

Algoritma dan Pemrograman R

Bakti Siregar, M.Sc

20 Agustus 2022

Contents

Kata Pengantar	7
Ringkasan Pembelajaran	7
Tim Penyusun	7
Ucapan Terima Kasih	9
Masukan & Saran	9
1 Pengenalan R?	11
1.1 Fitur Utama R	11
1.2 Mengapa Belajar R?	12
1.3 Download R & Rstudio:	12
1.4 Tutorial Instal R & Studio	12
1.5 Video Instalasi R & RStudio	14
1.6 Interface R & RStudio:	14
1.7 Sintaks Dasar R	15
1.8 Bantuan (Help) R	15
1.9 Shortcut Penggunaan Rstudio	16
1.10 Praktikum	16
2 Operasi Dasar R	17
2.1 Variabel	17
2.2 Operator	18
2.3 Tipe Data	21
2.4 Statistika Dasar	22

2.5	Contoh Pemrograman Dasar	22
2.6	Latihan	24
3	Struktur Data	27
3.1	Vektor	28
3.2	Matriks	29
3.3	Array	30
3.4	Faktor	30
3.5	Data Frame	30
3.6	Lists	31
3.7	Rekayasa Data Frame	33
3.8	Latihan	36
4	Ekstraksi Data	37
4.1	Impor Dataset	37
4.2	Ekstraksi Baris	37
4.3	Ekstraksi Subset	38
4.4	Ekstraksi Dengan Variabel Baru	38
4.5	Ekstraksi Summary Statistik	39
4.6	Mengubah Nama	39
4.7	Latihan	41
5	Fungsi dalam R	43
5.1	Fungsi Satu Argumen	44
5.2	Simpan Fungsi	46
5.3	Fungsi Multi Argumen	47
5.4	Fungsi untuk Data Frame	49
5.5	Latihan	56

CONTENTS	5
6 Struktur Kontrol	59
6.1 Pengambilan Keputusan	59
6.2 Pengulangan Rekursif	62
6.3 Interupsi Pengulangan	67
6.4 Pengulangan Berkala	70
6.5 Skip Iterasi Pengulangan	71
6.6 Latihan	73
7 Interface Data	75
7.1 Impor/Ekspor CSV	76
7.2 Impor/Ekspor Excel	77
7.3 Impor/Ekspor TXT and RDS	78
7.4 Impor/Ekspor XML	78
7.5 Impor/Ekspor JSON	79
7.6 Impor Data dari Web	79
7.7 Basis Data R	80
7.8 Menambah Data Web	80
7.9 Latihan	80
8 Manipulasi Data	81
8.1 Impor Data	82
8.2 Struktur Data	82
8.3 Missing Value	83
8.4 Mengganti Nilai yang Hilang	84
8.5 Memilih Data	85
8.6 Menyaring dan Mengurutkan Data	86
8.7 Mengganti Nama dan Mengubah (Mutate)	87
8.8 Menggabungkan Data	88
8.9 Kelompok dan Ringkasan	90
8.10 Membagi Data	91
9 Referensi	95

Kata Pengantar

Bahasa pemrograman R telah menjadi alat yang kuat bagi para ilmuwan data, analis statistik, dan praktisi analisis numerik di seluruh dunia. Dengan kemampuan yang luar biasa dalam manipulasi data, visualisasi, dan analisis statistik, R memungkinkan para profesional untuk menggali wawasan berharga dari kumpulan data yang kompleks.

Dalam buku ini, penulis menyediakan materi dasar-dasar bahasa pemrograman R hingga tingkat yang lebih mendalam. Penulis juga menjelaskan beberapa konsep-konsep penting, sintaksis dasar, struktur data, serta memberikan contoh nyata tentang bagaimana R dapat digunakan dalam berbagai konteks. Modul ini dirancang untuk membantu pembaca yang baru mengenal pemrograman maupun yang telah memiliki pengalaman sebelumnya dalam bahasa lain.

Ringkasan Pembelajaran

Adapun isi pembelajaran dalam modul ini adalah sebagai berikut:

-

Siswa diminta untuk membuat proyek kecil menggunakan R yang menggabungkan konsep-konsep yang telah dipelajari. Proyek dapat berupa analisis data, pemecahan masalah, atau aplikasi sederhana.

Tim Penyusun

Berikut ini adalah nama dan biografi singkat para penulis:

- **Bakti Siregar, M.Sc** adalah Ketua Program Studi di Jurusan Statistika Universitas Matana. Lulusan Magister Matematika Terapan dari National Sun Yat Sen University, Taiwan. Beliau juga merupakan dosen

dan konsultan Data Scientist di perusahaan-perusahaan ternama seperti JNE, Samora Group, Pertamina, dan lainnya. Beliau memiliki antusiasme khusus dalam mengajar Big Data Analytics, Machine Learning, Optimisasi, dan Analisis Time Series di bidang keuangan dan investasi. Keahliannya juga terlihat dalam penggunaan bahasa pemrograman Statistik seperti R Studio dan Python. Beliau mengaplikasikan sistem basis data MySQL/NoSQL dalam pembelajaran manajemen data, serta mahir dalam menggunakan tools Big Data seperti Spark dan Hadoop. Beberapa project beliau dapat dilihat di link berikut: Rpubs, Github, Website, dan Kaggle.

- **Trisha Magdalena Adelheid Januaviani, M.Mat** adalah seorang dosen yang sekaligus menjabat sebagai Sekerataris Program Studi Statistika di Universitas Matana. Sebagai lulusan Magister Matematika dari Universitas Padjadjaran, Beliau telah mengembangkan kompetensi terbaiknya dalam Pengajaran, Penelitian dan Pengabdian Kepada Masyarakat. Penelitiannya difokuskan pada matematika keuangan dan sains data, dua bidang yang penting dalam kehidupan saat ini. Beliau mampu memberikan wawasan berharga tentang penerapan konsep matematika dalam konteks keuangan. Kemampuannya dalam ilmu data juga memungkinkannya menganalisis dan menginterpretasi data dengan cermat untuk pengambilan keputusan yang lebih baik.
- **Yonathan Anggraiwan, S.Stat** adalah seorang alumni Statistika yang bersemangat dalam dunia pemrograman dan analisis data. Lahir di Tangerang, minatnya terhadap teknologi dan komputer muncul sejak usia dini. Ia tumbuh dengan rasa ingin tahu yang kuat terhadap bahasa pemrograman, dan ini membawanya menuju dunia analisis data menggunakan bahasa pemrograman R dan Python. Selama menjalankan tugas sebagai asisten lab, Yonathan Anggraiwan berperan dalam membantu mahasiswa dalam memahami konsep-konsep dasar dan kompleks dalam pemrograman R dan Python. Ia memberikan penjelasan yang jelas dan dukungan kepada mahasiswa yang mengalami kesulitan. Selain itu, ia juga terlibat dalam merancang tugas dan ujian praktikum, serta memberikan umpan balik konstruktif kepada para mahasiswa. Dalam perjalanan waktu, Yonathan Anggraiwan mulai mengambil tanggung jawab lebih besar dalam laboratorium. Ia membantu mengembangkan materi pembelajaran tambahan, seperti tutorial online tentang analisis data menggunakan R dan Python. Ia juga aktif dalam berbagai proyek penelitian di bawah bimbingan dosen, yang melibatkan pengolahan data besar untuk analisis statistik dan visualisasi. Dengan semangat yang tinggi, dedikasi, dan keterampilan yang dimilikinya, Yonathan Anggraiwan adalah contoh nyata dari seorang mahasiswa yang berhasil menggabungkan minatnya dalam pemrograman R dan Python dengan peran yang produktif sebagai asisten laboratorium dan kontributor dalam dunia analisis data.

Ucapan Terima Kasih

Kami berharap modul ini akan menjadi panduan yang bermanfaat bagi Anda dalam menguasai bahasa pemrograman R. Semoga dengan memahami konsep-konsep yang disajikan dalam modul ini, Anda akan dapat mengaplikasikan R dalam proyek-proyek analisis data dan statistik yang sebenarnya.

Terima kasih kepada semua yang telah berkontribusi dalam pembuatan modul ini, serta kepada Anda, pembaca, yang telah memilih modul ini sebagai sumber pengetahuan Anda. Kami berharap Anda menikmati perjalanan Anda dalam memahami bahasa pemrograman R.

Masukan & Saran

Semua masukan dan tanggapan Anda sangat berarti bagi kami untuk memperbaiki modul ini kedepannya. Bagi para pembaca/pengguna yang ingin menyampaikan masukan dan tanggapan, dipersilahkan melalui kontak dibawah ini!

Email: dsciencelabs@outlook.com

Chapter 1

Pengenalan R?

R adalah bahasa pemrograman dan lingkungan komputasi yang digunakan untuk analisis statistik, visualisasi data, pengolahan data, dan pemodelan prediktif. R dikembangkan oleh Ross Ihaka dan Robert Gentleman di Universitas Auckland, Selandia Baru. R menjadi populer dalam dunia analisis data dan ilmu data karena kemampuannya dalam mengolah dan menganalisis data secara efisien.

1.1 Fitur Utama R

1. **Open Source:** R adalah perangkat lunak open source yang dapat diunduh dan digunakan secara gratis.
2. **Fleksibilitas:** Anda dapat membuat fungsi sendiri, mengontrol alur program, dan mengakses berbagai pustaka eksternal.
3. **Mengimpor dan Mengekspor Data:** R mendukung berbagai format file, seperti CSV, Excel, SQL, dan format data lainnya.
4. **Data Manipulasi:** R memiliki pustaka seperti dplyr dan tidyverse yang memudahkan manipulasi dan transformasi data.
5. **Lingkungan Komputasi:** R tidak hanya bahasa pemrograman, tetapi juga lingkungan komputasi lengkap yang menyediakan alat untuk analisis dan visualisasi data.
6. **Statistik dan Analisis Data:** R memiliki beragam pustaka dan paket yang mendukung analisis statistik, visualisasi data, dan pemodelan prediktif.
7. **Grafik dan Visualisasi:** R memiliki kemampuan visualisasi yang kuat dengan pustaka seperti ggplot2 untuk membuat grafik yang informatif dan menarik.
8. **Komunitas Aktif:** Komunitas R sangat aktif, dan ada banyak sumber

daya online, forum, dan pustaka yang dapat membantu dalam pembelajaran dan pemecahan masalah.

