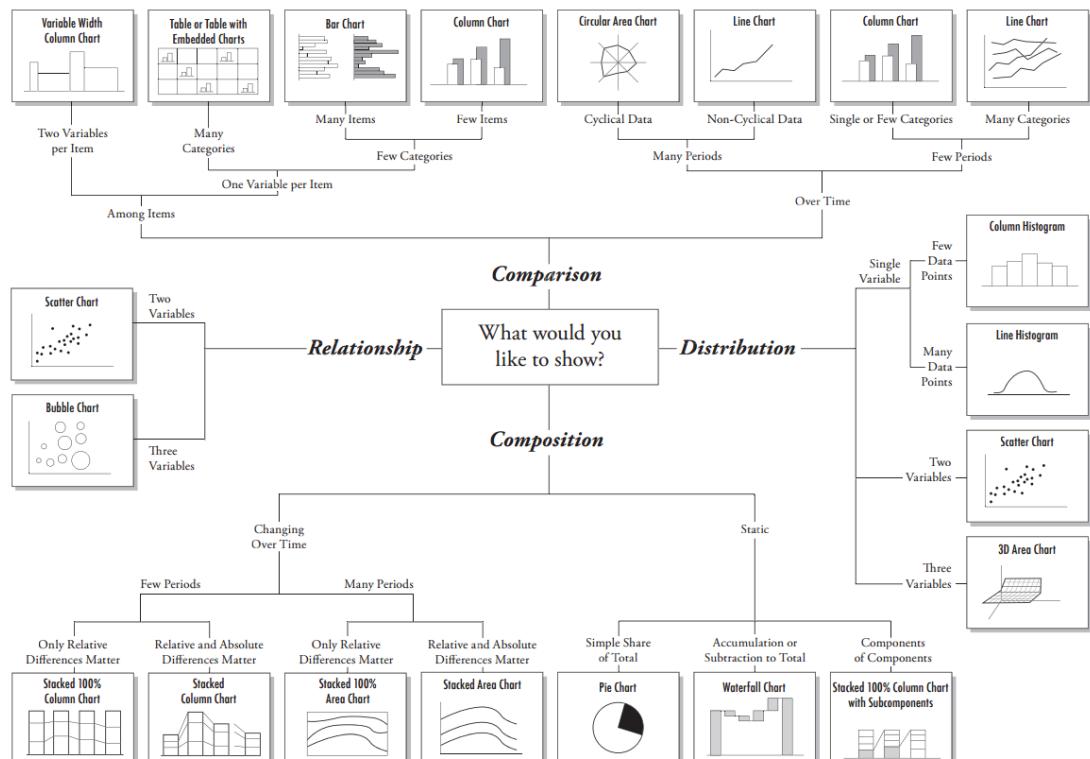
Visualisasi data adalah cara untuk mengubah angka menjadi informasi yang berguna dan mudah dimengerti oleh orang awam. Ada 4 hal yang umum dilakukan pada proses visualisasi data, yaitu:

- Perbandingan.
- Komposisi.
- Distribusi.
- Relasi atau hubungan.

Untuk pemilihan chart dapat ditentukan oleh setelah menjawab beberapa pertanyaan di bawah ini, yaitu:

- Berapa jumlah variable yang akan ditampilkan pada sebuah chart?
- Berapa poin data yang akan ditampilkan untuk setiap variable?
- Apakah nilai akan ditampilkan berdasarkan kelompok periode waktu, group atau cara lain?

Setelah itu untuk membantu pemilihan chart yang cocok dapat melihat gambar di bawah ini.



Gambar 48. Tipe presentasi (sumber: Dr. Andrew Abela, <http://extremepresentation.typepad.com>)

Pada bab ini akan diberikan penjelasan fungsi-fungsi untuk membuat chart seperti gambar di atas. Selain menjelaskan tentang fungsi-fungsi grafik, pada bab ini juga menambahkan pembahasan tentang fungsi animasi.

Grafik

Pada sub bab ini akan dijelaskan beberapa fungsi R untuk membuat grafik sebagai salah satu cara mengenal dan eksplorasi data.

plot()

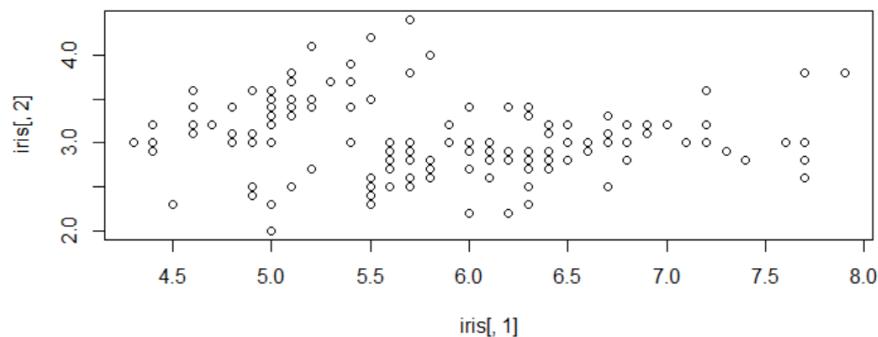
Fungsi plot() untuk membuat titik dari nilai sumbu x dan sumbu y. Sintaks umum fungsi plot() adalah sebagai berikut.

```
plot(x, y)
```

Parameter x dan y berisi nilai-nilai numerik. Sebagai contoh, parameter x diisi dengan nilai-nilai numerik dari kolom ke-1 dataset iris. Dan parameter y diisi dengan nilai-nilai dari kolom ke-2 dataset iris.

```
plot(iris[,1], iris[,2])
```

Output dari fungsi ini adalah sebagai berikut.



Gambar 49. Grafik plot(iris[,1], iris[,2])

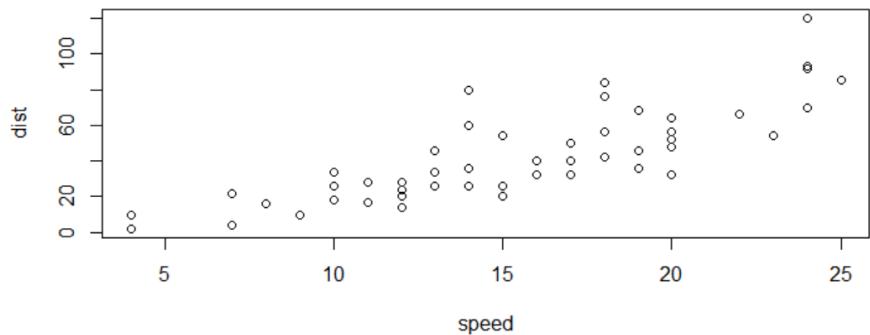
Jika dataset hanya terdiri atas 2 feature saja, sebagai contoh dataset cars.

```
> head(cars)
   speed dist
1      4    2
2      4   10
3      7    4
4      7   22
5      8   16
6      9   10
. . .
```

Maka fungsi plot() dapat digunakan dengan cara berikut ini.

```
plot(cars)
```

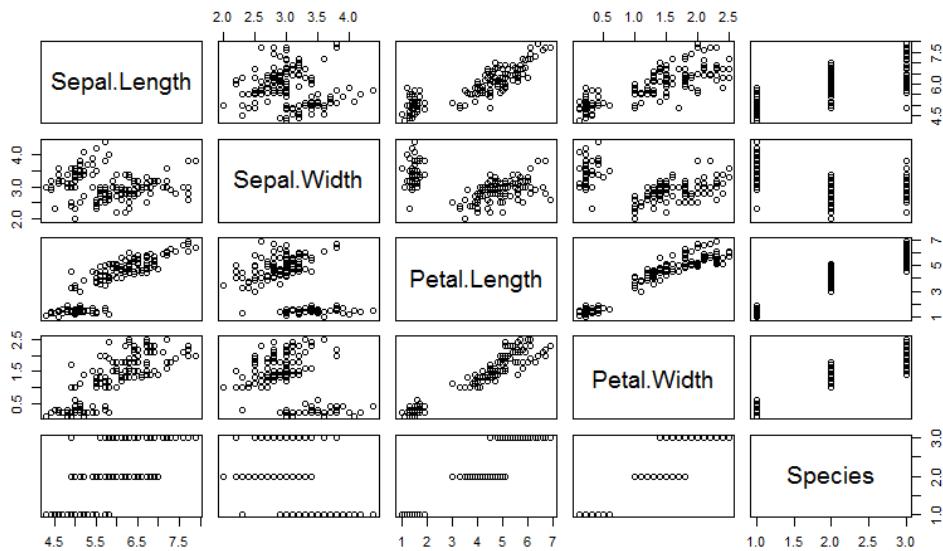
Output dari kode di atas adalah sebagai berikut.



Gambar 50. Grafik plot(cars).

Jika cara diatas digunakan pada dataset yang terdiri lebih dari 2 feature, sebagai contoh dataset iris yang memiliki 5 feature, maka fungsi plot() akan secara otomatis membuat banyak grafik. Grafik akan berisi kombinasi seperti pada gambar di bawah ini.

```
plot(iris)
```



Gambar 51. Grafik plot(iris).

scatterplot3()

Untuk membuat plot 3 dimensi diperlukan package tambahan yaitu scatterplot3d. Langkah pertama adalah menginstall package ini dengan fungsi berikut.

```
install.packages("scatterplot3d")
```

Kemudian muat package ini dengan fungsi di bawah ini.

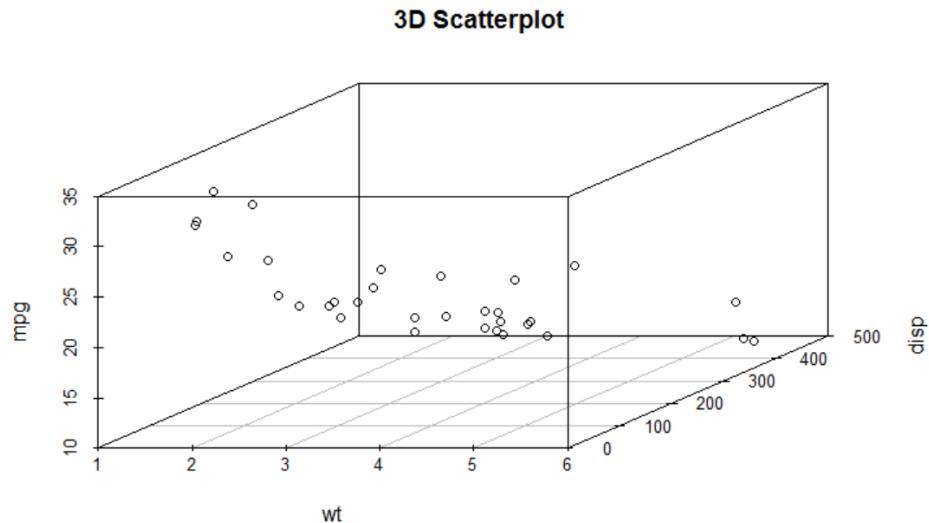
```
library(scatterplot3d)
```

Fungsi untuk membuat plot 3 dimensi adalah scatterplot3d(). Sintaks dari fungsi ini adalah sebagai berikut.

```
scatterplot3d(x, y, z, main="YourTitle")
```

Sebagai contoh untuk membuat plot 3 dimensi dari data mtcars digunakan kode di bawah ini.

```
attach(mtcars)
scatterplot3d(wt,disp,mpg, main="3D Scatterplot")
```



Gambar 52. Grafik scatterplot3d().

plot3d()

Fungsi ini juga dapat digunakan untuk membuat grafik 3 dimensi. Kelebihan fungsi ini adalah interaksi user. User dapat melihat grafik dari sudut yang diinginkan dengan cara memutar ke arah yang diinginkan.

Langkah pertama untuk menggunakan fungsi ini adalah memuat package rgl.

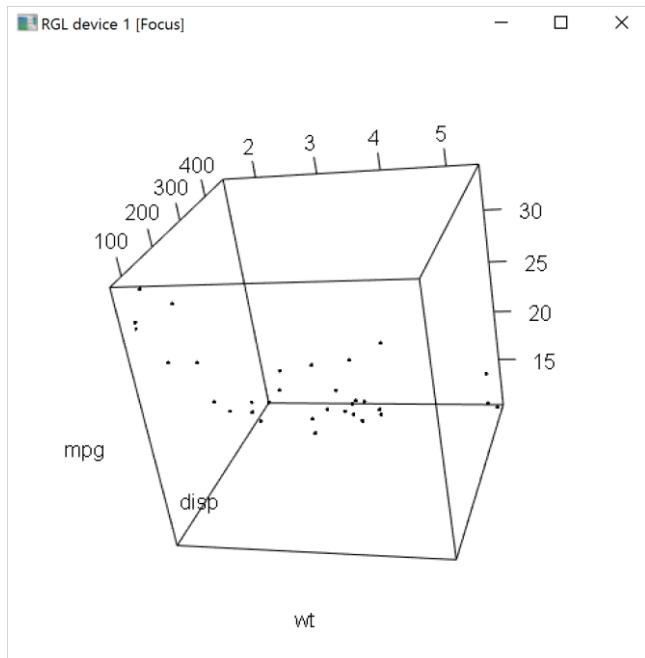
```
library(rgl)
```

Sintaks dari fungsi plot3d() adalah sebagai berikut.

```
plot3d(x, y, z)
```

Untuk membuat plot dari data mtcars dapat dilihat pada kode di bawah ini.

```
plot3d(wt, disp, mpg)
```



Gambar 53. Grafik plot3d().

Gerakkan grafik dengan mengarahkan cursor mouse pada grafik. Kemudian klik tombol mouse sebelah kiri dan tahan. Kemudian gerakkan cursor ke kiri atau ke kanan, dan ke atas atau ke bawah.

hist()

Fungsi hist() digunakan untuk membuat grafik histogram. Grafik ini dapat memberikan informasi frekuensi distribusi data. Sintaks fungsi hist() adalah sebagai berikut.

```
hist(x)
```

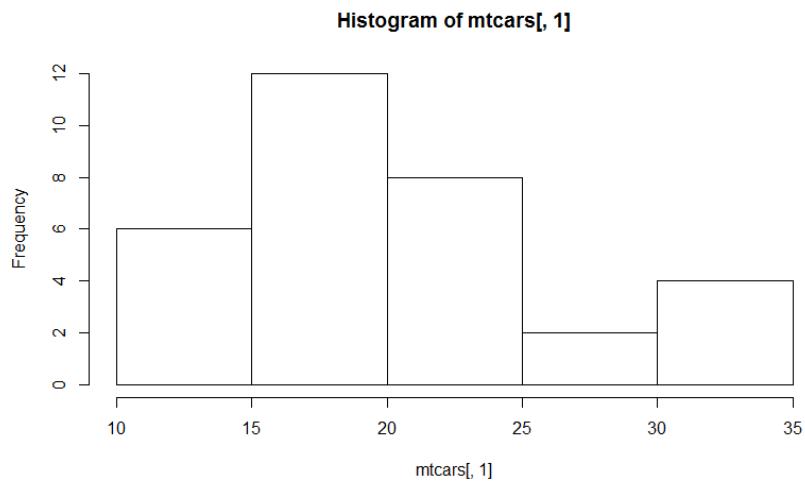
Parameter x dapat diisi dengan obyek numerik.

Selain dataset iris, platform R memiliki dataset lain yang dapat digunakan. Salah satunya adalah dataset mtcars.

```
> head(mtcars)
      mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4     21.0   6 160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag 21.0   6 160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710    22.8   4 108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive 21.4   6 258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360 175 3.15 3.440 17.02  0  0    3    2
Valiant       18.1   6 225 105 2.76 3.460 20.22  1  0    3    1
```

Jika parameter x diisi dengan nilai numerik feature mpg, maka akan didapat grafik histogram sebagai berikut.

```
hist(mtcars[,1])
atau
hist(mtcars$mpg)
```



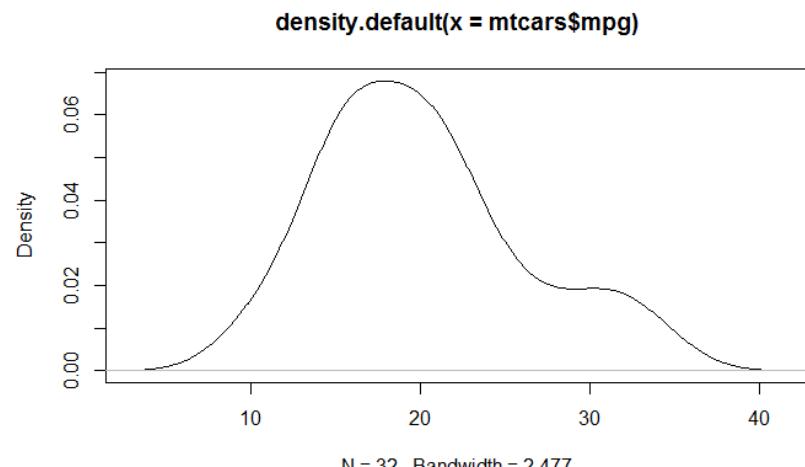
Gambar 54. Grafik hist(mtcars[,1])

density()

Fungsi density() berguna untuk melakukan estimasi kernel density. Output fungsi ini dapat dibuat dalam bentuk grafik dengan cara seperti contoh di bawah ini.

```
plot(density(mtcars$mpg))
```

Hasil dari kode di atas adalah sebagai berikut.

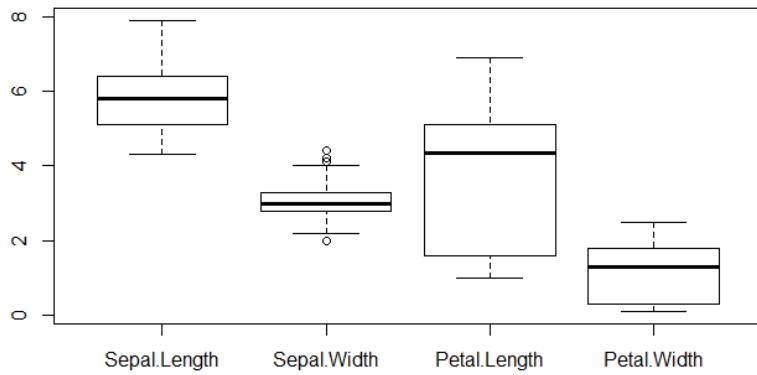


Gambar 55. Grafik density.

boxplot()

Fungsi boxplot() berfungsi untuk membuat grafik box dan whisker dari sekumpulan nilai. Contoh penggunaan fungsi ini adalah sebagai berikut.

```
boxplot(iris[,1:4])
```



Gambar 56. Grafik fungsi boxplot().

featurePlot()

Fungsi featurePlot() membuat plot density. Fungsi ini memudahkan membuat plot density berdasarkan feature-feature pada dataset. Plot dapat dibuat berdasarkan kelompok data tersebut.

Fungsi featurePlot() adalah bagian dari package caret. Untuk menggunakan fungsi ini, terlebih dahulu perlu diinstall pakcage caret dengan perintah berikut ini.

```
install.packages("caret")
```

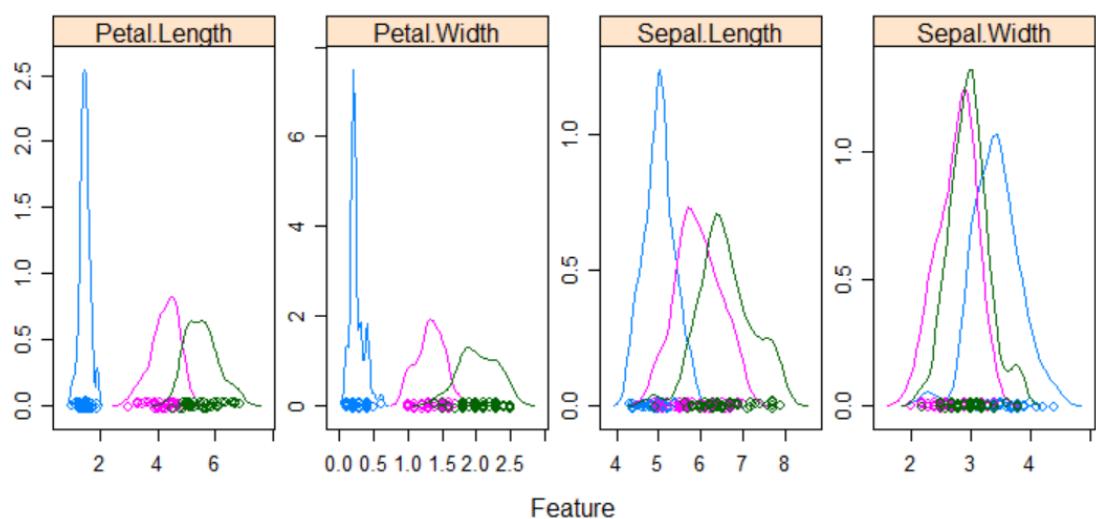
Dan berikut adalah contoh kode penggunaan fungsi ini.

```
library(caret)

iris_features = iris[,1:4]
iris_class = iris[,5]
scales = list(x=list(relation="free"), y=list(relation="free"))

featurePlot(x=iris_features, y=iris_class, plot="density", scales=scales)
```

Hasilnya adalah sebagai berikut.



Gambar 57. featurePlot() tipe density untuk dataset iris.

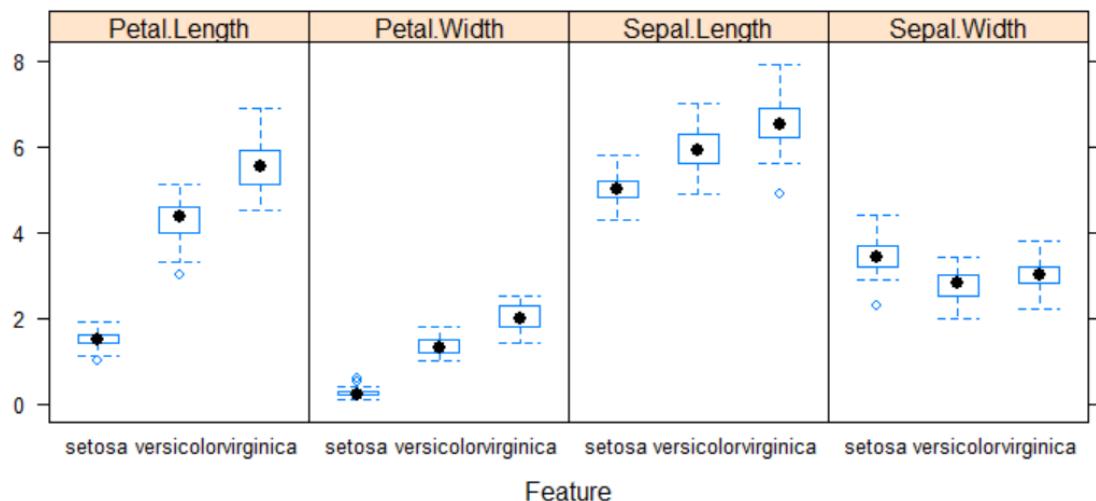
Dari gambar di atas dapat dilihat plot density dari feature-feature dataset iris. Dan dapat pula dilihat perbedaan setiap plot density berdasarkan class. Dari plot di atas juga dapat dilihat

ada beberapa class yang tidak overlap seperti pada feature Petal.Length dan Petal.Width. Sedangkan pada feature Sepal.Width terlihat susah untuk digunakan membedakan class.

Fungsi featurePlot() juga dapat digunakan untuk melihat feature dalam bentuk boxplot. Berikut adalah contoh kode yang dapat digunakan.

```
library(caret)  
  
iris_features = iris[,1:4]  
iris_class = iris[,5]  
  
featurePlot(x=iris_features, y=iris_class, plot="box")
```

Hasilnya dapat dilihat pada gambar di bawah ini.



Gambar 58. featurePlot() tipe box untuk dataset iris.

pie()

Fungsi pie() digunakan untuk membuat grafik pie. Sintaks fungsi ini adalah sebagai berikut.

```
pie(x, labels = y)
```

Sebagai contoh adalah sebagai berikut.

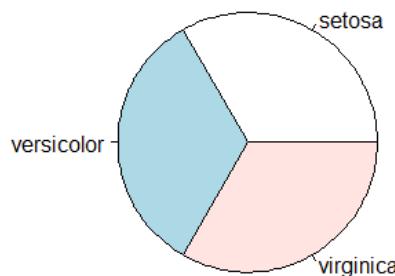
```
pie(c(30,70), labels = c("wanita", "pria"))
```



Gambar 59. Grafik pie()

Jika dimiliki obyek table maka akan lebih mudah untuk membuat grafik pie. Sebagai contoh sebagai berikut.

```
pie(table(iris[,5]))
```



Gambar 60. Grafik pie dataset iris.

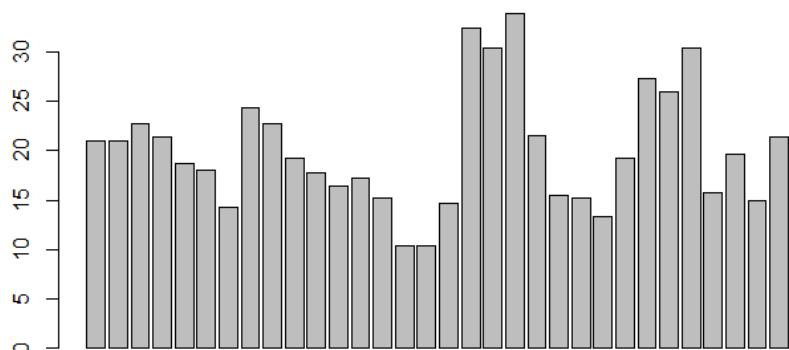
barplot()

Fungsi barplot() digunakan untuk membuat grafik berbentuk bar. Sintaks fungsi ini adalah sebagai berikut.

```
barplot(x)
```

Berikut adalah contoh penggunaan fungsi barplot().