1.2 Mengapa Belajar R?

Berikut ini adalah beberapa alasan mengapa penting untuk belajar R:

1. **Pengolahan Data:** R dapat membantu Anda membersihkan, merubah format, dan mengolah data sebelum analisis lebih lanjut.
2. **Analisis Data:** R adalah alat yang kuat untuk menganalisis data, membuat visualisasi yang menarik, dan mengidentifikasi pola dalam dataset.
3. **Karir di Ilmu Data:** Penguasaan R menjadi salah satu keahlian yang sangat dihargai dalam industri ilmu data dan analisis data.
4. **Komunitas Besar:** Anda akan menjadi bagian dari komunitas besar yang mendukung dan berkontribusi dalam pengembangan R serta membagikan pengetahuan.

1.3 Download R & Rstudio:

1. **Unduh dan Instalasi:** Kunjungi situs resmi R (<https://www.r-project.org/>) untuk mengunduh installer sesuai dengan sistem operasi Anda.
2. **RStudio (Opsional tapi Disarankan):** RStudio adalah lingkungan pengembangan terintegrasi (IDE) yang mempermudah pengembangan dalam R. Anda dapat mengunduh RStudio (<https://www.rstudio.com/>) dan menggunakananya untuk menulis dan menjalankan kode R.

1.4 Tutorial Instal R & Studio

Berikut adalah panduan langkah demi langkah untuk menginstal R dan RStudio

1.4.1 Instalasi R

R adalah bahasa pemrograman inti yang digunakan oleh RStudio. Ikuti langkah-langkah di bawah ini untuk menginstal R:

Windows

1. Kunjungi situs resmi R di <https://cran.r-project.org/mirrors.html>.
2. Pilih cermin (mirror) terdekat untuk mengunduh installer R.

3. Unduh installer R untuk Windows dan jalankan file installer yang diunduh.
4. Ikuti panduan instalasi, pilih opsi default kecuali jika Anda tahu persis apa yang Anda lakukan.
5. Setelah instalasi selesai, R akan terinstal di komputer Anda.

MacOS

1. Kunjungi situs resmi R di <https://cran.r-project.org/mirrors.html>.
2. Pilih cermin (mirror) terdekat untuk mengunduh installer R.
3. Unduh installer R untuk macOS dan jalankan file installer yang diunduh.
4. Ikuti panduan instalasi, pilih opsi default kecuali jika Anda tahu persis apa yang Anda lakukan.
5. Setelah instalasi selesai, R akan terinstal di komputer Anda.

Linux

Di sistem Linux, Anda dapat menggunakan perintah terminal untuk menginstal R. Berikut adalah contoh untuk beberapa distribusi umum:

Ubuntu/Debian:

Buka Program csharp anda dan run koding dibawah ini!

```
Copy code
sudo apt-get update
sudo apt-get install r-base
```

CentOS/Fedora:

Buka Program Command Prompt anda dan run koding dibawah ini!

```
sudo yum install R
```

1.4.2 Instalasi RStudio:

RStudio adalah Integrated Development Environment (IDE) yang mempermudah pengembangan dalam R. Ikuti langkah-langkah di bawah ini untuk menginstal RStudio:

Windows, macOS, dan Linux:

1. Kunjungi situs resmi RStudio di <https://www.rstudio.com/products/rstudio/download/>.

2. Pilih “RStudio Desktop” yang sesuai dengan sistem operasi Anda.
3. Unduh installer RStudio dan jalankan file installer yang diunduh.
4. Ikuti panduan instalasi dan pilih opsi default kecuali jika Anda tahu persis apa yang Anda lakukan.
5. Setelah instalasi selesai, RStudio akan terinstal di komputer Anda.

1.5 Video Instalasi R & RStudio

1.5.1 Windows

1.5.2 MacOS

1.6 Interface R & RStudio:

Interface adalah tampilan aplikasi R dan Rstudio yang telah terpasang diperlukan pada Gambar 1.1 dan Gambar 1.2.

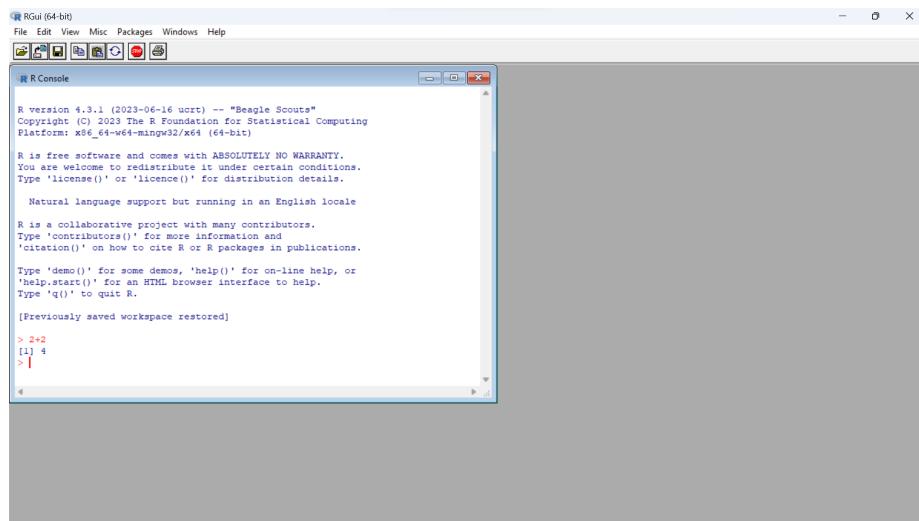


Figure 1.1: Jendela R.

Tampilan ini memiliki beberapa komponen utama, termasuk:

- **Script Panel:** Tempat Anda menulis kode R dalam skrip.
- **Console Panel:** Tempat hasil dari kode R ditampilkan, serta tempat Anda dapat menjalankan kode secara interaktif.
- **Environment Panel:** Menampilkan daftar variabel yang ada dalam sesi R Anda.

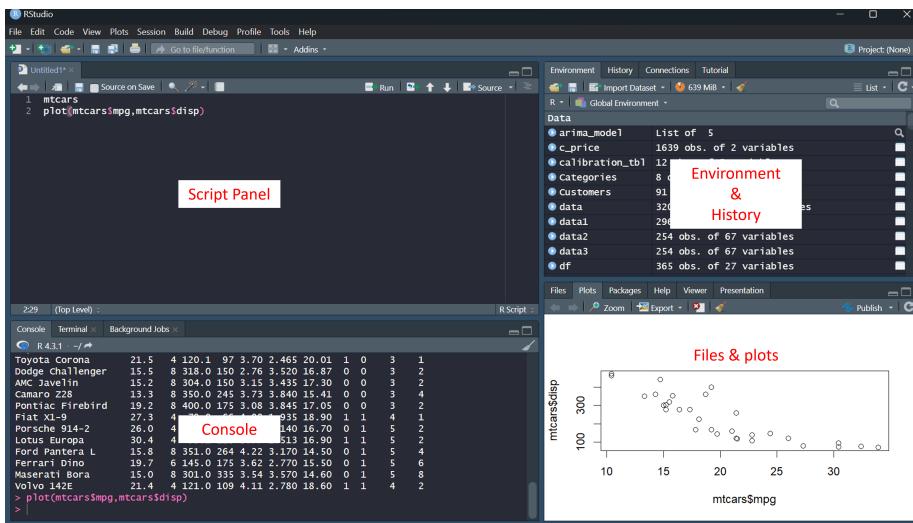


Figure 1.2: Jendela RStudio.

- **History Panel:** Menampilkan riwayat perintah yang telah dijalankan.
- **Files/Plots/Packages/Help Panel:** Panel tambahan yang membantu Anda mengelola file, visualisasi, pustaka, dan panduan bantuan.

Interface ini memudahkan pengguna untuk menulis, menjalankan, dan mengejola kode R serta menganalisis data dengan nyaman.

1.7 Sintaks Dasar R

Berikut ini beberapa kode sederhana yang bisa dipelajari untuk memulai memahami cara kerja Bahasa pemrograman R.

```
3+7
3-7
3^7
3/7
3*7
9^(1/3)
```

1.8 Bantuan (Help) R

Salah satu bagian terpenting dalam bekerja dengan bahasa R adalah mengetahui di mana mencari bantuan. R memiliki beberapa fasilitas in-line, selain berbagai

sumber daya bantuan di ekosistem R. Anda dapat menggunakan bantuan untuk fungsi tertentu.

```
help.start()          # menu di mana Anda dapat menavigasi bantuan lokal berbasis web
?help                # menu di mana Anda dapat menavigasi bantuan lokal berbasis web
?class               # mendapatkan bantuan untuk fungsi `class`
help(class)          # mendapatkan bantuan untuk fungsi `class`
??class              # jika Anda tidak tahu nama fungsi yang Anda cari
help.search('class') # jika Anda tidak tahu nama fungsi yang Anda cari
```

1.9 Shortcut Penggunaan Rstudio

Beberapa petunjuk bermanfaat untuk Rstudio (IDE) meliputi:

Kata Kunci	Perintah	Detail
Ctrl + Return (Enter)	untuk menjalankan baris dari editor	~
Ctrl + Shift + #	untuk fokus pada tab bantuan	kontradiktif
Alt + Shift + k	untuk jalur pintas keyboard RStudio	~
Ctrl + r	untuk menelusuri sejarah perintah	~
Alt + Shift + j	untuk menavigasi antar bagian kode	~
Ctrl + 1	untuk melompat ke editor	tab untuk penyelesaian otomatis
Ctrl + 2	untuk melompat ke konsol	tab untuk penyelesaian otomatis
Ctrl + 8	untuk melompat ke environment list	tab untuk pelengkapan otomatis
Alt + 1	Collapse chunk	Code Folding
Alt + Shift + l	Unfold chunk	Code Folding
Alt + o	Collapse all	Code Folding
Alt + Shift + o	Unfold all	Code Folding
Alt + “-”	untuk operator penugasan <-	~
Alt + Shift + c	kode komentar/tanda komentar dalam file	.R kontradiktif

1.10 Praktikum

Buatlah tutorial Instalasi R dan R Studio dalam M.word! Lengkapi setiap prosesnya dengan gambar dan penjelasan.

Chapter 2

Operasi Dasar R

Pemrograman R merujuk pada proses menulis kode dan mengembangkan program menggunakan bahasa pemrograman R. R adalah bahasa pemrograman yang fokus pada analisis statistik, manipulasi data, dan visualisasi. Pada bab ini akan dibahas beberapa unsur utama dalam pemrograman menggunakan bahasa pemrograman R.