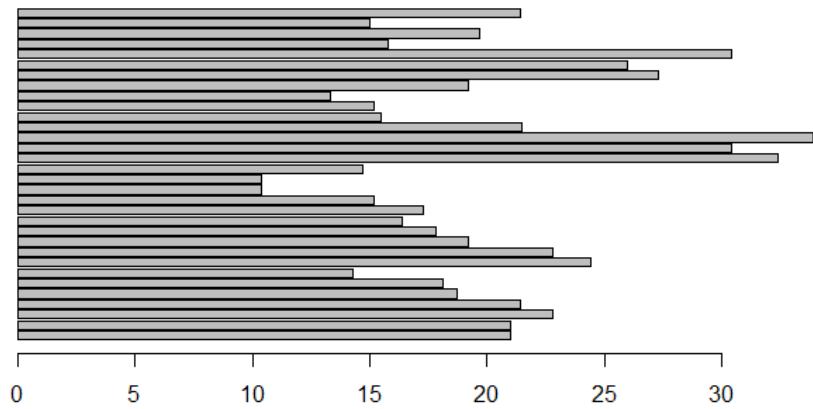
```
barplot(mtcars$mpg)
```



Gambar 61. Grafik barplot vertical.

Untuk membuat grafik horizontal maka digunakan opsi horiz = TRUE seperti contoh di bawah ini.

```
barplot(mtcars$mpg, horiz = TRUE)
```



Gambar 62. Grafik barplot horizontal.

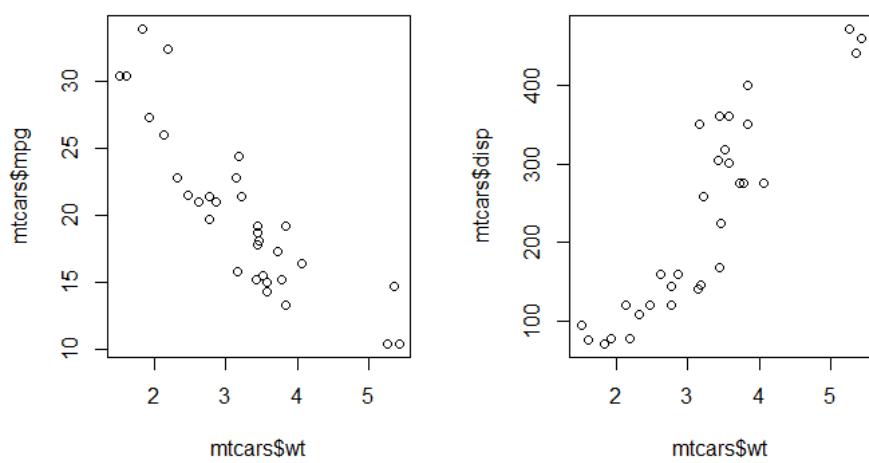
par()

Fungsi `par()` dapat digunakan untuk membuat beberapa grafik hasil fungsi-fungsi di atas dalam satu gambar. Sintaks dari fungsi `par()` adalah sebagai berikut:

```
par(mfrow=c(jumlah_grafik_dalam_baris, jumlah_grafik_dalam_kolom))
```

Sebagai contoh jika ingin membuat 2 output grafik dalam satu gambar maka digunakan kode berikut.

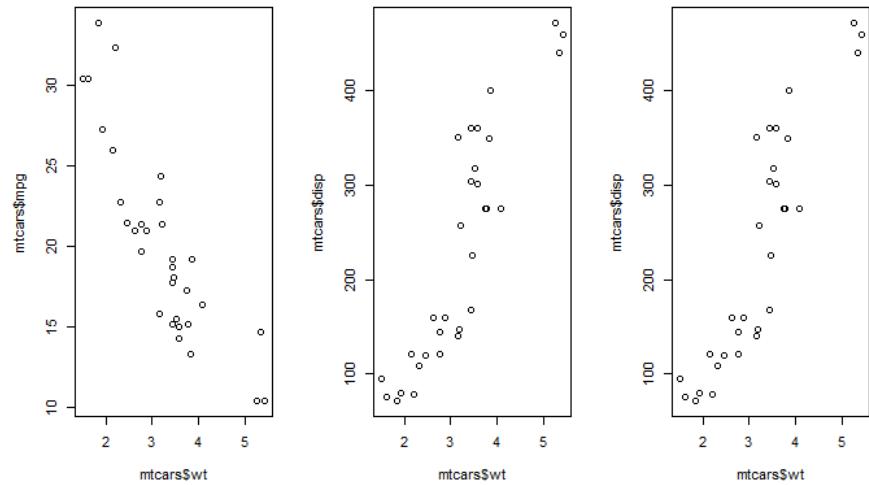
```
par(mfrow=c(1,2))  
plot(mtcars$wt,mtcars$mpg)  
plot(mtcars$wt,mtcars$disp)
```



Gambar 63. Fungsi par() untuk membuat 2 grafik dalam 1 kolom.

Jika ingin membuat 3 grafik dalam 1 baris maka digunakan kode seperti berikut.

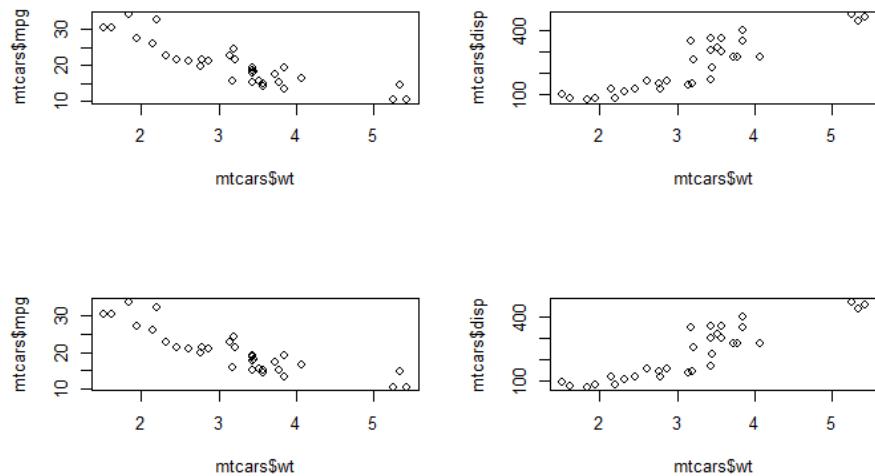
```
par(mfrow=c(1, 3))
plot(mtcars$wt, mtcars$mpg)
plot(mtcars$wt, mtcars$disp)
plot(mtcars$wt, mtcars$disp)
```



Gambar 64. Fungsi par() untuk membuat 3 grafik dalam 1 kolom.

Untuk membuat 4 grafik dalam 2 baris dan 2 kolom maka digunakan kode berikut ini.

```
par(mfrow=c(2, 2))
plot(mtcars$wt, mtcars$mpg)
plot(mtcars$wt, mtcars$disp)
plot(mtcars$wt, mtcars$mpg)
plot(mtcars$wt, mtcars$disp)
```



Gambar 65. Fungsi par() untuk membuat 4 grafik dalam 2 baris dan 2 kolom.

dotchart()

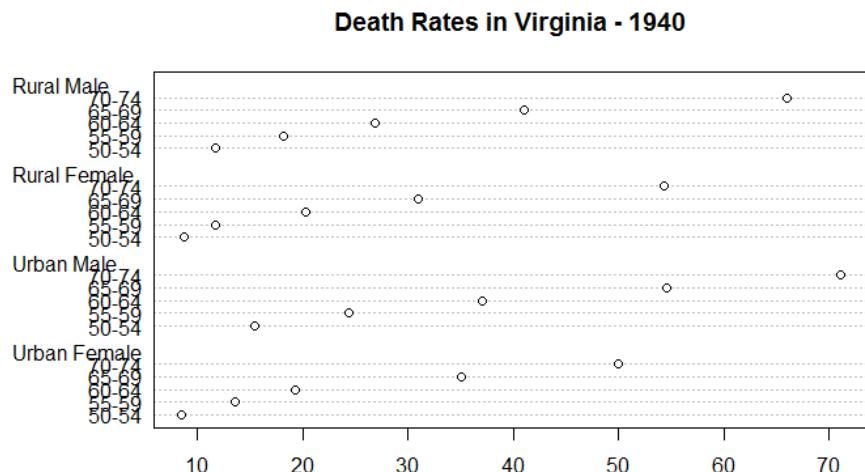
Fungsi ini digunakan untuk menggambar Cleveland dot plot. Sintaks fungsi ini adalah sebagai berikut.

```
dotchart(data, main = "title")
```

Sebagai contoh adalah sebagai berikut.

```
dotchart(VADeaths, main = "Death Rates in Virginia - 1940")
```

Hasilnya sebagai berikut.



Gambar 66. Fungsi dotchart() untuk membuat Cleveland dot plot.

missmap()

Data yang dimiliki kadang tidak sempurna. Kadang terdapat nilai yang hilang. Hilangnya data akan berpengaruh besar pada proses selanjutnya seperti proses modeling klasifikasi atau proses clustering.

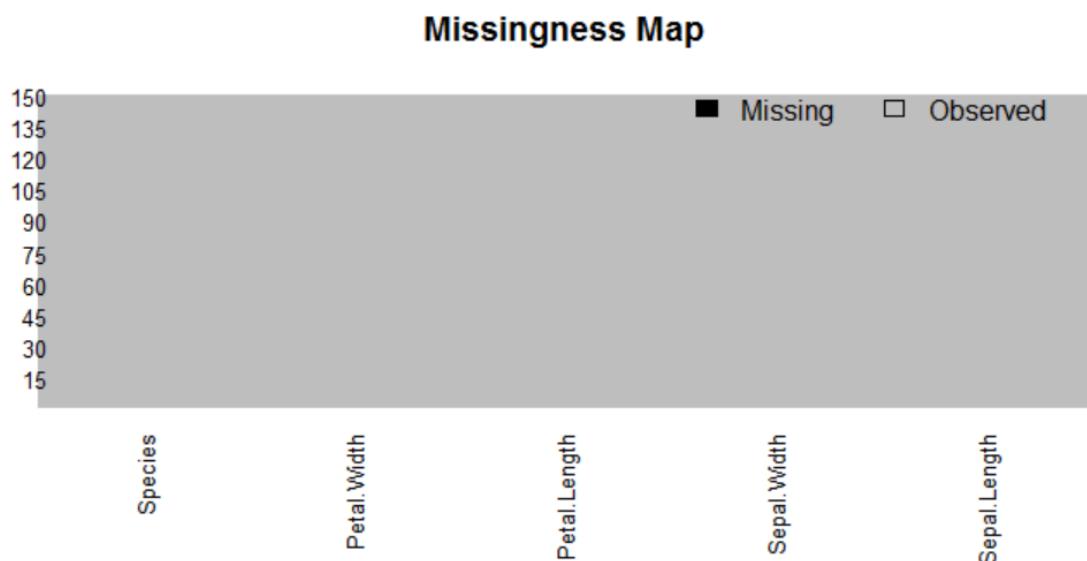
Fungsi missmap() dapat memberikan informasi jumlah data yang hilang pada dataset. Fungsi ini merupakan bagian dari package Amelia. Untuk menginstall package ini digunakan kode berikut.

```
install.packages("Amelia")
```

Dan berikut ini adalah contoh penggunaannya.

```
library(Amelia)
missmap(iris, col=c("black", "grey"))
```

Hasilnya seperti pada gambar di bawah ini.



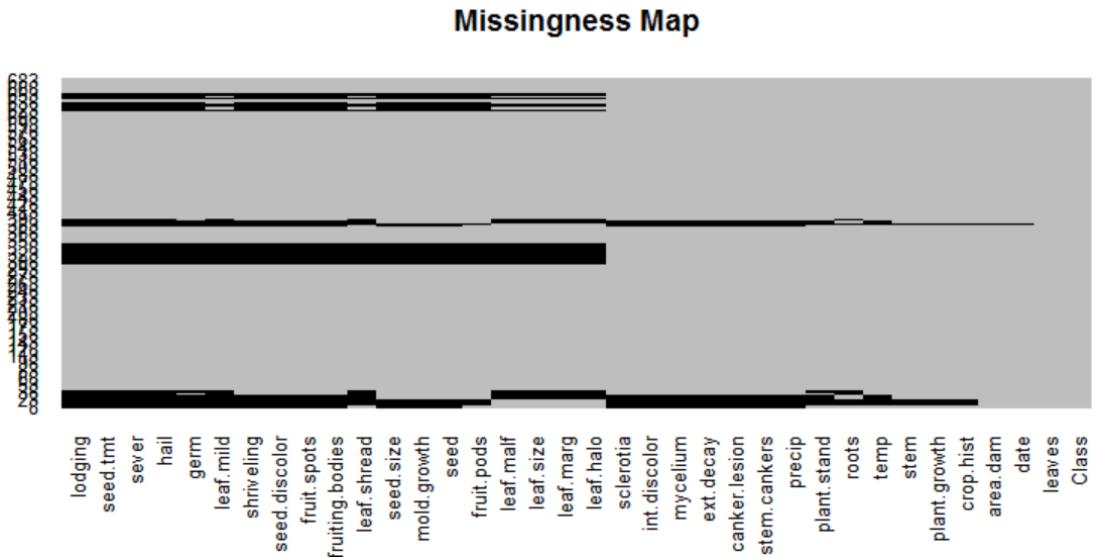
Gambar 67. missmap() dataset iris.