2.1 Variabel

Variabel dalam bahasa pemrograman R digunakan untuk menyimpan dan mengejelaskan data. Variabel memungkinkan Anda untuk menampung nilai-nilai berbagai jenis, seperti angka, karakter (teks), atau nilai logika (benar/salah). Berikut ini adalah cara untuk mendefinisikan dan menggunakan variabel dalam R:

2.1.1 Mendefinisikan Variabel

Untuk membuat variabel, Anda cukup menggunakan tanda <- atau = untuk memberikan nilai pada variabel.

```
x <- 10          # Mendefinisikan variabel x  
y = 12           # Mendefinisikan variabel y
```

2.1.2 Aturan Nama Variabel

- Nama variabel harus dimulai dengan huruf atau tanda .
- Nama variabel bisa terdiri dari huruf, angka, dan tanda _.

- Karakter khusus seperti $+$, $-$, $*$, $/$, \wedge dll. tidak diperbolehkan dalam nama variabel.
- Nama variabel bersifat case-sensitive, artinya `x` dan `X` dianggap berbeda.

2.2 Operator

Operator adalah simbol yang mengarahkan compiler untuk melakukan berbagai macam operasi terhadap beberapa penugasan. Operator mensimulasikan berbagai operasi matematis, logika, dan keputusan yang dilakukan pada sekumpulan Bilangan Kompleks, Integer, dan Numerik sebagai penugasan masukan (input). R mendukung sebagian besar empat jenis operator biner antara satu set penugasan. Dalam ini, kita akan melihat berbagai jenis operator yang tersedia di R penggunaannya.

2.2.1 Aritmatika

Penggunaan operator aritmatika dalam program R adalah untuk mensimulasikan berbagai operasi matematika, seperti penambahan, pengurangan, perkalian, pembagian, dan modulo. Operator aritmatika yang dilakukan bisa saja berupa nilai skalar, bilangan kompleks, atau vektor.

Operator	R
Penjumlahan	<code>+</code>
Pengurangan	<code>-</code>
Perkalian	<code>*</code>
Divisi/Pembagian	<code>/</code>
Pemangkatan	<code>^</code>
Modulo	<code>%%</code>

Perhatikan cuplikan R berikut:

```
x <- c(2,3,5)      # memuat vektor x
y <- c(2,4,6)      # memuat vektor y
x+y                # hasil penjumlahan vektor x dan y
print (x+y)        # hasil penjumlahan vektor x dan y
print (x-y)        # hasil pengurangan vektor x dan y
print (x*y)        # hasil perkalian vektor x dan y
print (x/y)        # hasil pembagian vektor x dan y
print (x^y)        # hasil pemangkatan vektor x dan y
print (x%%y)        # hasil modulo vektor x dan y
```

Adakalanya anda perlu menampilkan keterangan/komentar yang juga melekat pada hasil perhitungan R itu sendiri. Maka anda dapat melakukannya dengan cara berikut:

```
cat("Penjumlahan vektor x dan y :", x + y, "\n")
cat("Pengurangan vektor x dan y :", x - y, "\n")
cat("Perkalian vektor x dan y :", x * y, "\n")
cat("Pembagian vektor x dan y :", x / y, "\n")
cat ("Pemangkatan vektor x dan y :", x ^ y)
cat("Modulo vektor x dan y :", x %% y, "\n")
```

Catatan: Penjelasan lebih lekap mengenai modulo dapat lihat pada link ini

2.2.2 Relasional

Operator relasional melakukan operasi perbandingan antara elemen yang bersesuaian pada setiap operan. Mengembalikan nilai Boolean TRUE jika operan pertama memenuhi relasi dibandingkan dengan operan kedua. Nilai TRUE selalu dianggap lebih besar dari FALSE.

Operator	R	Keterangan
Kurang dari	<	Mengembalikan TRUE jika elemen yang bersesuaian pada operan pertama lebih kecil dari operan kedua. Selain itu akan mengembalikan FALSE
Kurang dari sama dengan	<=	Mengembalikan TRUE jika elemen yang bersesuaian pada operan pertama kurang dari atau sama dengan elemen operan kedua. Selain itu akan mengembalikan FALSE
Lebih besar dari	>	Mengembalikan TRUE jika elemen yang bersesuaian pada operan pertama lebih besar dari operan kedua. Selain itu akan mengembalikan FALSE

Lebih besar dari sama dengan	$>=$	Mengembalikan BENAR jika elemen yang bersesuaian pada operan pertama lebih besar atau sama dengan dari operan kedua. Selain itu akan mengembalikan FALSE
Sama Dengan	$==$	Mengembalikan BENAR jika dan hanya jika kedua sisi bernilai sama
Tidak Sama dengan	$!=$	Mengembalikan BENAR jika elemen yang bersesuaian pada operan pertama tidak sama dengan dari operan kedua

```
x <- c(2,3,5)    # memuat vektor x
y <- c(2,4,6)    # memuat vektor y
cat("Vektor x kurang dari Vektor y:", x < y, "\n")
cat("Vector x kurang dari sama dengan Vector y:", x <= y, "\n")
cat("Vector x lebih besar dari Vector y :", x > y, "\n")
cat("Vector x lebih besar dari sama dengan Vector y :", x >= y, "\n")
cat("Vector x sama dengan Vector y:", x == y, "\n")
cat("Vector x tidak sama dengan Vector y:", x != y)
```

2.2.3 Logika

Operator logis mensimulasikan operasi keputusan, berdasarkan operator yang ditentukan antara operan, yang kemudian dievaluasi ke nilai Boolean Benar atau Salah. Nilai bilangan bulat bukan nol dianggap sebagai nilai BENAR, baik itu bilangan kompleks atau bilangan real.

Operator	R	Keterangan
NOT	!	Operasi negasi/kebalikan pada status elemen operan
AND	&	Mengembalikan TRUE jika kedua operan bernilai Benar
OR		Mengembalikan TRUE jika salah satu operan adalah Benar

XOR	\wedge	Mengembalikan TRUE jika salah satu dari kedua elemen pertama operan bernilai Benar
-----	----------	--

```
x <- c(0,TRUE,FALSE)
y <- c(TRUE,0.1,4+3i)

# Melakukan operasi logika pada Operan
cat("Logika Negasi (~) untuk vektor x:", !x, "\n")
cat("Logika Negasi (~) untuk vektor y :", !y, "\n")
cat ("Logika Konjungsi (Dan) :", x & y, "\n")
cat ("Logika Disjungsi (Atau) :", x | y, "\n")
```

2.2.4 Operator Lain-lain

Berikut ini juga ada beberapa operator yang kemungkinan besar juga akan anda perlukan pada saat akan menggunakan R.

```
x <- c(2,3,5)                      # memuat vektor x
y <- c(2,4,6)                      # memuat vektor y
sqrt(x*y)                           # Bentuk akar
log(x)                               # logaritma
exp(y)                               # eksponen
(x/y) + y                            # Tanda kurung
```

Catatan: Sifat Komutatif Asosiatif dan Distributif juga berlaku dalam program R.

2.3 Tipe Data

Dalam pemrograman seperti R dan Python, tipe data merupakan konsep penting. Keduanya dapat menggunakan variabel untuk menyimpan tipe yang berbeda-beda, berikut adalah tipe data paling mendasar yang harus diketahui:

Tipe Data	R	Penjelasan
Double/Float	5.6	Bilangan yang mempunyai koma
Integer	5	Bilangan bulat 1,2,...,n

Bolean/Logical	TRUE/FALSE	Benar bernilai 1 dan Salah bernilai 0
String/Character	'Dscienclabs'	karakter/kalimat bisa berupa huruf angka, dll (diapit tanda " atau ')
Complex	1 + 5i	Pasangan angka real dan imajiner

Berikut ini adalah coding R yang dapat digunakan untuk menetapkan kelima tipe data diatas:

```
d1 = 5.6                                # Tetapkan nilai desimal
d2 = as.integer(5)                         # tetapkan nilai integer
d2 = 5L                                     # cara lain untuk memuat nilai integer di R
d3 = c(TRUE,FALSE)                         # Boolean/Logical
d3 = as.logical(c(0,1))                     # cara lain untuk memuat Boolean/Logical
d4 = c("a",'b','123')                      # String/Character
d5 = 1 + 5i                                 # Complex
```

Untuk memeriksa tipe data dalam R:

```
class(d1)                               # cetak nama kelas variabel
typeof(d1)                               # cetak tipe variabel x
```

2.4 Statistika Dasar

```
data <- c(85, 90, 78, 92, 88)    # Data
sum(data)                           # Jumlahan data
length(data)                         # Banyaknya data
mean (data)                          # Menghitung rata-rata
var (data)                           # Menghitung variansi
sd (data)                            # Simpangan baku
min(data)                            # Minimum
max(data)                            # Maksimum
```

2.5 Contoh Pemrograman Dasar

Berikut ini dilampirkan contoh kasus dasar pemrograman R.

2.5.1 Menghitung Rata-rata

```
data <- c(85, 90, 78, 92, 88)
jumlah_data <- length(data)
total <- sum(data)
rata_rata <- total / jumlah_data
print(paste("Rata-rata:", rata_rata))

## [1] "Rata-rata: 86.6"
```

2.5.2 Membandingkan Angka

```
a <- 10
b <- 20

if (a < b) {
  print("a lebih kecil dari b")
} else if (a > b) {
  print("a lebih besar dari b")
} else {
  print("a sama dengan b")
}
```

```
## [1] "a lebih kecil dari b"
```

2.5.3 Membandingkan Karakter

```
kata1 <- "Apel"
kata2 <- "apel"

if (kata1 == kata2) {
  print("kata1 sama dengan kata2")
} else {
  print("kata1 berbeda dari kata2")
}
```

```
## [1] "kata1 berbeda dari kata2"
```

2.5.4 Mengecek Kondisi Gabungan

```

umur <- 25
pendapatan <- 5000

if (umur > 18 && pendapatan > 4000) {
  print("Anda memenuhi syarat")
} else {
  print("Anda tidak memenuhi syarat")
}

## [1] "Anda memenuhi syarat"

```

2.5.5 Penggunaan Operator Logika

```

x <- 5
y <- 10

if (x > 0 || y > 0) {
  print("Salah satu variabel positif")
} else {
  print("Kedua variabel non-positif")
}

## [1] "Salah satu variabel positif"

```

2.5.6 Pemeriksaan Kondisi dengan ifelse()

```

nilai <- 75
keterangan <- ifelse(nilai >= 70, "Lulus", "Tidak lulus")
print(paste("Nilai:", nilai, "--> Status:", keterangan))

## [1] "Nilai: 75 --> Status: Lulus"

```

2.6 Latihan

Berikut adalah beberapa contoh soal latihan dasar pemrograman dalam bahasa pemrograman R:

1. **Menghitung Luas Lingkaran:** Buatlah sebuah program R yang menerima input berupa jari-jari lingkaran dan menghitung serta mencetak luas lingkaran.
2. **Konversi Suhu:** Buatlah sebuah program R yang dapat mengonversi suhu dari Celsius ke Fahrenheit. Program harus menerima input suhu dalam Celsius dan menghasilkan output suhu dalam Fahrenheit.
3. **Menghitung Faktorial:** Buatlah sebuah program R yang menghitung faktorial dari sebuah bilangan bulat positif. Program harus menerima input bilangan bulat positif dan menghasilkan output faktorialnya.
4. **Mencari Bilangan Prima:** Buatlah sebuah program R yang menerima input sebuah bilangan bulat dan menghasilkan output apakah bilangan tersebut merupakan bilangan prima atau bukan.
5. **Menghitung Pangkat:** Buatlah sebuah program R yang menerima input bilangan dasar dan eksponen, kemudian menghitung hasil dari bilangan dasar dipangkatkan dengan eksponen tersebut.
6. **Menghitung Total Nilai:** Buatlah sebuah program R yang menerima input sejumlah nilai mata kuliah dan menghitung total nilai serta rata-ratanya. Program harus menerima input nilai-nilai mata kuliah dan menghasilkan total nilai serta rata-ratanya.
7. **Menentukan Ganjil/Genap:** Buatlah sebuah program R yang menerima input bilangan bulat dan mencetak apakah bilangan tersebut ganjil atau genap.
8. **Menghitung Keliling dan Luas Persegi:** Buatlah sebuah program R yang menerima input panjang sisi persegi dan menghitung serta mencetak keliling dan luasnya.

Chapter 3

Struktur Data

Struktur data dalam R adalah cara di mana Anda dapat mengatur dan menyimpan data dalam bentuk yang terstruktur agar mudah diakses, dikelola, dan dimanipulasi. Struktur data memungkinkan Anda untuk mengelompokkan nilai-nilai data ke dalam objek yang sesuai dengan jenis dan sifat data yang Anda miliki. R memiliki beberapa jenis struktur data yang dapat digunakan untuk berbagai tujuan.

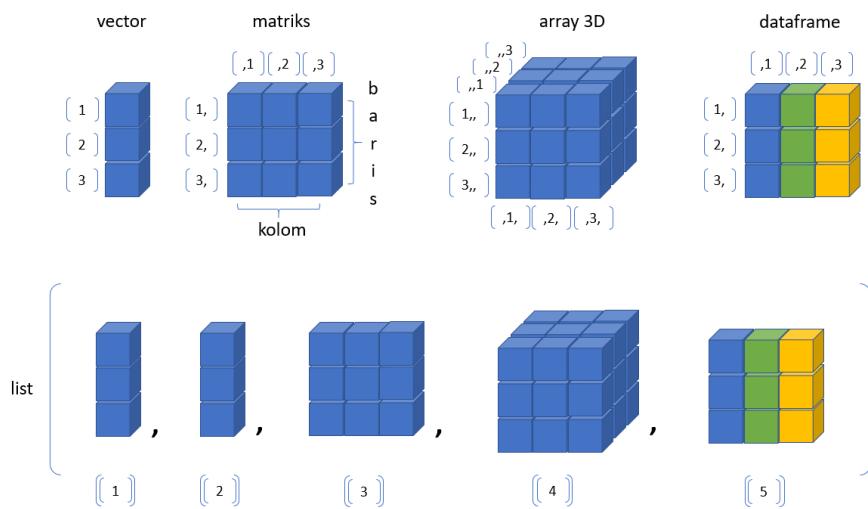


Figure 3.1: Struktur Data dalam R

3.1 Vektor

Elemen paling dasar dalam R adalah vektor, yang berisikan kumpulan elemen data dengan tipe yang sama. Terdapat dua jenis vector, yaitu vector numerik dan vector karakter. Misalnya:

```

vektor <- seq(from=10, to=21, by=1)      # fungsi `seq()` dengan "by"
vektor<- seq(from=10, to=21, len=12)      # fungsi `seq()` dengan "len"
vektor <- 10:21                           # tetapkan data dalam vektor
vektor <- vektor+2                         # Operasi berdasarkan elemen
vektor <- vektor*2                         # Tambahkan 2 untuk setiap elemen
vektor <- vektor^2                         # Pangkat 2 untuk setiap elemen
vektor <- sqrt(vektor)                     # Akar kuadrat untuk setiap elemen
vektor <- log(vektor)                      # Logaritma untuk setiap elemen
vektor<- c(0.5, 0.6)                      # Numerik
vektor <- c(TRUE, FALSE)                   # Logis
vektor <- c(T, F)                          # Logis
vektor <- c("a", "b", "c")                 # Karakter
vektor <- 9:29                            # Integer
vektor <- c(1+0i, 2+4i)                   # Kompleks
vektor <- vector("numeric", length = 10)   # untuk inisialisasi vektor.

```

Catatan: Menurut dokumentasi R untuk `typeof()` dan `class()`, pernyataan tentang “perbedaan utama/main difference” adalah tidak benar. Kelas adalah atribut dari objek yang dapat ditetapkan terlepas dari mode penyimpanan internalnya, sedangkan `typeof()` menentukan tipe (R internal) atau mode penyimpanan dari objek apa pun. Salah satu menggambarkan karakteristik logis sedangkan yang lain adalah karakteristik fisik dari suatu objek.

```

class(vektor)                      # Periksa kelas vektor
as.numeric(vektor)                  # Menetapkan vektor sebagai numerik
as.logical(vektor)                  # Menetapkan vektor sebagai logis
as.character(vektor)                # Menetapkan vektor sebagai karakter
as.numeric(c(FALSE,TRUE,TRUE,FALSE)) # Menetapkan vektor logis sebagai angka

```

Terkadang, R tidak dapat menemukan cara untuk memaksa suatu objek dan ini dapat menghasilkan NA.

```

vektor <- c("a", "b", "c","1")        # menetapkan nilai vektor
as.numeric(vektor)                    # menetapkan vektor sebagai numerik
                                         ## Warning: NAs introduced by coercion

```

```
as.logical(vektor)                      # menetapkan vektor sebagai logis
as.complex(vektor)                      # menetapkan vektor sebagai karakter

## Warning: NAs introduced by coercion
```

Catatan: Saat paksaan tidak masuk akal terjadi, Anda biasanya akan mendapat peringatan dari R.

Kita sudah melihat bahwa elemen dasar dari objek R adalah vektor. Vektor dapat ditetapkan dengan berbagai jenis berikut:

- **character:** di mana setiap elemen adalah string, mis., urutan simbol alfanumerik.
- **numeric:** di mana setiap elemen adalah bilangan real dalam format floating point ganda.
- **integer:** di mana setiap elemen adalah integer.
- **logis:** di mana setiap elemen adalah TRUE, FALSE, atau NA3
- **complex:** di mana setiap elemen adalah bilangan kompleks.

3.2 Matriks

Matriks adalah vektor dengan atribut dimensi. Matriks dibuat berdasarkan kolom, sehingga entri dapat dianggap dimulai dari sudut “kiri atas” dan mengejutkan di kolom.

```
matriks <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3)
matriks
```

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

Matriks juga dapat dibuat langsung dari vektor dengan menambahkan atribut dimensi.

```
matriks <- 1:6                                # Membuat vektor
dim(matriks) <- c(2, 3)                      # rubah vektor sebagai matriks sebesar 2x3
matriks                                     # Mencetak hasilnya
```

Matriks dapat dibuat dengan pengikatan kolom atau pengikatan baris dengan fungsi `cbind()` dan `rbind()`.

```
x <- 1:3                      # Membuat vektor `x`
y <- 10:12                     # Membuat vektor `y`
cbind(x, y)                   # Menggabungkan vektor `x` dan `y` dengan kolom
rbind(x, y)                   # Menggabungkan vektor `x` dan `y` dengan baris
```

3.3 Array

Array mirip dengan matrix, tetapi dapat memiliki lebih dari dua dimensi. Masing-masing dimensi dalam array memiliki ukuran tertentu.

```
array_data <- array(c(1, 2, 3, 4, 5, 6), dim = c(2, 3, 1))
```

3.4 Faktor

Faktor-faktor digunakan untuk mewakili data kategorikal dan dapat menjadi tidak teratur atau teratur. Orang dapat menganggap faktor sebagai vektor integer di mana setiap integer memiliki label. Menggunakan faktor dengan label lebih baik daripada menggunakan bilangan bulat karena faktor menggambarkan diri sendiri. Memiliki variabel yang memiliki nilai “Laki-laki” dan “Perempuan” lebih baik daripada variabel yang memiliki nilai 1 dan 2. Objek-objek dapat dibuat dengan fungsi `faktor()`.

```
x <- factor(c("yes", "no", "yes", "no"))  # Membuat objek faktor
x
table(x)                                # Cetak hasilnya
unclass(x)                               # Tabel dari `x`
attr(x, "levels")                         # Melihat representasi faktor yang mendasarinya
                                         # Melihat representasi faktor yang mendasarinya
```

3.5 Data Frame

Kerangka data (data frame) adalah tabel atau struktur mirip array dua dimensi di mana setiap kolom berisi nilai satu variabel dan setiap baris berisi satu set nilai dari setiap kolom.

Berikut ini adalah karakteristik data frame.

- Nama kolom tidak boleh kosong;
- Nama baris harus unik;
- Data yang disimpan dalam data frame bisa dari numerik, faktor atau tipe karakter;

- Setiap kolom harus berisi jumlah item data yang sama.

```
# Buat data frame pertama.
df1 <- data.frame(id = c(1:5),
                   name = c("Julian", "Vanessa", "Jeffry", "Angel", "Nikki"),
                   salary = c(623.3, 515.2, 611.0, 729.0, 843.25),
                   start_date = as.Date(c("2022-01-01", "2022-09-23", "2022-11-15",
                                         "2022-01-01", "2022-09-23")),
                   dept = c("DS", "DS", "BA", "DA", "DS"), stringsAsFactors = F)
df1

# Buat data frame kedua.
df2 <- data.frame(id = c(6:10),
                   name = c("Ardifo", "Irene", "Kefas", "Sherly", "Bakti"),
                   salary = c(578.0, 722.5, 632.8, 632.8, NA),
                   start_date = as.Date(c("2022-05-21", "2022-07-30", "2022-06-17",
                                         "2022-07-30", "2018-09-03")),
                   dept = c("Actuaries", "Actuaries", "CA", "DE", "Lecturer"), stringsAsFactors = F)
df2

df3 <- rbind(df1, df2)                                # Gabungkan dua frame data
print(df3)                                            # Cetak hasilnya `df3`
head(df3)                                             # Cetak enam baris pertama
head(df3, 6)                                           # Cetak enam baris pertama
#View(df3)                                            # Menggunakan RStudio seperti penampil Excel
class(df3)                                            # objeknya bertipe data.frame
str(df3)                                              # Dapatkan struktur data frame
dim(df3)                                               # Periksa dimensi data
```

Data frame biasanya dibuat dengan membaca dalam dataset menggunakan `read.table()` atau `read.csv()`. Namun, data frame juga dapat dibuat secara eksplisit dengan fungsi `data.frame()` atau mereka dapat dipaksakan dari jenis objek lain seperti list.