Contoh yang lain adalah menggunakan dataset Soybean dari package mlbench.

```
library(Amelia)
library(mlbench)
data(Soybean)

missmap(Soybean, col=c("black", "grey"), legend = FALSE)
```

Hasilnya adalah sebagai berikut.



Gambar 68. Missmap dataset Soybean.

Dari kedua contoh di atas, dapat dilihat jika dataset iris tidak terdapat data yang hilang. Berbeda dengan dataset Soybean yang banyak data yang hilang.

corrplot()

Untuk melihat korelasi antara sepasang atribut numerik dapat digunakan fungsi cor(). Sedangkan untuk melihat korelasi tersebut dalam bentuk grafik dapat digunakan fungsi corrplot() yang merupakan bagian dari package corrplot.

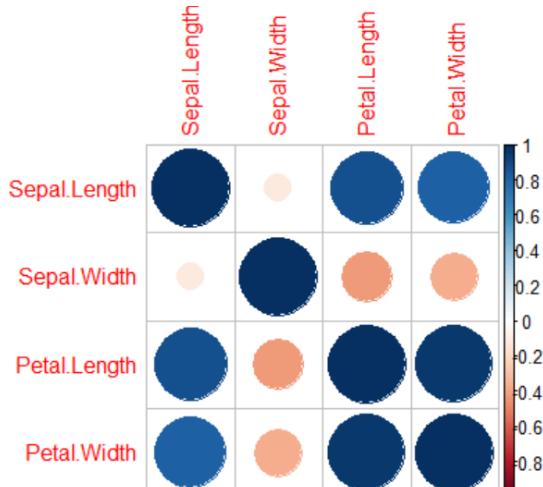
Untuk menginstall package corrplot digunakan perintah berikut.

```
install.packages("corrplot")
```

Sedangkan contoh kode yang dapat digunakan adalah sebagai berikut.

```
library(corrplot)

data(iris)
correlations <- cor(iris[,1:4])
corrplot(correlations, method="circle")
```



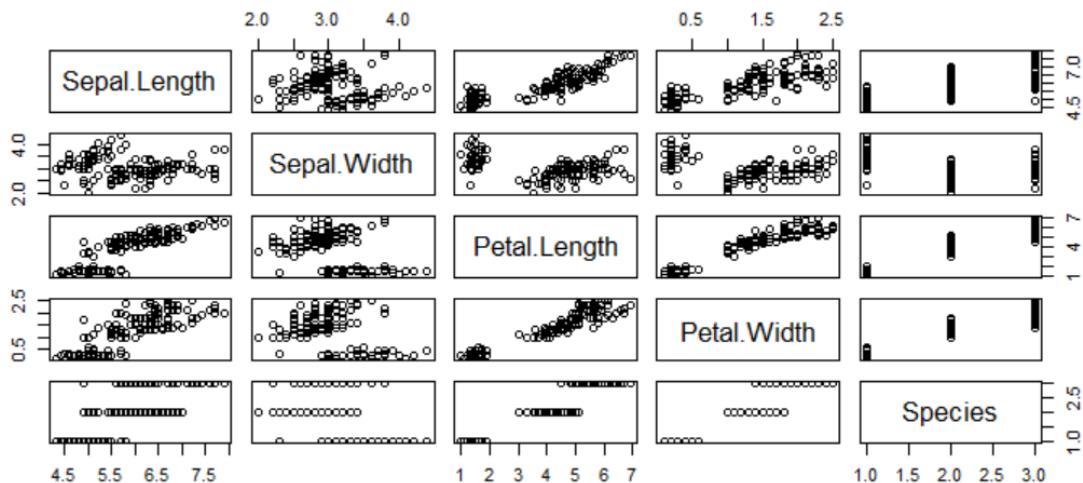
Gambar 69. `corrplot()` dataset iris.

Dengan melihat ukuran lingkaran dan warna dari gambar di atas dapat dilihat jika korelasi Sepal.Width dan Sepal.Length tidak tinggi. Sedangkan korelasi Petal.Length dan Sepal.Length mempunyai korelasi yang tinggi.

pairs()

Fungsi pairs() berfungsi untuk membuat plot antara dua variable. Fungsi ini dapat digunakan untuk melihat interaksi antara dua variable di dalam dataset. Berikut ini adalah contoh kode penggunaan fungsi pairs().

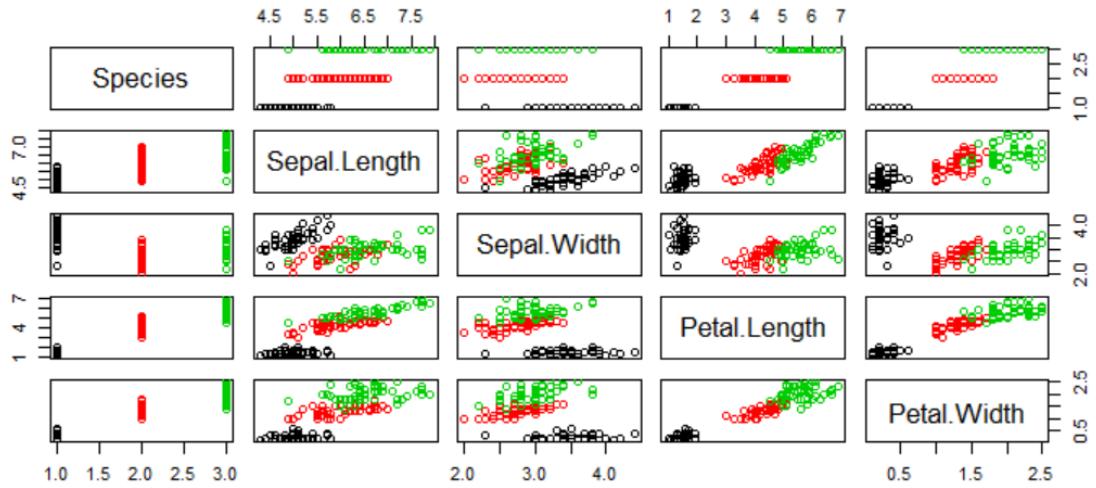
```
pair(iris)
```



Gambar 70. `pairs()` dataset iris.

Seperti diketahui bahwa dataset iris membagi data menjadi 3 class. Jika ingin membuat plot plot di atas berdasarkan pembagian 3 class tersebut, maka dapat digunakan kode berikut.

```
pairs(Species~, data=iris, col=iris$Species)
```



Gambar 71. `pair()` dataset iris berdasarkan class.

Simbol

Berikut contoh simbol-simbol yang dapat digunakan pada grafik atau plot.

```
plot(1, 1, xlim=c(1,5.5), ylim=c(0,7), type="n", ann=FALSE)

text(1:5, rep(6,5), labels=c(0:4), cex=1:5, col=1:5)

points(1:5, rep(5,5), cex=1:5, col=1:5, pch=0:4)
text((1:5)+0.4, rep(5,5), cex=0.6, (0:4))

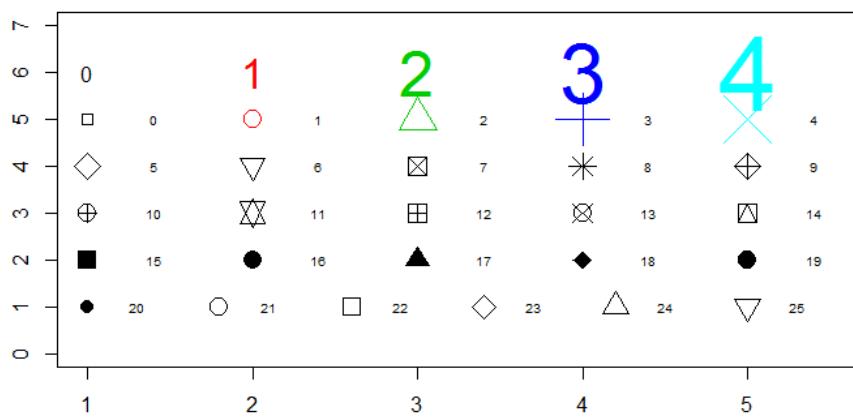
points(1:5, rep(4,5), cex=2, pch=(5:9))
text((1:5)+0.4, rep(4,5), cex=0.6, (5:9))

points(1:5, rep(3,5), cex=2, pch=(10:14))
text((1:5)+0.4, rep(3,5), cex=0.6, (10:14))

points(1:5, rep(2,5), cex=2, pch=(15:19))
text((1:5)+0.4, rep(2,5), cex=0.6, (15:19))

points((1:6)*0.8+0.2, rep(1,6), cex=2, pch=(20:25))
text((1:6)*0.8+0.5, rep(1,6), cex=0.6, (20:25))
```

Berikut adalah simbol-simbol hasil dari kode-kode di atas.



Gambar 72. Simbol grafik.

Export Gambar

Selain mengekspor data yang disimpan pada objek, juga dimungkinkan untuk mengekspor gambar chart hasil plot untuk disimpan dalam file. Format file yang bisa digunakan adalah:

1. PDF menggunakan fungsi `pdf("nama_file.pdf")`.
2. WMF, windows metafile menggunakan fungsi `win.metafile("nama_file.wmf")`.
3. PNG, menggunakan fungsi `png("nama_file.png")`.
4. JPEG, menggunakan fungsi `jpeg("nama_file.jpg")`.
5. BMP, menggunakan fungsi `bmp("nama_file.bmp")`.
6. Postscript, menggunakan fungsi `postscript("nama_file.ps")`.

Berikut ini adalah beberapa contoh kode yang digunakan untuk mengekspor chart hasil plot ke dalam file dengan format di atas.

```
pdf("kumpulan_chart.pdf")
barplot(table(mtcars$gear), main="Distribusi Mobil", xlab="Jumlah Gears")
dev.off()
```

Maka outputnya adalah file 1 halaman pdf yang berisi 1 gambar chart bar. Jika ingin dalam 1 file pdf tersebut memiliki lebih dari 1 gambar chart maka bisa dibuat kode seperti berikut.

```
pdf("kumpulan_chart.pdf")

barplot(table(mtcars$gear), main="Distribusi Mobil", xlab="Jumlah Gears")
boxplot(mpg~cyl,data=mtcars, main="Data Jarak Tempuh", xlab="Jumlah Silinder", ylab="mil/galon")
plot(wt, mpg, main="Scatterplot", xlab="Berat Mobil", ylab="mil/galon")

dev.off()
```

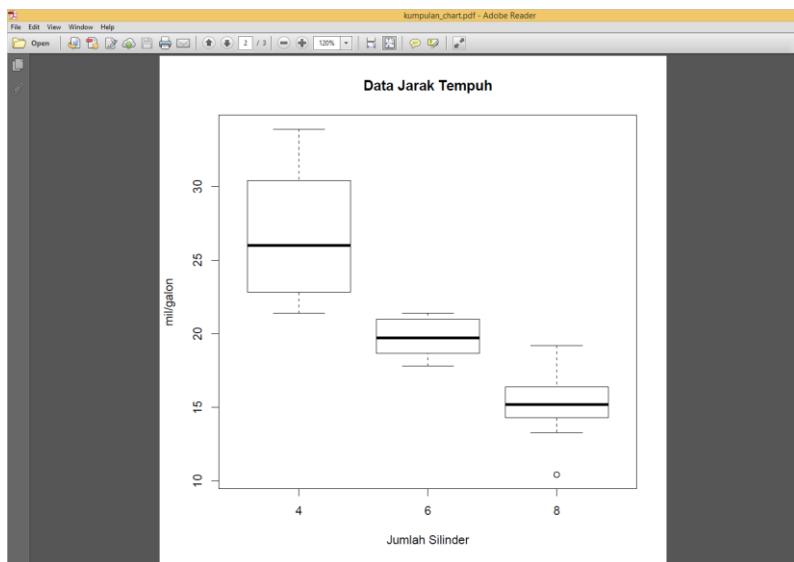
Jika digunakan 5 baris di atas maka akan dihasilkan 1 file pdf dengan nama kumpulan_chart.pdf dengan halaman sebanyak 3 halaman, dimana setiap halaman akan berisi sebuah gambar chart.

Sebagai contoh dapat dilihat pada gambar halaman PDF berikut ini. Pada lingkaran merah di gambar di bawah ini dapat dilihat jumlah halaman.

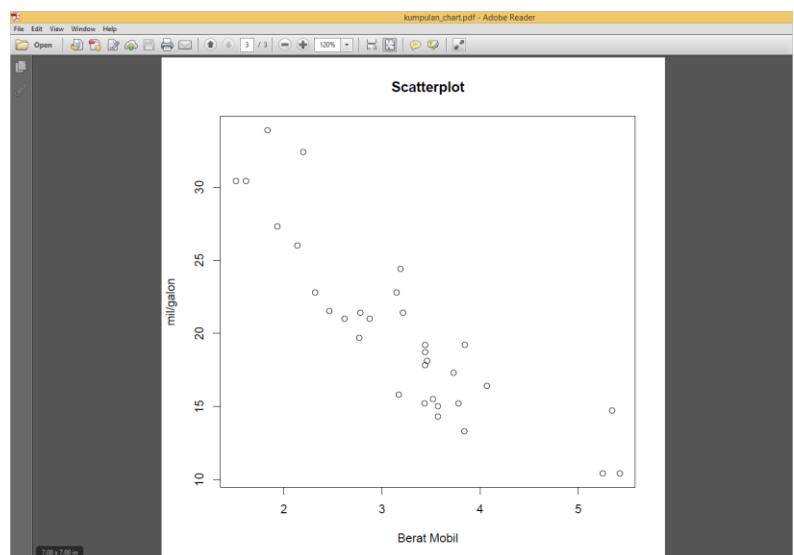


Gambar 73. Halaman pertama file kumpulan_chart.pdf.

Pada gambar selanjutnya dapat dilihat chart yang lain pada halaman kedua dan ketiga.



Gambar 74. Halaman kedua file kumpulan_chart.pdf.



Gambar 75. Halaman ketiga file kumpulan_chart.pdf.

Hal ini tidak berlaku jika diekspor ke file gambar seperti PNG, JPG, BMP dan lainnya karena pada file gambar tidak mengenal halaman seperti file bertipe dokumen seperti PDF. Jika cara seperti contoh di atas dilakukan maka yang disimpan adalah gambar cari perintah plot terakhir.

Dari contoh di atas maka penggunaan fungsi 1 sampai 6 di atas membuat fungsi plot yang digunakan tidak akan langsung ditampilkan di layar komputer pada tab plot seperti contoh-contoh yang telah diberikan pada bagian Plot Data.

Pada baris terakhir pada setiap contoh di atas terdapat fungsi dev.off(). Fungsi tersebut telah dibahas pada bagian Plot Data. Jika fungsi tersebut belum dipanggil maka file pdf atau gambar tidak akan bisa dibuka karena dianggap file tersebut masih diakses untuk ditulisi.

Visualisasi Dengan Package ggplot

Masih kosong

Animasi

Selain membuat grafik statik, dengan package animation dapat juga dibuat animasi. Untuk menginstall package ini digunakan perintah berikut ini.

```
install.packages("animation")
```

Dan untuk memuat package tersebut digunakan fungsi berikut.

```
library(animation)
```

ani.options()

Fungsi ani.option() digunakan untuk setup opsi animasi. Sintaks fungsi ini adalah sebagai berikut.

```
ani.options(interval, nmax)
```

Keterangan:

- interval adalah waktu interval dalam detik.
- nmax adalah jumlah maksimal langkah atau iterasi untuk membuat frame animasi.

ani.pause()

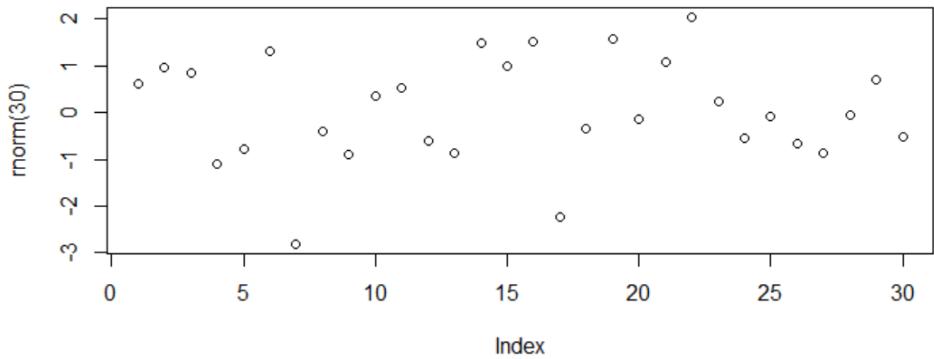
Fungsi ani.pause() digunakan untuk menghentikan (pause) sementara sesuati dengan waktu interval yang telah ditentukan.

Dengan mengetahui fungsi ani.option() dan ani.pause() maka telah dapat dibuat animasi sederhana dengan kode sebagai berikut.

```
oopt = ani.options(interval = 0.2, nmax = 10)

for (i in 1:ani.options("nmax")) {
  plot(rnorm(30))
  ani.pause()
}
```

Hasilnya dapat dilihat animasi seperti berikut.



Gambar 76. Fungsi ani.options() & ani.pause().

saveHTML()

Fungsi saveHTML() digunakan untuk menyimpan animasi menjadi halaman HTML. Sintaks fungsi ini adalah sebagai berikut.

```
saveHTML (
  {
    #Program animasi
  },
  htmlfile, ani.height, ani.width, title
)
```

Keterangan:

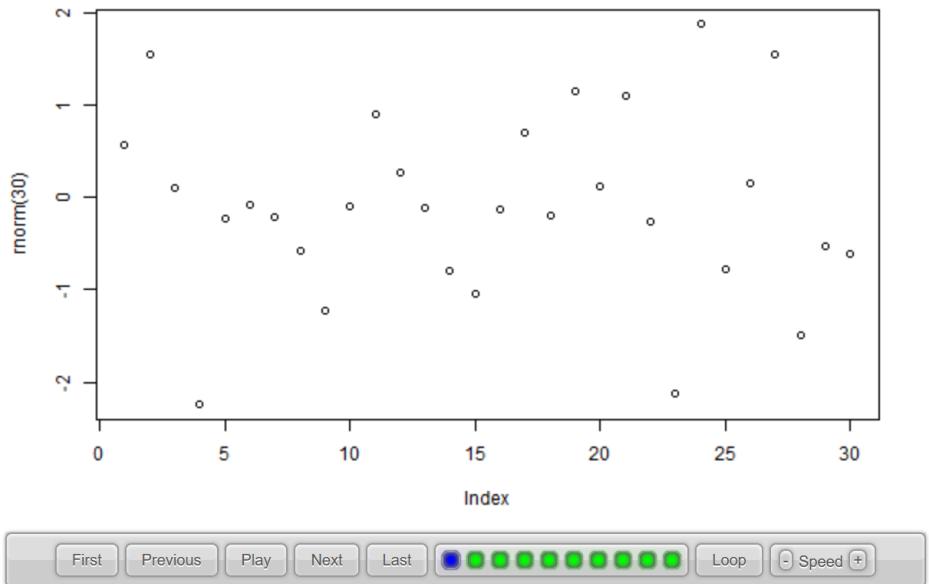
- pada blok program animasi akan berisi program animasi.
- htmlfile adalah nama file HTML yang menyimpan animasi.
- ani.height adalah tinggi animasi.
- ani.weight adalah lebar animasi.
- title adalah titel animasi.

Berikut adalah contoh penggunaan fungsi di atas.

```
saveHTML (
  {
    oopt = ani.options(interval = 0.2, nmax = 10)

    for (i in 1:ani.options("nmax")) {
      plot(rnorm(30))
      ani.pause()
    }
  },
  htmlfile = "contoh_animasi.html", ani.height = 400, ani.width = 600, title
= "Contoh Animasi"
)
```

Hasilnya adalah sebagai berikut.



Gambar 77. Fungi saveHTML().

saveGIF()

Fungsi saveGIF() digunakan untuk menyimpan animasi ke dalam file gambar GIF animated. Fungsi ini dapat digunakan jika pada komputer telah diinstall ImageMagick. Sintaks fungsi ini adalah sebagai berikut.

```
saveGIF (
  {
    #Program animasi
  },
  movie.name, ani.height, ani.width
)
```

Dan berikut adalah contoh penggunaan fungsi ini.

```
saveGIF (
  {
    oopt = ani.options(interval = 0.2, nmax = 10)

    for (i in 1:ani.options("nmax")) {
      plot(rnorm(30))
      ani.pause()
    }
  },
  movie.name = "contoh_animasi.gif", ani.height = 400, ani.width = 600
)
```

Fungsi-fungsi untuk menyimpan animasi lainnya adalah :

- saveLatex().
- saveSWF().
- saveVideo().

Seperti halnya saveGIF(), fungsi-fungsi di atas dapat digunakan jika aplikasi pendukung telah diinstall pada komputer.

Referensi

- <http://shinyapps.org/apps/RGraphCompendium/index.php>
- <https://www.analyticsvidhya.com/blog/2015/08/cheat-sheet-data-visualization-r/>
- <https://www.analyticsvidhya.com/blog/2015/07/guide-data-visualization-r/>
- <http://machinelearningmastery.com/data-visualization-in-r/>
- [http://www.cookbook-r.com/Graphs/Plotting_distributions_\(ggplot2\)/](http://www.cookbook-r.com/Graphs/Plotting_distributions_(ggplot2)/)
- <http://t-redactyl.io/blog/2016/02/creating-plots-in-r-using-ggplot2-part-7-histograms.html>
- <http://www.cookbook-r.com/Graphs/>
- <https://rpubs.com/mccanneccology/53464>
- <http://machinelearningmastery.com/data-visualization-in-r/>
- <http://www.sthda.com/english/wiki/static-and-interactive-heatmap-in-r-unsupervised-machine-learning>
- <https://www.r-bloggers.com/7-visualizations-you-should-learn-in-r/> (PENTING)
- <http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#8.%20Spatial> (KEREN!)

6

Contoh Kasus

Pada bab ini akan diberikan beberapa contoh-contoh masalah dan penyelesaiannya dengan menggunakan bahasa pemrograman R.

Pengolahan Data Awal

Pada bidang data science, data yang akan digunakan pada proses klasifikasi atau clustering harus mempunyai struktur yang sederhana seperti data tabular seperti yang umum ditemui pada aplikasi spreadsheets seperti MS Excel.

Tetapi umumnya data yang ditemui belum siap untuk langsung digunakan pada kasus tersebut. Data yang mungkin terpisah-pisah dalam beberapa media penyimpanan atau file. Oleh karena itu perlu ada pengolahan data awal atau pre processing data.

Sebagai contoh misalnya dimiliki file-file sebagai berikut. File pertama adalah compound_ids.txt yang berisi daftar senyawa kimia sebagai berikut.

compound_ids.txt
DB00014
DB00035
DB00050
DB00091
DB00093
DB00104
DB00114
DB00115
DB00116
DB00117

File yang kedua adalah protein_ids.txt yang berisi daftar protein dengan isi sebagai berikut.

protein_ids.txt
A0R559
A1A2B5
A1AFW1
A2QLK4
A3FMN7
A5W4F4
A7B6P0
A7J8L3
A8MPY1
A9CJ63

File ketiga adalah compounds_proteins_interactions.txt berisi pemetaan interaksi antara senyawa dengan protein, jika bernilai 1 artinya keduanya memiliki interaksi. Dan jika bernilai 0 maka keduanya belum diketahui interaksinya. Pencatatan pemetaan ini dalam bentuk matrix sebagai berikut ini.

compounds	proteins	interactions.txt
0	1	0
0	1	0
1	0	0
1	0	1
0	1	0

0	1	0	1	0	0	0	1	0	1
1	1	0	1	0	0	0	1	0	1
0	1	0	1	0	0	0	0	1	1
0	1	0	1	0	1	0	0	0	1
1	1	0	0	0	0	0	1	0	0

Baris menyatakan compound dan kolom menyatakan protein. Selanjutnya adalah file compound_features.csv yang berisi nilai-nilai dari feature-feature yang dimiliki oleh senyawa.

compound features.csv
5.1,3.5,1.4,0.2
4.9,3.0,1.4,0.2
4.7,3.2,1.3,0.2
4.6,3.1,1.5,0.2
5.0,3.6,1.4,0.2
5.4,3.9,1.7,0.4
4.6,3.4,1.4,0.3
5.0,3.4,1.5,0.2
4.4,2.9,1.4,0.2
4.9,3.1,1.5,0.1

Selanjutnya adalah protein_features.csv yang berisi nilai-nilai dari feature-feature yang dimiliki oleh protein.

protein features.csv
6.7,3.1,4.4,1.4
5.6,3.0,4.5,1.5
5.8,2.7,4.1,1.0
6.2,2.2,4.5,1.5
5.6,2.5,3.9,1.1
5.9,3.2,4.8,1.8
6.1,2.8,4.0,1.3
6.3,2.5,4.9,1.5
6.1,2.8,4.7,1.2
6.4,2.9,4.3,1.3

Dari file-file di atas akan dibuat sebuah file yang siap digunakan untuk diolah dalam proses klasifikasi. Yaitu gabungan nilai-nilai feature senyawa, nilai-nilai feature protein, nama senyawa, nama protein dan nilai interaksinya.