3.6 Lists

List dalam R adalah struktur data yang mengizinkan Anda untuk menyimpan berbagai jenis objek, termasuk vektor, matriks, array, dataframe, dan objek list lainnya, dalam satu objek tunggal. Ini memungkinkan Anda untuk membuat struktur data yang kompleks dan fleksibel dengan menggabungkan objek-objek yang berbeda ke dalam satu wadah. List sering digunakan ketika Anda perlu mengorganisir dan mengelompokkan objek-objek yang terkait.

Berikut adalah contoh penggunaan dan pembuatan list dalam R:

```

# Membuat vektor dan matriks
vektor <- c(1.5, 2.7, 3.2, 4.0)
matriks <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3)

# Membuat dataframe
data_frame <- data.frame(name = c("Alice", "Bakti", "Charlie"),
                          age = c(25, 30, 28),
                          score = c(95, 88, 76))

faktor <- "List, Sudah Jadi"

# Membuat list
my_list <- list(vektor, matriks, data_frame, faktor)

# Menampilkan list
print(my_list)

```

Anda juga dapat memberi nama pada setiap elemen dalam list untuk membuat list yang lebih mudah dibaca:

```

nama_list <- list(elemen1 = vektor,
                   elemen2 = matriks,
                   elemen3 = data_frame,
                   elemen4 = faktor)

# Menampilkan elemen dalam list berdasarkan nama
print(nama_list$elemen1)

## [1] 1.5 2.7 3.2 4.0

print(nama_list$elemen2)

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

print(nama_list$elemen3)

##      name age score
## 1 Alice 25    95
## 2 Bakti 30    88
## 3 Charlie 28    76

```

```
print(nama_list$elemen4)
```

```
## [1] "List, Sudah Jadi"
```

Anda dapat mengakses elemen-elemen dalam list menggunakan indeks atau nama. Misalnya:

```
# Mengakses elemen pertama dalam list menggunakan indeks
elemen1 <- my_list[[1]]
elemen2 <- my_list[[2]]
elemen3 <- my_list[[2]]
elemen4 <- my_list[[2]]

# Menampilkan hasil
print(elemen1)
print(elemen2)
print(elemen3)
print(elemen4)
```

List memungkinkan Anda mengorganisir, mengelompokkan, dan mengakses objek-objek yang beragam dalam struktur data tunggal, sehingga sangat berguna dalam analisis data yang kompleks dan beragam.

3.7 Rekayasa Data Frame

3.7.1 Tanpa Packages

Sebagai seorang Data Scientist, ketika mencoba menyimulasikan proses analisis data, pemodelan, bahkan prediksi. Anda harus mampu secara intuitif membangun dataframe untuk memperkirakan kumpulan data sampel. Terutama, ketika Anda tidak memiliki kumpulan data sampel sama sekali. Oleh karena itu, pada bagian ini, kita akan belajar sedikit mengenai cara menghasilkan dataframe. Harap perhatikan baik-baik contoh berikut:

```
# Misalkan Anda ingin membangun kumpulan data karyawan di sebuah perusahaan

No<-(1:52)                                # Menghasilkan bilangan 1-52
Name<-c(LETTERS,letters)                  # 26 LETTERS dan 26 letters
Gender<-sample(rep(c("Male","Female"),times=26)) # 26 Laki-laki dan 26 perempuan

# Menghasilkan tanggal lahir
year_in_3<-seq(as.Date("2000/01/01"), by="year", length.out=4)
```

```

Birthday <- rep(year_in_3, times=13)

# Menghasilkan kategori universitas
univ1<-rep("National",times=26)                                # 26 universitas negeri
univ2<-rep("Private",times=16)                                 # 16 universitas swasta
univ3<-rep("Overseas",times=10)                                # 10 universitas luar negeri
Universities<-sample(c(univ1,univ2,univ3))                  # Menggabungkan data (vetor)

gpa<-runif(52,min=3.00,max=4.00)                               # Menghasilkan 52 bilangan acak (min=3.00, max=4.00)
GPA<-round(gpa,digits=2)                                         # Mengatur digit bilangan acak Anda
Salary<-sample(600:1200,52,replace=T)                            # Menghasilkan sampel antara 600-1200
Employees<-data.frame(No,
                      Name,
                      Birthday,
                      Gender,
                      Universities,
                      GPA,
                      Salary)
Employees

```

##	No	Name	Birthday	Gender	Universities	GPA	Salary
## 1	1	A	2000-01-01	Male	National	3.05	630
## 2	2	B	2001-01-01	Male	National	3.61	851
## 3	3	C	2002-01-01	Male	Overseas	3.42	600
## 4	4	D	2003-01-01	Female	Private	3.49	857
## 5	5	E	2000-01-01	Female	National	3.17	865
## 6	6	F	2001-01-01	Male	National	3.39	1134
## 7	7	G	2002-01-01	Female	National	3.77	756
## 8	8	H	2003-01-01	Male	Private	3.38	635
## 9	9	I	2000-01-01	Male	National	3.24	1040
## 10	10	J	2001-01-01	Female	National	3.62	969
## 11	11	K	2002-01-01	Female	Private	3.42	1162
## 12	12	L	2003-01-01	Female	Private	3.32	1190
## 13	13	M	2000-01-01	Female	Private	3.09	819
## 14	14	N	2001-01-01	Female	National	3.22	612
## 15	15	O	2002-01-01	Male	National	3.41	974
## 16	16	P	2003-01-01	Female	Overseas	3.70	777
## 17	17	Q	2000-01-01	Male	Private	3.36	612
## 18	18	R	2001-01-01	Female	Overseas	3.07	615
## 19	19	S	2002-01-01	Female	National	3.41	1103
## 20	20	T	2003-01-01	Male	National	3.60	1093
## 21	21	U	2000-01-01	Male	Private	3.22	897
## 22	22	V	2001-01-01	Male	National	3.66	999
## 23	23	W	2002-01-01	Male	National	3.82	777
## 24	24	X	2003-01-01	Female	National	3.96	1187

```

## 25 25      Y 2000-01-01   Male    National 3.18    809
## 26 26      Z 2001-01-01 Female   National 3.45    664
## 27 27      a 2002-01-01   Male    Private 3.48    738
## 28 28      b 2003-01-01   Male    Private 3.47    610
## 29 29      c 2000-01-01   Male    National 3.56   1177
## 30 30      d 2001-01-01 Female   Overseas 3.53   902
## 31 31      e 2002-01-01 Female   National 3.07   974
## 32 32      f 2003-01-01   Male    Private 3.26    840
## 33 33      g 2000-01-01   Male    National 3.40    832
## 34 34      h 2001-01-01   Male    Overseas 3.43   915
## 35 35      i 2002-01-01 Female   Private 3.03   1100
## 36 36      j 2003-01-01   Male    Private 3.92   1130
## 37 37      k 2000-01-01   Male    Overseas 3.31   894
## 38 38      l 2001-01-01 Female   Overseas 3.92   1045
## 39 39      m 2002-01-01 Female   Private 3.11    683
## 40 40      n 2003-01-01 Female   Overseas 3.72    709
## 41 41      o 2000-01-01 Female   National 3.99   1112
## 42 42      p 2001-01-01 Female   Private 3.92   1022
## 43 43      q 2002-01-01   Male    National 3.12    821
## 44 44      r 2003-01-01   Male    National 3.86    905
## 45 45      s 2000-01-01   Male    Overseas 3.54   1164
## 46 46      t 2001-01-01   Male    Private 3.19    759
## 47 47      u 2002-01-01 Female   Overseas 3.62   1148
## 48 48      v 2003-01-01 Female   National 3.98   1122
## 49 49      w 2000-01-01 Female   Private 3.01    878
## 50 50      x 2001-01-01 Female   National 3.49    922
## 51 51      y 2002-01-01 Female   National 3.94    932
## 52 52      z 2003-01-01   Male    National 3.24   892

```

3.7.2 Menggunakan Packages

Dalam contoh kedua ini, digunakan pustaka (*Packages*) `faker` untuk menghasilkan data palsu seperti nama, alamat, dan lain-lain. Pastikan Anda telah menginstal pustaka tersebut menggunakan perintah `install.packages("fakir")` jika belum terinstal, mengikuti langkah berikut.

```

install.packages("remotes")
remotes::install_github("ThinkR-open/fakir")

```

Selanjutnya, anda dapat membuat data frame palsu seperti diperlihatkan berikut:

```

library(fakir)
fake_ticket_client(vol = 10)

```

```
## # A tibble: 10 x 25
##   ref      num_client first last  job    age region
##   <chr>     <chr>      <chr> <chr> <chr> <dbl> <chr>
## 1 DOSS-AMQN~ 79        Jovan O'Ke~ Gene~    22 Franc~
## 2 DOSS-NCKJ~ 69        Miss  Lean~ Emer~    68 Breta~
## 3 DOSS-GPBE~ 120       Odell Stok~ Engi~    24 Bourg~
## 4 DOSS-GRLN~ 31        Loren Lars~ <NA>    NA Rhône~
## 5 DOSS-LEPJ~ 59        Mayb~ Maye~ Furt~    18 Rhône~
## 6 DOSS-DUCL~ 118       Jama~ Ober~ Engi~    18 <NA>
## 7 DOSS-OCED~ 77        Lee   Scha~ Admi~    NA Rhône~
## 8 DOSS-KXSJ~ 65        Deme~ Auer  Cont~    21 Lorra~
## 9 DOSS-UITD~ 141       Wilf~ Harv~ Educ~    53 Centre
## 10 DOSS-SHKL~ 182      Addy~ Nien~ Earl~   65 Poito~
## # i 18 more variables: id_dpt <chr>,
## #   departement <chr>, cb_provider <chr>, name <chr>,
## #   entry_date <dttm>, fidelity_points <dbl>,
## #   priority_encoded <dbl>, priority <fct>,
## #   timestamp <date>, year <dbl>, month <dbl>,
## #   day <int>, supported <chr>,
## #   supported_encoded <int>, type <chr>, ...
```

Catatan: Pustaka fakir Menyimpan beberapa dataset didalamnya, antara lain:

```
fake_products(10)                                # Rekayasa Data Produk
fake_visits(from = "2017-01-01", to = "2017-01-31") # Pengunjung Website
fake_sondage_answers(n = 10)                      # Kuisioner transfortasi
```

3.8 Latihan

1. Buatlah Rekayasa dataframe Mahasiswa dengan empat kolom: “Nama”, “Usia”, “Kota”, dan “Nilai”. Sebanyak 100 baris, dengan syarat tidak boleh ada nama yang sama.
2. Buatlah Rekayasa dataframe Karyawan dengan tujuh kolom: “No”, “Name”, “Birthday”, “Gender”, “Universities”, “GPA”, “Salary”. Sebanyak 100 baris, dengan syarat tidak boleh ada nama yang sama.
3. Buatlah Rekayasa dataframe pengunjung Website, sebanyak 200 baris.

Catatan: Kumpulkan hasil latihan anda, tidak boleh sama dengan teman ma-hasiswa lainnya.

Chapter 4

Ekstraksi Data

Ekstraksi data mengacu pada proses mengambil sebagian atau elemen tertentu dari struktur data seperti vektor, matriks, dataframe, atau list.

4.1 Impor Dataset

Berikut ini digunakan dataset yang sudah ada didalam R.

```
library(stima)                      # Pustaka Dataset
View(employee)                      # Data employee dari Pustaka stima
typeof(employee)                   # Memeriksa jenis data
class(employee)                     # Memeriksa kelas data
```

4.2 Ekstraksi Baris

Ekstraksi baris berdasarkan indeks dapat dilakukan dengan operator [] .

```
employee[1,5]                         # ekstrak elemen di baris 1 dan kolom 5
employee$gender                        # Mengekstrak kolom tertentu (`Gender`)
employee[,c('jobcat','gender')]       # Mengekstrak kolom tertentu (jobcat, gender)
employee[1:5,]                          # Mengekstrak lima baris pertama dari employee
employee[,1:5]                         # Mengekstrak lima kolom pertama dari employee
employee$edu >12                       # Subset kolom dengan kondisi
employee[employee$edu>12 & employee$salary>40200,] # Subset Kondisi Kombinasi indeks
```

4.3 Ekstraksi Subset

Ekstraksi subset mengacu pada proses mengambil sebagian data dari suatu struktur data berdasarkan kriteria tertentu. Ini memungkinkan Anda untuk fokus pada subset data yang relevan untuk analisis atau tugas tertentu tanpa harus memanipulasi atau mengakses keseluruhan data. Subset dapat diambil berdasarkan kondisi tertentu, indeks, atau kombinasi keduanya.

Dalam R, Anda dapat melakukan ekstraksi subset menggunakan berbagai metode tergantung pada jenis struktur data yang Anda gunakan. Beberapa teknik umum untuk ekstraksi subset adalah:

```
subset(employee, select='jobcat')      # subset kolom tertentu
subset(employee, select=6)             # subset kolom tertentu
subset(employee, select= c(6,7))       # subset kolom pertama dan kedua
subset(employee, select= c(2:5))       # subset kolom tertentu
```

4.4 Ekstraksi Dengan Variabel Baru

Menambah variabel baru ke dalam dataframe dapat dilakukan dengan menempatkan vektor baru. Kekuatan objek dataframe adalah menerima hampir semua jenis vektor, mis. integer, numerik, logika, faktor, dan karakter.

4.5 Ekstraksi Summary Statistik

```
min(employee$salary)          # Temukan nilai minimum dari `Salary`
max(employee$salary)          # Temukan nilai maksimum dari `Salary`
mean(employee$salary)         # Temukan nilai rata-rata dari `Salary`
var(employee$salary)          # Temukan nilai variansi dari `Salary`
sd(employee$salary)           # Temukan nilai standar deviasi dari `Salary`
summary(employee)             # Ringkasan statistik sederhana dari `employee`
```

Catatan: Berhati-hatilah saat mengekstrak kumpulan data yang berisi nilai yang hilang, jangan lupa mengabaikannya, atau menghapusnya terlebih dahulu. Lihat contoh berikut:

```
#View(mtcars)                  # Lihat dataset `mtcars` (environment R)
#?mtcars
min(mtcars$mpg)               # Temukan minimum gallon Miles/(US)

## [1] 10.4

max(mtcars$mpg , na.rm = TRUE) # Temukan maksimum gallon Miles/(US)

## [1] 33.9

mean(mtcars$mpg , na.rm = TRUE) # Temukan rata-rata gallon Miles/(US)

## [1] 20.09

var(mtcars$mpg , na.rm = TRUE)  # Temukan varians gallon Miles/(US)

## [1] 36.32

sd(mtcars$mpg , na.rm = TRUE)   # Temukan standar deviasi gallon Miles/(US)

## [1] 6.027
```

4.6 Mengubah Nama

Objek R dapat memiliki nama, yang sangat berguna untuk menulis kode yang dapat dibaca dan menggambarkan objek sendiri. Di sini, Anda akan belajar cara mengganti nama kolom dari dataframe pada R menggunakan fungsi `names()`.

```

rename_1<-employee[,1:5]                      # Mengubah nama variabel
names(rename_1)<-c("gaji",
                  "usia",
                  "gaji_awal",
                  "jam_kerja",
                  "pengalaman_kerja")# Mengubah nama variabel menjadi Indonesia
rename_1                                         # Cetak hasilnya

```

Jika dataframe Anda dihasilkan dari fungsi `matrix()`, Anda dapat mengubah nama kolom dan barisnya.

```

m <- matrix(1:52, nrow = 26, ncol = 2)
dimnames(m) <- list(c(LETTERS), c("AA", "BB"))
m

##      AA BB
## A   1 27
## B   2 28
## C   3 29
## D   4 30
## E   5 31
## F   6 32
## G   7 33
## H   8 34
## I   9 35
## J  10 36
## K  11 37
## L  12 38
## M  13 39
## N  14 40
## O  15 41
## P  16 42
## Q  17 43
## R  18 44
## S  19 45
## T  20 46
## U  21 47
## V  22 48
## W  23 49
## X  24 50
## Y  25 51
## Z  26 52

```

Nama kolom dan nama baris dapat diatur secara terpisah menggunakan fungsi `colnames()` dan `rownames()`.

```
colnames(m) <- c("Column 1", "Column 2") # Mengubah nama kolom
rownames(m) <- c(letters)                 # Mengubah nama baris
m

##   Column 1 Column 2
## a      1      27
## b      2      28
## c      3      29
## d      4      30
## e      5      31
## f      6      32
## g      7      33
## h      8      34
## i      9      35
## j     10      36
## k     11      37
## l     12      38
## m     13      39
## n     14      40
## o     15      41
## p     16      42
## q     17      43
## r     18      44
## s     19      45
## t     20      46
## u     21      47
## v     22      48
## w     23      49
## x     24      50
## y     25      51
## z     26      52
```

4.7 Latihan

Gunakan berikut ini:

```
data_frame <- fake_products(100)           # Rekayasa Data Produk
View(data_frame)
```

1. Lakukan Proses ekstraksi data seperti materi diatas!
2. Hintunglah Jumlah produk yang berasal dari suatu Negara (Contoh: “Taiwan”)!
3. Hitunglah Jumlah Produk yang ada dalam dataset tersebut!

Catatan: Kumpulkan hasil latihan anda, tidak boleh sama dengan teman ma-hasiswa lainnya.

Chapter 5

Fungsi dalam R

Suatu fungsi (*function*) dalam lingkungan pemrograman adalah satu set instruksi untuk melaksanakan tugas-tugas tertentu. Seorang programmer membangun sebuah fungsi untuk menghindari pengulangan tugas yang sama atau mengurangi kompleksitas.

Dari gambar diatas diperlihatkan bahwa:

- Jika $x = 5$ adalah masukan (*input*)
- Diberikan instruksi (fungsi) $f(x) = x + 3$, yang disebut badan program (*body*)
- Maka $y = 8$ adalah keluaran (*output*)

Sehingga, komponen yang harus terkandung didalam fungsi adalah:

- f adalah **Nama Fungsi** yang digunakan untuk menjalankan fungsi (perintah) pada program tertentu.
- x adalah **Masukan**, tetapi mungkin saja tidak ada argumen.
- $x + 3$ adalah **Badan Program** yang mendefinisikan fungsi yang dilakukan.
- **Keluaran** adalah perintah pengembalian satu atau lebih nilai dan mungkin saja tidak memuat pengembalian nilai.

Dalam berbagai kesempatan, programer diharapkan untuk mampu membangun fungsi (algoritma) sendiri dikarenakan tugas tertentu tidak dapat diselesaikan dengan fungsi yang sudah ada atau tidak ditemukannya fungsi dalam bahasa pemograman yang sedang digunakan. Pembentukan fungsi ini akan dibagi menjadi dua, yaitu fungsi dengan satu-argumen dan fungsi dengan multi-argument.

5.1 Fungsi Satu Argumen

Fungsi menerima nilai dan mengembalikan kuadrat dari suatu nilai. Berikut ini, dilampirkan struktur penulisan suatu fungsi dengan satu argumen secara garis besar:

```
nama.fungsi <- function(argumen)
{
  perhitungan perintah yang dilakukan terhadap argumen
  beberapa kode lain
}
```

5.1.1 Fungsi Kuadrat

Jika diberikan suatu vector $x = 1, 2, 3, \dots, n$, maka nilai x^2 dapat dihitung dengan fungsi berikut;

```
masukan=c(1:5) # nilai x awal (1,2,3,4,5)

x_kuadrat <- function(x)
{
  x*x
}

x_kuadrat(masukan) # menggunakan fungsi

## [1]  1  4  9 16 25

cat("Hasil kuadrat:",x_kuadrat(masukan)) # menggunakan fungsi
```

Hasil kuadrat: 1 4 9 16 25

atau,

```
masukan=c(1:5) # nilai x awal (1,2,3,4,5)

x_kuadrat <- function(x)
{
  keluaran = x^2
  return(cat("Hasil kuadrat:",keluaran))
}

x_kuadrat(masukan) # menggunakan fungsi

## Hasil kuadrat: 1 4 9 16 25
```

5.1.2 Fungsi Akar

Jika diberikan suatu vector adalah kelipatan 9 dari 3 sampai 27, maka nilai $\sqrt[3]{x}$ dapat dihitung dengan fungsi berikut;

```
masukan = seq(from=3, to=27, by = 9)          # kelipatan 9 dari 3 sampai 27

akar <- function(x)                          # nama fungsi dan argumen
{
  x^(1/3)                                    # pembukaan fungsi
}                                              # perintah yang dilakukan
                                                # penutupan fungsi

akar(masukan)                                # menggunakan fungsi

## [1] 1.442 2.289 2.759
```

5.1.3 Nilai Rerata

Jika diberikan suatu vector adalah 100 sampel acak dengan rata-rata 160 dan standar deviasi 15, maka nilai rata-ratanya \bar{x} dapat dihitung dengan fungsi berikut;

```
masukan = rnorm(100, mean=160, sd=15)        # 100 sampel, mean=160, dan sd=15

rerata<- function(x)                          # nama fungsi dan argumen
{
  sum(x)/length(x)                           # pembukaan fungsi
}                                              # perintah yang dilakukan
                                                # penutupan fungsi

rerata(masukan)                                # menggunakan fungsi

## [1] 162.3
```

5.1.4 Konversi Persen

Misalkan anda ingin menyajikan angka pecahan sebagai nilai persentase, dibulatkan dengan baik ke dalam dua digit desimal. Berikut cara mencapainya:

- Kalikan bilangan pecahan dengan 100.
- Bulatkan hasilnya ke satu tempat desimal: Anda dapat menggunakan fungsi `round()` untuk melakukan ini.
- Tempelkan tanda persentase setelah angka yang dibulatkan dengan menggunakan fungsi `paste()`

- Cetak hasilnya: dengan menggunakan `print()`.