Langkah pertama adalah membaca file-file tersebut dan disimpan ke dalam object-object.

```
rm(list=ls())
filename_compound = "compound_ids.txt"
filename_protein = "protein_ids.txt"
filename_matrix = "compounds_proteins_interactions.txt"
filename_compound_feature = "compound_features.csv"
filename_protein_feature = "protein_features.csv"

data_compound = read.table(filename_compound)
data_protein = read.table(filename_protein)
data_matrix = read.table(filename_matrix)
data_compound_feature = read.csv(filename_compound_feature, header = FALSE)
data_protein_feature = read.csv(filename_protein_feature, header = FALSE)
```

Jika dilihat isi object-object di atas akan berisi data dengan struktur sebagai berikut ini.

> data_matrix
V1 V2 V3 V4 V5 V6 V7 V8 V9 V10
1 0 1 0 1 0 0 0 0 0 1
2 0 1 0 0 0 0 0 1 0 1
3 1 0 0 0 0 0 0 0 0 1
4 1 0 0 1 0 1 0 0 0 1
5 0 1 0 1 0 0 0 0 0 1
6 0 1 0 1 0 0 0 1 0 1
7 1 1 0 1 0 0 0 1 0 1
8 0 1 0 1 0 0 0 0 1 1

```

9   0   1   0   1   0   1   0   0   0   0   1
10  1   1   0   0   0   0   0   1   0   0   0

> data_compound_feature
  V1  V2  V3  V4
1  5.1 3.5 1.4 0.2
2  4.9 3.0 1.4 0.2
3  4.7 3.2 1.3 0.2
4  4.6 3.1 1.5 0.2
5  5.0 3.6 1.4 0.2
6  5.4 3.9 1.7 0.4
7  4.6 3.4 1.4 0.3
8  5.0 3.4 1.5 0.2
9  4.4 2.9 1.4 0.2
10 4.9 3.1 1.5 0.1

> data_protein_feature
  V1  V2  V3  V4
1  6.7 3.1 4.4 1.4
2  5.6 3.0 4.5 1.5
3  5.8 2.7 4.1 1.0
4  6.2 2.2 4.5 1.5
5  5.6 2.5 3.9 1.1
6  5.9 3.2 4.8 1.8
7  6.1 2.8 4.0 1.3
8  6.3 2.5 4.9 1.5
9  6.1 2.8 4.7 1.2
10 6.4 2.9 4.3 1.3

```

Agar data di atas dapat lebih mudah dilihat maka bisa diatur nama kolom dan nama barisnya sesuai dengan nama senyawa dan protein. Untuk memberi nama kolom dan baris pada object `data_matrix` dapat dilakukan dengan cara berikut ini.

```

> colnames(data_matrix) = unlist(data_protein)
> rownames(data_matrix) = unlist(data_compound)

> data_matrix
      A0R559 A1A2B5 A1AFW1 A2QLK4 A3FMN7 A5W4F4 A7B6P0 A7J8L3 A8MPY1 A9CJ63
DB00014    0     1     0     1     0     0     0     0     0     1
DB00035    0     1     0     0     0     0     0     1     0     1
DB00050    1     0     0     0     0     0     0     0     0     1
DB00091    1     0     0     1     0     1     0     0     0     1
DB00093    0     1     0     1     0     0     0     0     0     1
DB00104    0     1     0     1     0     0     0     1     0     1
DB00114    1     1     0     1     0     0     0     1     0     1
DB00115    0     1     0     1     0     0     0     0     1     1
DB00116    0     1     0     1     0     1     0     0     0     1
DB00117    1     1     0     0     0     0     0     0     1     0

```

Sedangkan untuk memberikan nama baris pada object `data_compound_feature` dan `data_protein_feature` digunakan cara sebagai berikut ini.

```

rownames(data_protein_feature) = unlist(data_protein)
rownames(data_compound_feature) = unlist(data_compound)

```

Hasilnya dapat dilihat di bawah ini.

```

> data_protein_feature
  V1  V2  V3  V4
A0R559 6.7 3.1 4.4 1.4
A1A2B5 5.6 3.0 4.5 1.5
A1AFW1 5.8 2.7 4.1 1.0
A2QLK4 6.2 2.2 4.5 1.5
A3FMN7 5.6 2.5 3.9 1.1
A5W4F4 5.9 3.2 4.8 1.8
A7B6P0 6.1 2.8 4.0 1.3
A7J8L3 6.3 2.5 4.9 1.5
A8MPY1 6.1 2.8 4.7 1.2
A9CJ63 6.4 2.9 4.3 1.3
> data_compound_feature

```

	V1	V2	V3	V4
DB00014	5.1	3.5	1.4	0.2
DB00035	4.9	3.0	1.4	0.2
DB00050	4.7	3.2	1.3	0.2
DB00091	4.6	3.1	1.5	0.2
DB00093	5.0	3.6	1.4	0.2
DB00104	5.4	3.9	1.7	0.4
DB00114	4.6	3.4	1.4	0.3
DB00115	5.0	3.4	1.5	0.2
DB00116	4.4	2.9	1.4	0.2
DB00117	4.9	3.1	1.5	0.1

Selanjutnya adalah membaca interaksi antara senyawa dan protein yang disimpan pada object data_matrix. Sebagai contoh, pertama akan dibaca nilai pada baris 1 kolom 1 yang merupakan interaksi senyawa DB00014 dan protein A0R559 dengan nilai iteraksi adalah 0. Kemudian untuk baris 1 dan kolom 2 merupakan interaksi senyawa DB00014 dan protein A1A2B5 dengan nilai interaksi adalah 1. Jika nilai 0 diganti dengan "no" dan nilai 1 diganti dengan nilai "yes" maka akan didapat data sebagai berikut.

5.1,3.5,1.4,0.2,6.7,3.1,4.4,1.4,DB00014,A0R559,no
5.1,3.5,1.4,0.2,6.7,3.1,4.4,1.4,DB00014,A1A2B5,yes

Untuk mendapatkan data seperti di atas secara lengkap dengan kode program berikut ini.

```
pre_processing_data.R
rm(list=ls())

filename_compound = "compound_ids.txt"
filename_protein = "protein_ids.txt"
filename_matrix = "compounds_proteins_interactions.txt"
filename_compound_feature = "compound_features.csv"
filename_protein_feature = "protein_features.csv"

data_compound = read.table(filename_compound)
data_protein = read.table(filename_protein)
data_matrix = read.table(filename_matrix)
data_compound_feature = read.csv(filename_compound_feature, header = FALSE)
data_protein_feature = read.csv(filename_protein_feature, header = FALSE)

colnames(data_matrix) = unlist(data_protein)
rownames(data_matrix) = unlist(data_compound)

rownames(data_protein_feature) = unlist(data_protein)
rownames(data_compound_feature) = unlist(data_compound)

if(exists("result_all")){
  rm(result_all)
}

for(i in 1:nrow(data_matrix)){
  for(j in 1:ncol(data_matrix)){
    #mengambil nama baris ke-i dari object data_matrix
    compound_id = rownames(data_matrix)[i]

    #mengambil sebaris data dari object data_compound_feature
    #baris yang dipilih sesuai nama row dengan nilai compound_id
    compound_feature = data_compound_feature[compound_id,]

    #mengambil nama kolom ke-j dari object data_matrix
    protein_id = colnames(data_matrix)[j]

    #mengambil sebaris data dari object data_protein_feature
    #baris yang dipilih sesuai nama row dengan nilai protein_id
    protein_feature = data_protein_feature[protein_id,]

    #mengubah nilai data_matrix[i,j]
    #jika nilai adalah 0 maka menjadi no
  }
}
```

```

#jika nilai adalah 1 maka menjadi yes
class_label = "no"
if(data_matrix[i,j] == 1) {
    class_label = "yes"
}

#mengabung seluruh object-object di atas ke dalam object result
result = cbind.data.frame(compound_feature, protein_feature,
compound_id, protein_id, class_label)

#mengubah nama kolom dari object result
colnames(result) =
c("c1","c2","c3","c4","p1","p2","p3","p4","compound_id","protein_id","class_label")

#memasukkan object result ke dalam object result_all
#object result_all akan berisi ixj baris (100 baris)
if(!exists("result_all")){
    assign("result_all", result)
} else {
    result_all = rbind.data.frame(result_all, result)
}
}

#mengubah nama baris object result_all menjadi angka
rownames(result_all) = c(1:nrow(result_all))

#menyimpan object result_all ke dalam file format csv
write.csv(result_all, "result_all.csv", quote = FALSE, row.names = FALSE)

```

Hasilnya dapat dilihat isi object result_all seperti gambar di bawah ini.

	c1	c2	c3	c4	p1	p2	p3	p4	compound_id	protein_id	class_label
1	5.1	3.5	1.4	0.2	6.7	3.1	4.4	1.4	DB00014	A0R559	no
2	5.1	3.5	1.4	0.2	5.6	3.0	4.5	1.5	DB00014	A1A2B5	yes
3	5.1	3.5	1.4	0.2	5.8	2.7	4.1	1.0	DB00014	A1AFW1	no
4	5.1	3.5	1.4	0.2	6.2	2.2	4.5	1.5	DB00014	A2QLK4	yes
5	5.1	3.5	1.4	0.2	5.6	2.5	3.9	1.1	DB00014	A3FMN7	no
6	5.1	3.5	1.4	0.2	5.9	3.2	4.8	1.8	DB00014	A5W4F4	no
7	5.1	3.5	1.4	0.2	6.1	2.8	4.0	1.3	DB00014	A7B6P0	no
8	5.1	3.5	1.4	0.2	6.3	2.5	4.9	1.5	DB00014	A7J8L3	no
9	5.1	3.5	1.4	0.2	6.1	2.8	4.7	1.2	DB00014	A8MPY1	no
10	5.1	3.5	1.4	0.2	6.4	2.9	4.3	1.3	DB00014	A9Cj63	yes
11	4.9	3.0	1.4	0.2	6.7	3.1	4.4	1.4	DB00035	A0R559	no
12	4.9	3.0	1.4	0.2	5.6	3.0	4.5	1.5	DB00035	A1A2B5	yes
13	4.9	3.0	1.4	0.2	5.8	2.7	4.1	1.0	DB00035	A1AFW1	no
14	4.9	3.0	1.4	0.2	6.2	2.2	4.5	1.5	DB00035	A2QLK4	no
15	4.9	3.0	1.4	0.2	5.6	2.5	3.9	1.1	DB00035	A3FMN7	no
16	4.9	3.0	1.4	0.2	5.9	3.2	4.8	1.8	DB00035	A5W4F4	no
17	4.9	3.0	1.4	0.2	6.1	2.8	4.0	1.3	DB00035	A7B6P0	no
18	4.9	3.0	1.4	0.2	6.3	2.5	4.9	1.5	DB00035	A7J8L3	yes

Gambar 78. Pengolahan data awal – result_all.

Persiapan Data Cross Validation

Pada kasus klasifikasi, akan ada dua proses utama yaitu pelatihan dan pengujian. Setiap proses akan menggunakan data. Pada proses pelatihan, data yang digunakan disebut sebagai data training. Sedangkan pada proses pengujian, data yang digunakan disebut data testing. Data training dan data testing sebaiknya menggunakan sample yang berbeda.

Agar teknik atau algoritma klasifikasi dapat melakukan kedua proses itu dengan baik, maka perlu disiapkan data dengan baik. Salah satu teknik yang digunakan untuk menguji teknik klasifikasi tersebut adalah cross validation.

Sebagai contoh adalah dataset iris.

```
> summary(iris)
   Sepal.Length   Sepal.Width    Petal.Length   Petal.Width      Species
   Min. :4.300   Min. :2.000   Min. :1.000   Min. :0.100   setosa   :50
   1st Qu.:5.100 1st Qu.:2.800  1st Qu.:1.600  1st Qu.:0.300  versicolor:50
   Median :5.800 Median :3.000   Median :4.350   Median :1.300  virginica :50
   Mean   :5.843 Mean   :3.057   Mean   :4.358   Mean   :1.399 
   3rd Qu.:6.400 3rd Qu.:3.300  3rd Qu.:5.100  3rd Qu.:1.800 
   Max.   :7.900 Max.   :4.400   Max.   :6.900   Max.   :2.500
```

Untuk pembagian data dengan 5-folds cross validation maka akan data akan dibagi menjadi sebagai berikut.

Part 1	Part 2	Part 3	Part 4	Part 5
10 setosa				
10 versicolor				
10 virginica				

Gambar 79. Persiapan data cross validation – dataset iris.

Kemudian untuk proses pelatihan dan pengujian akan dilakukan sebanyak 5 kali, dan akan digunakan data training dan data testing sebagai berikut.

Proses pertama akan digunakan data sebagai berikut.

Data Testing	Data Training			
	Part 1	Part 2	Part 3	Part 4

Gambar 80. Persiapan data cross validation – proses I.

Proses kedua akan digunakan data sebagai berikut.

Data Training	Data Testing	Data Training		
		Part 1	Part 2	Part 3

Gambar 81. Persiapan data cross validation – proses II.

Proses ketiga akan digunakan data sebagai berikut.

Data Training	Data Testing	Data Training		
		Part 1	Part 2	Part 3

Gambar 82. Persiapan data cross validation – proses III.

Proses keempat akan digunakan data sebagai berikut.