Sebenarnya, akan sangat mudah menerjemahkan langkah-langkah ini ke dalam skrip R berikut ini:

```
x <- c(0.8765, 0.4321, 0.1234, 0.05678)
persen <- round(x*100, digits = 2)
a <- paste(persen, "%")
print(a)

## [1] "87.65 %" "43.21 %" "12.34 %" "5.68 %"
```

Untuk membuat skrip ini menjadi sebuah fungsi, Anda perlu melakukan beberapa hal berikut:

```
x <- c(0.8765, 0.4321, 0.1234, 0.05678)

persen <- function(x)
{
  persen_2digit <- round(x * 100, digits = 2)
  keluaran <- paste(persen_2digit, "%")
  return(keluaran)
}

persen(x)

## [1] "87.65 %" "43.21 %" "12.34 %" "5.68 %"
```

5.2 Simpan Fungsi

Jika anda menyimpan skrip ini sebagai file .R: misalnya, `percent.R` ke komputer/PC anda dalam sebuah folder. Kemudian, kapapun anda dapat memanggil skrip ini di konsol dengan perintah berikut:

```
x <- c(0.8765, 0.4321, 0.1234, 0.05678)
source('persen.R')                      # pastikan direktori anda benar
persen(x)                                # menggunakan fungsi anda

## [1] "87.65 %" "43.21 %" "12.34 %" "5.68 %"
```

Catatan: Sebenarnya sudah ada library yang dapat anda gunakan untuk mengubah suatu array di R yaitu:

```
library(scales)
percent(x,accuracy=0.01)

## [1] "87.65%" "43.21%" "12.34%" "5.68%"
```

5.3 Fungsi Multi Argumen

Kita dapat menulis fungsi dengan lebih dari satu argumen. Berikut ini adalah fungsi langsung mengalikan dua variabel.

```
nama.fungsi <- function(argumen1, argumen2, ..., argumen_n)
{
  perhitungan perintah yang dilakukan terhadap argumen
  beberapa kode lain
}
```

5.3.1 Luas & Keliling

Misalkan anda ingin menghitung luas dan keliling suatu persegi (panjang), diketahui panjang p dan lebarnya l . Berikut ini diperlihatkan penyelesaian dengan menggunakan fungsi:

```
luas_keliling <- function(p,l)                      # nama fungsi dan argumen
{                                                       # pembukaan fungsi
  luas= p*l                                         # menghitung luas
  keliling= 2*(p+l)                                 # menghitung keliling
  return (cat(c("Luas:",luas, "\n",
              "Keliling:",keliling)))                  # penutupan fungsi
}                                                       # menggunakan fungsi
luas_keliling(8,6)
```

Luas: 48
Keliling: 28

5.3.2 Luas, Keliling dan Volume

Misalkan anda ingin menghitung luas, keliling dan volume suatu balok (kubus), diketahui panjang p , lebarnya l , dan tinggi t . Berikut ini diperlihatkan penyelesaian dengan menggunakan fungsi:

```

lukelvol <- function(p,l,t)          # nama fungsi dan argumen
{                                     # pembukaan fungsi
  luas_permukaan = 2*((p*l)+(p*t)+(t*l)) # menghitung luas permukaan
  keliling = 2*(p + l)                  # menghitung keliling rusuk
  volume = p*l*t                      # menghitung volume balok (kubus)
  return (cat(c("Luas Permukaan:",
               luas_permukaan, sep = "\n",
               "Keliling:",keliling,sep="\n",
               "Volume:",volume)))
}                                     # penutupan fungsi
lukelvol(6,7,8)                      # menggunakan fungsi

## Luas Permukaan: 280
## Keliling: 26
## Volume: 336

```

5.3.3 Rerata Frekuensi

Jika diberikan data berfrekensi sebagai berikut:

Tinggi Badan	Frekuensi
150	16
155	23
160	28
165	40
170	39
175	22
180	9

Maka cara menghitung reratanya dengan bantuan fungsi adalah sebagai berikut:

```

Tinggi<-seq(150,180,5)           # masukan/argumen 1
Frek<-c(15,23,28,40,39,22,9)     # masukan/argumen 2

rerata_frek <- function(x,frek)   # nama fungsi dan argumen
{                                     # pembukaan fungsi
  keluaran=sum(x*frek)/length(frek) # menghitung rerata frekuensinya
  return(cat("Reratanya:",keluaran)) # print hasil dengan komentar
}                                     # penutupan fungsi

rerata_frek(Tinggi,Frek)           # menggunakan fungsi

## Reratanya: 4142

```

5.4 Fungsi untuk Data Frame

5.4.1 Normalisasi

Seperti yang telah saya sebutkan sebelumnya, ilmuwan data perlu melakukan banyak tugas berulang. Sebagian besar waktu, kami menyalin dan menempelkan potongan kode berulang-ulang. Contoh lain, normalisasi suatu variabel sangat disarankan sebelum kita menjalankan algoritma pembelajaran mesin. Rumus untuk menormalkan variabel adalah:

$$\text{Normalisasi} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

Mari kita buat kerangka data seperti yang telah kita pelajari di dasar-dasar R di bagian terakhir.

```
set.seed(123)                                     # statik random set
a = rnorm(100, 5, 1)                            # 100 bilangan acak normal
b = rnorm(100, 5, 1)                            # 100 bilangan acak normal
c = rnorm(100, 5, 1)                            # 100 bilangan acak normal
df<- data.frame(a,b,c)                         # buat data frame
df                                                 # mencetak hasil data frame

##          a      b      c
## 1  4.440 4.290 7.199
## 2  4.770 5.257 6.312
## 3  6.559 4.753 4.735
## 4  5.071 4.652 5.543
## 5  5.129 4.048 4.586
## 6  6.715 4.955 4.524
## 7  5.461 4.215 4.211
## 8  3.735 3.332 4.405
## 9  4.313 4.620 6.651
## 10 4.554 5.919 4.946
## 11 6.224 4.425 5.119
## 12 5.360 5.608 5.244
## 13 5.401 3.382 6.232
## 14 5.111 4.944 4.484
## 15 4.444 5.519 4.007
## 16 6.787 5.301 6.676
## 17 5.498 5.106 4.559
## 18 3.033 4.359 4.277
## 19 5.701 4.150 3.764
## 20 4.527 3.976 3.715
## 21 3.932 5.118 4.426
```

```
## 22 4.782 4.053 5.618
## 23 3.974 4.509 6.110
## 24 4.271 4.744 5.708
## 25 4.375 6.844 4.636
## 26 3.313 4.348 5.060
## 27 5.838 5.235 4.295
## 28 5.153 5.078 4.283
## 29 3.862 4.038 5.885
## 30 6.254 4.929 3.984
## 31 5.426 6.445 6.955
## 32 4.705 5.452 4.910
## 33 5.895 5.041 5.215
## 34 5.878 4.578 4.261
## 35 5.822 2.947 4.426
## 36 5.689 6.131 3.683
## 37 5.554 3.539 4.817
## 38 4.938 5.740 5.419
## 39 4.694 6.909 5.324
## 40 4.620 3.556 4.218
## 41 4.305 5.702 4.211
## 42 4.792 4.738 4.498
## 43 3.735 3.428 6.496
## 44 7.169 3.485 3.863
## 45 6.208 3.398 4.821
## 46 3.877 4.469 6.902
## 47 4.597 3.538 4.899
## 48 4.533 5.688 3.640
## 49 5.780 7.100 4.335
## 50 4.917 3.713 5.485
## 51 5.253 5.788 4.624
## 52 4.971 5.769 4.438
## 53 4.957 5.332 4.656
## 54 6.369 3.992 5.090
## 55 4.774 4.881 6.599
## 56 6.516 4.720 4.911
## 57 3.451 5.563 6.081
## 58 5.585 4.628 5.631
## 59 5.124 5.977 4.886
## 60 5.216 4.625 3.467
## 61 5.380 6.053 4.479
## 62 4.498 3.951 4.510
## 63 4.667 3.740 5.047
## 64 3.981 8.241 6.300
## 65 3.928 4.583 7.293
## 66 5.304 5.298 6.548
## 67 5.448 5.637 4.867
```

```

## 68 5.053 4.516 3.243
## 69 5.922 5.517 4.611
## 70 7.050 5.369 5.089
## 71 4.509 4.785 5.845
## 72 2.691 5.065 5.963
## 73 6.006 4.966 5.684
## 74 4.291 7.128 3.605
## 75 4.312 4.259 5.850
## 76 6.026 3.904 4.553
## 77 4.715 5.038 5.175
## 78 3.779 5.310 5.075
## 79 5.181 5.437 5.428
## 80 4.861 4.542 5.025
## 81 5.006 3.937 3.333
## 82 5.385 6.263 5.736
## 83 4.629 4.650 5.386
## 84 5.644 4.134 4.734
## 85 4.780 4.764 5.118
## 86 5.332 4.803 5.134
## 87 6.097 6.110 5.221
## 88 5.435 5.085 6.641
## 89 4.674 5.754 4.781
## 90 6.149 4.501 5.168
## 91 5.994 5.214 6.168
## 92 5.548 4.675 6.054
## 93 5.239 5.095 6.145
## 94 4.372 4.105 4.423
## 95 6.361 3.689 7.002
## 96 4.400 6.997 5.067
## 97 7.187 5.601 6.867
## 98 6.533 3.749 3.649
## 99 4.764 4.389 5.021
## 100 3.974 3.815 6.250