Data Training	Data Testing	Data Training		
		Part 1	Part 2	Part 3

Gambar 83. Persiapan data cross validation – proses IV.

Proses kelima akan digunakan data sebagai berikut.

Data Training				Data Testing
Part 1	Part 2	Part 3	Part 4	Part 5

Gambar 84. Persiapan data cross validation – proses V.

Untuk membuat data training dan data testing untuk kasus 5 folds cross validation dapat digunakan kode program di bawah ini.

```
pre_cross_validation.R
rm(list=ls())

#5 folds cross validation
cross_num = 5

#nomer kolom class label
class_index = 5

#nama file output
filename_save = "iris"

#mengacak data
main_data = iris[sample(nrow(iris), nrow(iris)),]

#menentukan nama-nama class label
class_names = row.names(table(main_data[,class_index]))

for(class_name in class_names){
  #mengelompokkan data sesuai class
  main_data_class = main_data[which(main_data[,class_index] == class_name),]

  #menghitung jumlah kelompok data
  main_data_class_count = round(nrow(main_data_class)/cross_num)

  #membagi kelompok data ke dalam cross_num bagian
  for(cross_i in 1:cross_num)
  {
    start_i = cross_i

    if(cross_i == 1){
      end_i = cross_i * main_data_class_count
    }

    if(cross_i > 1){
      start_i = end_i + 1

      end_i = (cross_i)*main_data_class_count
    }

    if(cross_i == cross_num){
      end_i = nrow(main_data_class)
    }

    part_data_name = paste0("part_data_", cross_i)
    if(!exists(part_data_name)){
      assign(part_data_name, main_data_class[c(start_i:end_i),])
    } else {
      temp = get(part_data_name)
      temp = rbind(temp, main_data_class[c(start_i:end_i),])
      assign(part_data_name, temp)
    }
  }
}

#mengelompokkan data menjadi:
```

```

#data testing (main_data.test)
#data training (main_data.train)
for(cross_i in 1:cross_num){
  main_data.test = get(paste0("part_data_", cross_i))
  for(i in c(1:cross_num)[-c(cross_i)]){
    if(!exists("main_data.train")){
      main_data.train = get(paste0("part_data_", i))
    } else {
      main_data.train = rbind.data.frame(main_data.train,
get(paste0("part_data_", i)))
    }
  }

  #menulis data testing ke dalam file
  write.csv(main_data.test, paste0(filename_save,".test.",cross_i,".csv"),
quote = FALSE, row.names = FALSE)

  #menulis data trainging ke dalam file
  write.csv(main_data.train,
paste0(filename_save,".train.",cross_i,".csv"), quote = FALSE, row.names =
FALSE)
}

```

Untuk kasus di atas akan dihasilkan 10 file yaitu:

- 5 file iris.test.1.csv sampai dengan iris.test.5.csv. Setiap file akan berisi 30 sample yaitu 10 sample untuk setiap class.
- 5 file iris.train.1.csv sampai dengan iris.train.5.csv. Setiap file akan berisi 120 sample yaitu 40 sample untuk setiap class.

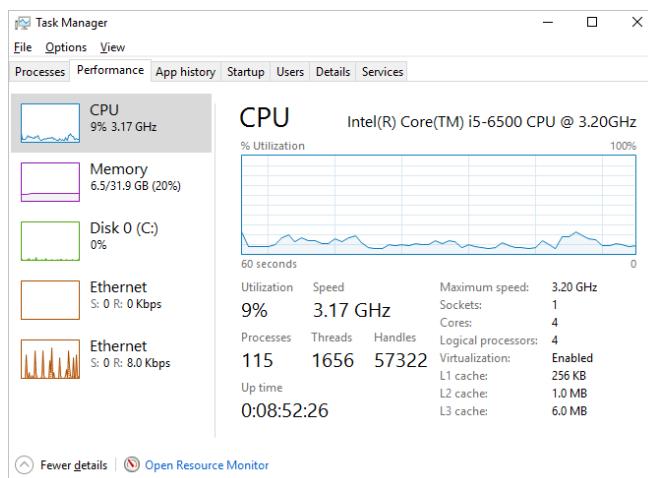
Komputasi Paralel

Pendahuluan

Jika komputer sumber daya processor dan memory yang besar maka akan mendorong kita untuk menggunakannya secara maksimal. Pada bidang data science, jika data yang digunakan sangat besar atau proses data yang rumit maka dipastikan memerlukan sumber daya komputasi yang besar. Oleh sebab itu pengetahuan pemrograman komputasi paralel pada bidang data science sangat diperlukan.

Saat ini sebagian besar komputer yang digunakan telah memiliki lebih dari 1 core. Dan umumnya program yang kita buat hanya memanfaatkan 1 core saja.

Pada sistem operasi MS Windows, untuk melihat berapa core yang dimiliki oleh komputer dapat dilakukan dengan menjalankan Task Manager. Pada gambar di bawah ini dapat dilihat komputer ini memiliki 4 core.



Gambar 85. Task Manager pada MS Windows.

Untuk mengetahui cara kerja program bekerja dan bagaimana baris-baris kode program dijalankan maka akan dijelaskan dalam bentuk contoh-contoh program di bawah ini. Berikut adalah contoh program kecil yang ditulis dengan R.

```
rm(list = ls())

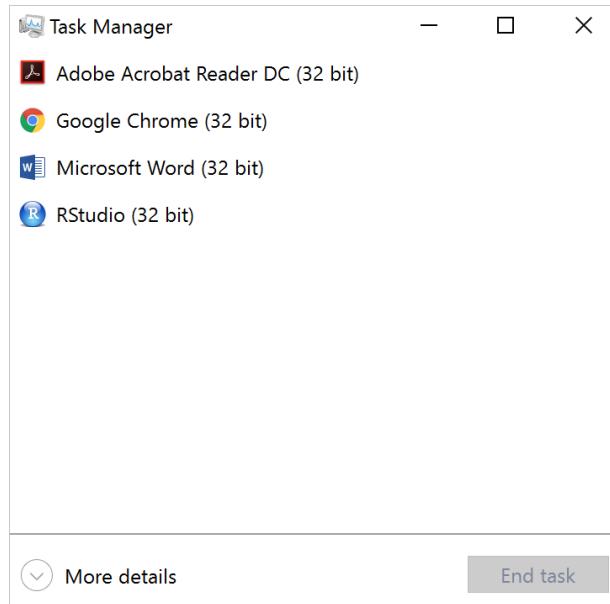
meanOfRandomNumber = function(seed) {
  set.seed(seed)
  x = mean(rnorm(30000000))
  assign(paste0("mean", seed), x, pos = .GlobalEnv)
}

system.time(
{
  for(i in 1:4){
    meanOfRandomNumber(i)
  }
})
```

Program kecil ini memerlukan waktu proses sebagai berikut.

user	system	elapsed
8.68	0.20	8.88

Dan ketika program tersebut dijalankan yang terlihat pada Task Manager hanyalah proses RStudio saja seperti yang terlihat pada gambar di bawah ini.



Gambar 86. Proses pada Task Manager.

Contoh program di atas akan membuat 4 buah obyek yang menyimpan 4 hasil hasil perhitungan rata-rata dari 30000000 bilangan random. Obyek tersebut dibuat secara pemrograman dengan memanfaatkan fungsi assign(). Perhitungan rata-rata tersebut dilakukan secara serial, artinya setelah perhitungan 30000000 bilangan pertama selesai, maka hasilnya akan disimpan dalam obyek mean1. Kemudian baru dilakukan perhitungan 30000000 kedua yang hasilnya akan disimpan dalam obyek mean2. Begitu seterusnya sampai 4 iterasi.

Program di atas juga dapat ditulis dengan cara sebagai berikut.

```
rm(list = ls())  
  
MeanRnorm = function(x) {  
  return(mean(rnorm(x)))  
}  
  
system.time({  
  lapply(c(30000000, 30000000, 30000000, 30000000), MeanRnorm)  
})
```

Dan waktu proses dari kode di atas adalah sebagai berikut.

user	system	elapsed
8.48	0.28	8.76

Atau lebih sederhana lagi kode di atas dapat ditulis seperti berikut.

```
system.time(  
{  
  mean(rnorm(30000000))  
  mean(rnorm(30000000))  
  mean(rnorm(30000000))  
  mean(rnorm(30000000))  
})
```

Dengan waktu proses sebagai berikut.

```
user    system elapsed
8.34      0.16   8.50
```

Dapat dilihat secara jelas keempat perintah tersebut dijalankan secara berurutan. Dan hanya akan menggunakan 1 core saja untuk menyelesaikan program di atas. Implementasi pemrograman komputasi memungkinkan masing-masing proses tersebut dijalankan oleh 1 core. Dengan cara seperti itu maka diharapkan waktu proses program di atas akan lebih singkat.

Persiapan

Bahasa pemrograman R mendukung pemrograman komputasi paralel. Untuk melakukan komputasi paralel pada lingkungan R dapat dilakukan dengan menggunakan fungsi-fungsi dari package parallel, foreach dan doParallel.

Untuk menginstall package-package ini digunakan perintah berikut.

```
install.packages("parallel")
install.packages("foreach")
install.packages("doParallel")
```

Fungsi Dasar Komputasi Paralel

detectCore()

Langkah pertama adalah memeriksa core yang dimiliki oleh komputer dengan fungsi detectCores() dengan cara di bawah ini.

```
library(foreach)
library(doParallel)

detectCores()
```

Keluaran dari kode di atas adalah jumlah core dari komputer yang digunakan. Jika komputer memiliki 4 core, maka sebaiknya jangan semua core digunakan untuk program komputasi paralel. Sebaiknya disisakan setidaknya 1 core untuk keperluan proses yang lain.

makeCluster()

Fungsi makeCluster() digunakan untuk untuk parallel socket cluster, yang membuat satu set salinan proses R yang berjalan secara paralel dan saling berkomunikasi melalui socket. Sintaks fungsi ini adalah sebagai berikut.

```
makeCluster(x)
```

Parameter x adalah jumlah core yang dipergunakan. Pada implementasinya fungsi ini akan ditampung dalam suatu obyek seperti contoh di bawah ini.

```
cl = makeCluster(3)
```

Pada contoh di atas digunakan 3 core dan fungsi ini akan ditampung oleh obyek cl.

stopCluster()

Fungsi stopCluster() untuk menghentikan proses proses yang telah dimulai oleh fungsi makeCluster(). Sintaks fungsi ini adalah sebagai berikut.

```
stopCluster(cl)
```

Parameter cl adalah obyek yang menampung fungsi makeCluster().

parLapply()

Fungsi parLapply() memiliki cara kerja seperti fungsi lapply(). Bedanya fungsi parLapply() dapat mendukung komputasi paralel. Sintaks fungsi ini adalah sebagai berikut.

```
parLapply(cl, X, FUNCTION, ...)
```

Keterangan:

- cl adalah obyek yang menampung fungsi makeCluster().
- X adalah data bertipe list atau vector.
- FUNCTION adalah fungsi yang akan dijalankan untuk memproses setiap elemen data X.

Berikut adalah contoh penggunaan fungsi ini.

```
library(foreach)
library(doParallel)

rm(list = ls())

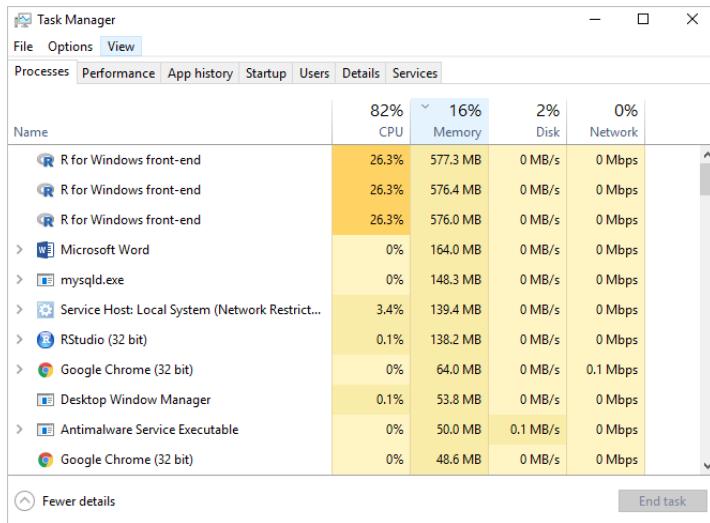
MeanRnorm = function(x) {
  return(mean(rnorm(x)))
}

system.time({
  cl<-makeCluster(3)
  parLapply(cl, c(30000000, 30000000, 30000000, 30000000), MeanRnorm)
  stopCluster(cl)
})
```

Contoh kode di atas sama fungsinya seperti contoh kode serupa yang telah diberikan pada sub bab Pendahuluan pada bab ini. Waktu yang diperlukan untuk proses kode ini adalah sebagai berikut.

user	system	elapsed
0.00	0.02	4.70

Dapat dilihat waktu proses kode ini lebih singkat jika dibandingkan contoh kode-kode pada sub bab sebelumnya. Pada contoh kode ini digunakan 3 core dari 4 core yang dimiliki oleh komputer ini. Selain itu juga dapat dilihat terlihat 3 proses tambahan pada Task Manager saat program di atas dijalankan.



Gambar 87. Proses saat menjalankan komputasi parallel.

Jika ingin menggunakan 4 core maka fungsi makeCluster() di atas diubah seperti berikut.

```
cl<-makeCluster(4)
```

Dan waktu proses menjadi lebih singkat, yaitu:

user	system	elapsed
0.00	0.01	2.86

parSapply()

Selain fungsi parLapply() juga dapat digunakan fungsi parSapply() seperti contoh berikut.

```
MeanRnorm = function(x) {
  return(mean(rnorm(x)))
}

system.time({
  cl<-makeCluster(3)
  parSapply(cl, c(30000000, 30000000, 30000000, 30000000), MeanRnorm)
  stopCluster(cl)
})
```

registerDoParallel()

Fungsi registerDoParallel() adalah fungsi untuk mendaftarkan backend komputasi paralel dengan package foreach. Fungsi pada package foreach memungkinkan kode yang akan diproses secara paralel dapat ditulis dengan lebih sederhana.

Sintaks registerDoParallel() adalah sebagai berikut.

```
registerDoParallel(cl)
```

Dimana obyek cl adalah obyek yang menampung fungsi makeCluster(). Sedangkan sintaks yang digunakan untuk implementasi komputasi paralel dengan menggunakan package foreach adalah sebagai berikut.

```
foreach(iterator) %dopar% {
  #kode yang akan dijalankan secara paralel
}
```

Parameter iterator menentukan jumlah iterasi yang akan dilakukan. Sebagai contoh dapat dilihat pada kode di bawah ini.

```
library(foreach)
library(doParallel)

system.time({
  cl<-makeCluster(3)
  registerDoParallel(cl)

  foreach(1:4) %dopar% {
    mean(rnorm(30000000))
  }

  stopCluster(cl)
})
```

Dari contoh di atas dapat dilihat kode di dalam blok foreach dijalankan secara paralel sesuai dengan jumlah core yang digunakan.

Berikut adalah modifikasi kode di atas, dengan menghilangkan penggunaan fungsi system.time().

```
library(foreach)
library(doParallel)

cl<-makeCluster(3)
registerDoParallel(cl)

foreach(1:4) %dopar% {
  mean(rnorm(30000000))
}

stopCluster(cl)
```

Dan akan dilihat hasilnya sebagai berikut.

```
[[1]]
[1] 0.0001380179

[[2]]
[1] -0.0002280828

[[3]]
[1] -0.0001759997

[[4]]
[1] 0.0001284795
```

Output dari kode di dalam blok foreach dapat ditampung dalam suatu obyek dengan cara sebagai berikut.

```
result_parallel = foreach(1:4) %dopar% {
  mean(rnorm(30000000))
}
```

Sehingga dengan memanggil obyek result_parallel maka seluruh hasil perhitungan setiap proses di dalam blok foreach dapat dilihat.

```
> result_parallel
[[1]]
[1] -0.0002514399

[[2]]
[1] 7.256806e-05

[[3]]
[1] 4.22742e-05

[[4]]
```

```
[1] 7.81653e-05
```

Contoh yang lain dapat dilihat pada kode di bawah ini.

```
library(foreach)
library(doParallel)

cl<-makeCluster(3)
registerDoParallel(cl)

results = foreach(i=1:10) %dopar% {
  data.frame(feature=rnorm(10))
}

stopCluster(cl)

View(results)
```

Pada contoh kode di atas, dapat dilihat keluaran dari proses di dalam blok foreach ditampung ke dalam obyek data.frame sehingga dapat dilihat seperti pada gambar di bawah ini.

	feature	feature.1	feature.2	feature.3	feature.4	feature.5	feature.6	feature.7	feature.8	feature.9
1	0.03107333	0.01771824	-1.2590101	0.08244188	2.3096477	1.44837045	0.5159266	-0.94687838	-0.07585526	1.6327075
2	1.51100490	0.21386099	0.9792395	0.75866642	2.3743388	0.28476734	0.7687231	0.31507583	-0.90317336	-0.6260465
3	0.81703852	-2.30063271	0.3098290	0.31304953	1.6862400	-1.23022715	0.3412017	-2.16702740	-0.32061138	-0.2413391
4	0.90458174	0.96405381	-2.2783008	0.81613242	2.1676294	0.28006257	0.8773462	1.99332949	-1.96260347	-1.3908600
5	-2.21474272	0.62765735	0.3999207	-0.76307753	0.2145991	0.03435748	-2.1148950	0.45216149	-0.37237566	-1.1169261
6	0.39040723	-0.26988401	-0.3380032	1.25236082	1.3450090	0.49030700	1.1278973	0.15746616	-1.28778920	-0.2197710
7	-0.26460103	-2.31389882	-0.4314531	-1.29339002	-1.5093216	0.69055952	-0.2451344	0.04966792	-1.35435806	-0.9679482
8	1.27496744	-0.98249938	-0.6464614	-0.27201913	0.7246561	-1.08311751	-0.3919515	0.72372276	-0.87880166	0.4656354
9	1.12806148	1.09706857	-0.6219941	-1.08436002	-1.5576142	-0.52617594	0.3812756	0.20210435	2.26938372	0.4339668
10	-0.36811384	0.14190366	-0.6754247	-0.67382941	-0.5139532	-0.85353681	-0.7325464	0.04942336	2.73819157	0.6917743

Gambar 88. Keluaran proses komputasi parallel.

Contoh Kasus

Referensi

- <https://www.r-bloggers.com/how-to-go-parallel-in-r-basics-tips/>
- <http://www.win-vector.com/blog/2016/01/parallel-computing-in-r/>
- <https://beckmw.wordpress.com/2014/01/21/a-brief-foray-into-parallel-processing-with-r/>
- <http://michaeljkoontz.weebly.com/uploads/1/9/9/4/19940979/parallel.pdf>
- <http://blog.aicry.com/r-parallel-computing-in-5-minutes/>
- <http://link.springer.com/article/10.1007/s00180-010-0206-4>
- https://hpcc.usc.edu/files/2014/07/ParallelComputingR_Snowfall.pdf (ada gambar)
- <https://www.packtpub.com/mapt/book/Big%20Data%20&%20Business%20Intelligence/9781784394004/1/ch01lvl1sec09/The+R+parallel+package>

8

Reduksi Dimensi Data

Computers have no problem processing that many dimensions. However, we humans are limited to three dimensions. Computers still need us (thankfully), so we often need ways to effectively visualize high-dimensional data before handing it over to the computer.

Pada contoh-contoh kasus di atas dapat dilihat data memiliki lebih dari 3 feature. Sebagai contoh untuk dataset iris terdiri atas 4 feature dan satu target variable. Pada umumnya data akan memiliki memiliki feature yang banyak, ada yang lebih dari 10 feature. Bahkan untuk data biology dapat memiliki lebih dari 1000 feature.

Principal Component Analysis (PCA)

Principal Component Analysis adalah teknik untuk membangun variable-variable baru yang merupakan kombinasi linear dari variable-variable asli. Atau teknik yang digunakan untuk menyederhanakan suatu data, dengan cara mentransformasi data secara linier sehingga terbentuk sistem koordinat baru dengan varians maksimum. Jumlah maximum dari variable-variable baru ini akan sama dengan jumlah dari variable lama, dan variable-variable baru ini tidak saling berkorelasi satu sama lain.

PCA dapat digunakan untuk mereduksi dimensi suatu data tanpa mengurangi karakteristik data tersebut secara signifikan atau tetap mempertahankan informasi yang terkandung di dalamnya.

Dengan penjelasan di atas, jika data memiliki feature lebih dari 3 maka PCA dapat mereduksi feature menjadi 2 atau 3 feature saja. Sehingga data dapat digambar pada grafik 2 dimensi atau 3 dimensi. Tetapi keberhasilan PCA untuk menggambarkan data ke dalam 2 dimensi dan 3 dimensi tergantung data tersebut. Jadi ada kemungkinan PCA tidak akan memberikan gambaran yang benar tentang sebaran data.

Implementasi PCA pada platform R adalah dengan menggunakan fungsi prcomp(). Data input untuk fungsi ini adalah numerik. Sebagai contoh, berikut ini adalah data iris.

```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
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
```

Kolom yang dapat digunakan sebagai input fungsi prcomp() adalah kolom ke-1 sampai ke-4. Sehingga penggunaan fungsi prcomp() dapat dituliskan sebagai berikut.

```
iris.pca = prcomp(iris[, -5]) # seluruh kolom, kecuali kolom ke-5
```

atau

```
iris.pca = prcomp(iris[, 1:4]) # kolom ke-1 sampai ke-4
```

Fungsi ini menghasilkan 4 output. Output yang pertama adalah standar deviasi principal component. Output ini dapat dilihat dengan cara di bawah ini.

```
> iris.pca$sdev  
[1] 2.0562689 0.4926162 0.2796596 0.1543862
```

Output kedua adalah matrix variable loading. Kode di bawah ini digunakan untuk melihat output ini.

```
> iris.pca$rotation  
PC1 PC2 PC3 PC4  
Sepal.Length 0.36138659 -0.65658877 0.58202985 0.3154872  
Sepal.Width -0.08452251 -0.73016143 -0.59791083 -0.3197231  
Petal.Length 0.85667061 0.17337266 -0.07623608 -0.4798390  
Petal.Width 0.35828920 0.07548102 -0.54583143 0.7536574
```

Output ketiga adalah nilai rotasi data. Output ini dapat dilihat dengan menggunakan kode berikut. Jumlah data output ini sesuai dengan jumlah data pada dataset iris yaitu 150 sample.

```
> iris.pca$x  
PC1 PC2 PC3 PC4  
[1,] -2.684125626 -0.319397247 0.027914828 0.0022624371  
[2,] -2.714141687 0.177001225 0.210464272 0.0990265503  
[3,] -2.888990569 0.144949426 -0.017900256 0.0199683897  
...  
[149,] 1.900941614 -0.116627959 -0.723251563 0.0445953047  
[150,] 1.390188862 0.282660938 -0.362909648 -0.1550386282
```

Output yang terakhir adalah nilai center. Nilai center dapat dilihat dengan perintah berikut.

```
> iris.pca$center  
Sepal.Length Sepal.Width Petal.Length Petal.Width  
5.843333 3.057333 3.758000 1.199333
```

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Multidimensional Scaling

Referensi

- <https://www.analyticsvidhya.com/blog/2015/07/dimension-reduction-methods/>
- <https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/>
- <https://www.analyticsvidhya.com/blog/2017/01/t-sne-implementation-r-python/>
- <http://www.statmethods.net/advstats/mds.html>
- <https://www.r-statistics.com/2016/01/multidimensional-scaling-with-r-from-mastering-data-analysis-with-r/>
- <https://www.r-bloggers.com/multidimensional-scaling-mds-with-r/>
- <http://setosa.io/ev/principal-component-analysis/> (PENTING!!)
- <https://www.r-bloggers.com/7-functions-to-do-metric-multidimensional-scaling-in-r/>

9

Referensi

- <https://www.analyticsvidhya.com/blog/2016/02/complete-tutorial-learn-data-science-scratch/>
- <http://rstudio.github.io/shiny/tutorial/>
- <http://pluto.huji.ac.il/~msby/StatThink/IntroStat.pdf>
- <http://www.statmethods.net/stATS/index.html>
- http://www.ssc.wisc.edu/sscc/pubs/RFR/RFR_DataExpl.html (PENTING)
- <http://www.bigskyassociates.com/blog/bid/356764/5-Most-Important-Methods-For-Statistical-Data-Analysis>
- <http://www.fao.org/docrep/W7295E/w7295e08.htm> (basic statistical tools)
- <http://www.kdnuggets.com/2016/01/7-common-data-science-mistakes.html>
- <http://www.kdnuggets.com/2015/03/machine-learning-data-science-common-mistakes.html/>
- https://rpubs.com/bradleyboehmke/data_wrangling (DATA PROCESSING - dplyr & tidyverse)

10

Penutup

Ebook ini dibuat untuk menjawab keingintahuan penulis akan R dan statistik, dan ketika keingintahuan itu sudah terjawab maka salah satu caranya agar jawaban itu tidak hilang adalah dengan menulisnya. Selain sebagai catatan pribadi juga dapat dimanfaatkan orang lain yang mempunyai pertanyaan yang serupa seperti saya sebelumnya.

Masih banyak pertanyaan-pertanyaan lain yang belum terjawab, jadi masih banyak hal-hal bidang statistik dan R yang masih belum masuk ke dalam ebook ini. Semoga pertanyaan-pertanyaan tersebut dapat dijawab pada ebook selanjutnya yang akan fokus pada pertanyaan tentang Machine Learning di lingkungan R.

Akhir kata, semoga ebook ini dapat bermanfaat. Jika ada kesalahan pada isi buku ini, silakan kirimkan saran dan kritiknya via email reza.faisal (at) gmail.com.

Terima kasih.