```

```
typeof(df)
```

```
## [1] "list"
```

Kita sudah mengetahui cara menggunakan fungsi `min()` dan `max()` di R. Oleh karena itu kita dapat menggunakan rumus normalisasi yang kita miliki di atas untuk mendapatkan nilai normalisasi `df` sebagai berikut:

```

df.norm <- data.frame(
  a = (df$a -min(df$a))/(max(df$a)-min(df$a)),
  b = (df$b -min(df$b))/(max(df$b)-min(df$b)),

```

```
c = (df$c -min(df$c))/(max(df$c)-min(df$c))
)
df.norm

##          a         b         c
## 1  0.38890 0.25364 0.97672
## 2  0.46236 0.43634 0.75784
## 3  0.86020 0.34123 0.36828
## 4  0.52923 0.32218 0.56789
## 5  0.54230 0.20808 0.33144
## 6  0.89497 0.37932 0.31615
## 7  0.61605 0.23957 0.23902
## 8  0.23220 0.07278 0.28692
## 9  0.36080 0.31600 0.84142
## 10 0.41443 0.56141 0.42041
## 11 0.78578 0.27915 0.46320
## 12 0.59357 0.50266 0.49393
## 13 0.60268 0.08223 0.73810
## 14 0.53816 0.37733 0.30632
## 15 0.38993 0.48593 0.18867
## 16 0.91095 0.44471 0.84755
## 17 0.62427 0.40778 0.32481
## 18 0.07618 0.26680 0.25520
## 19 0.66953 0.22733 0.12847
## 20 0.40840 0.19438 0.11651
## 21 0.27607 0.41004 0.29202
## 22 0.46507 0.20886 0.58636
## 23 0.28537 0.29517 0.70782
## 24 0.35145 0.33945 0.60848
## 25 0.37454 0.73610 0.34395
## 26 0.13844 0.26468 0.44851
## 27 0.69987 0.43228 0.25976
## 28 0.54766 0.40255 0.25664
## 29 0.26043 0.20614 0.65221
## 30 0.79239 0.37435 0.18296
## 31 0.60839 0.66067 0.91659
## 32 0.44793 0.47310 0.41145
## 33 0.71262 0.39561 0.48673
## 34 0.70884 0.30802 0.25138
## 35 0.69626 0.00000 0.29191
## 36 0.66670 0.60151 0.10853
## 37 0.63674 0.11193 0.38858
## 38 0.49978 0.52759 0.53722
## 39 0.44550 0.74842 0.51384
## 40 0.42893 0.11510 0.24076
```

```
## 41  0.35905 0.52038 0.23901
## 42  0.46731 0.33830 0.30974
## 43  0.23213 0.09087 0.80319
## 44  0.99591 0.10173 0.15291
## 45  0.78219 0.08532 0.38954
## 46  0.26377 0.28754 0.90352
## 47  0.42395 0.11172 0.40882
## 48  0.40977 0.51776 0.09796
## 49  0.68701 0.78450 0.26960
## 50  0.49501 0.14473 0.55363
## 51  0.56988 0.53661 0.34100
## 52  0.50720 0.53308 0.29500
## 53  0.50401 0.45057 0.34883
## 54  0.81792 0.19736 0.45610
## 55  0.46334 0.36526 0.82848
## 56  0.85080 0.33486 0.41188
## 57  0.16911 0.49416 0.70064
## 58  0.64356 0.31748 0.58951
## 59  0.54109 0.57236 0.40569
## 60  0.56157 0.31707 0.05522
## 61  0.59798 0.58666 0.30507
## 62  0.40183 0.18965 0.31279
## 63  0.43944 0.14980 0.44540
## 64  0.28702 1.00000 0.75482
## 65  0.27519 0.30909 1.00000
## 66  0.58105 0.44415 0.81591
## 67  0.61323 0.50806 0.40087
## 68  0.52534 0.29645 0.00000
## 69  0.71866 0.48545 0.33775
## 70  0.96948 0.45751 0.45578
## 71  0.40434 0.34714 0.64242
## 72  0.00000 0.40016 0.67144
## 73  0.73722 0.38139 0.60273
## 74  0.35583 0.78985 0.08921
## 75  0.36054 0.24780 0.64356
## 76  0.74163 0.18081 0.32348
## 77  0.45022 0.39496 0.47692
## 78  0.24207 0.44647 0.45216
## 79  0.55387 0.47027 0.53948
## 80  0.48266 0.30125 0.43985
## 81  0.51483 0.18698 0.02199
## 82  0.59923 0.62642 0.61562
## 83  0.43111 0.32178 0.52908
## 84  0.65685 0.22434 0.36815
## 85  0.46451 0.34319 0.46293
## 86  0.58733 0.35058 0.46685
```

```

## 87  0.75748 0.59747 0.48833
## 88  0.61033 0.40383 0.83894
## 89  0.44106 0.53025 0.37966
## 90  0.76904 0.29352 0.47525
## 91  0.73450 0.42833 0.72227
## 92  0.63551 0.32650 0.69407
## 93  0.56664 0.40569 0.71656
## 94  0.37390 0.21870 0.29115
## 95  0.81615 0.14024 0.92824
## 96  0.38005 0.76506 0.45022
## 97  1.00000 0.50129 0.89475
## 98  0.85439 0.15148 0.10016
## 99  0.46113 0.27238 0.43893
## 100 0.28528 0.16391 0.74240

```

Namun, metode ini rentan terhadap kesalahan penulisan koding. Kita bisa menyalin koding yang hampir serupa, tetapi mungkin saja lupa mengubah variabel yang perlu diganti. Oleh karena itu sebaiknya, kita dapat pertimbangkan penggunaan suatu fungsi untuk melakukannya:

```

normalize <- function(x){
  norm <- (x-min(x))/(max(x)-min(x))
  return(norm)
}

```

```

df.norm = normalize(df)
df.norm

```

```

##          a      b      c
## 1  0.31507 0.28805 0.81222
## 2  0.37458 0.46233 0.65251
## 3  0.69689 0.37160 0.36828
## 4  0.42875 0.35343 0.51392
## 5  0.43935 0.24459 0.34140
## 6  0.72506 0.40794 0.33024
## 7  0.49910 0.27463 0.27397
## 8  0.18812 0.11553 0.30892
## 9  0.29230 0.34754 0.71350
## 10 0.33575 0.58163 0.40632
## 11 0.63660 0.31239 0.43754
## 12 0.48088 0.52559 0.45996
## 13 0.48826 0.12455 0.63811
## 14 0.43599 0.40604 0.32307
## 15 0.31590 0.50963 0.23723
## 16 0.73801 0.47031 0.71797

```

```
## 17 0.50575 0.43509 0.33656
## 18 0.06172 0.30061 0.28577
## 19 0.54242 0.26296 0.19331
## 20 0.33087 0.23153 0.18458
## 21 0.22366 0.43725 0.31264
## 22 0.37678 0.24534 0.52740
## 23 0.23119 0.32767 0.61602
## 24 0.28472 0.36991 0.54354
## 25 0.30344 0.74827 0.35053
## 26 0.11215 0.29859 0.42682
## 27 0.56700 0.45846 0.28910
## 28 0.44368 0.43010 0.28683
## 29 0.21099 0.24275 0.57544
## 30 0.64195 0.40320 0.23307
## 31 0.49289 0.67632 0.76834
## 32 0.36289 0.49740 0.39978
## 33 0.57733 0.42348 0.45471
## 34 0.57427 0.33993 0.28299
## 35 0.56408 0.04611 0.31256
## 36 0.54013 0.61989 0.17876
## 37 0.51585 0.15288 0.38309
## 38 0.40490 0.54937 0.49154
## 39 0.36092 0.76002 0.47448
## 40 0.34750 0.15590 0.27524
## 41 0.29088 0.54249 0.27396
## 42 0.37859 0.36881 0.32557
## 43 0.18806 0.13279 0.68560
## 44 0.80684 0.14315 0.21114
## 45 0.63369 0.12750 0.38379
## 46 0.21370 0.32040 0.75881
## 47 0.34346 0.15268 0.39786
## 48 0.33197 0.54000 0.17104
## 49 0.55658 0.79443 0.29628
## 50 0.40103 0.18416 0.50352
## 51 0.46169 0.55798 0.34838
## 52 0.41091 0.55461 0.31482
## 53 0.40833 0.47590 0.35409
## 54 0.66264 0.23437 0.43236
## 55 0.37537 0.39453 0.70406
## 56 0.68928 0.36553 0.40009
## 57 0.13701 0.51749 0.61078
## 58 0.52138 0.34895 0.52970
## 59 0.43837 0.59208 0.39558
## 60 0.45496 0.34856 0.13986
## 61 0.48445 0.60572 0.32216
## 62 0.32555 0.22702 0.32779
```

```
## 63 0.35602 0.18900 0.42455
## 64 0.23253 1.00000 0.65031
## 65 0.22294 0.34094 0.82920
## 66 0.47074 0.46978 0.69488
## 67 0.49681 0.53074 0.39206
## 68 0.42560 0.32889 0.09957
## 69 0.58222 0.50918 0.34600
## 70 0.78542 0.48253 0.43212
## 71 0.32758 0.37724 0.56830
## 72 0.00000 0.42781 0.58947
## 73 0.59726 0.40991 0.53935
## 74 0.28827 0.79954 0.16466
## 75 0.29209 0.28248 0.56913
## 76 0.60083 0.21858 0.33559
## 77 0.36474 0.42286 0.44755
## 78 0.19611 0.47199 0.42948
## 79 0.44872 0.49470 0.49320
## 80 0.39103 0.33347 0.42050
## 81 0.41709 0.22447 0.11562
## 82 0.48547 0.64364 0.54875
## 83 0.34927 0.35305 0.48560
## 84 0.53215 0.26011 0.36819
## 85 0.37632 0.37348 0.43734
## 86 0.47583 0.38052 0.44020
## 87 0.61367 0.61603 0.45587
## 88 0.49446 0.43132 0.71169
## 89 0.35733 0.55191 0.37658
## 90 0.62304 0.32609 0.44633
## 91 0.59505 0.45469 0.62656
## 92 0.51486 0.35755 0.60599
## 93 0.45906 0.43309 0.62240
## 94 0.30292 0.25473 0.31201
## 95 0.66120 0.17988 0.77684
## 96 0.30790 0.77590 0.42807
## 97 0.81015 0.52428 0.75241
## 98 0.69219 0.19060 0.17265
## 99 0.37358 0.30593 0.41983
## 100 0.23112 0.20246 0.64125
```

5.5 Latihan

1. Buatlah Fungsi Summary Statistik untuk data berfrekuansi
2. Buatlah Fungsi summary Statistik untuk data frame

Catatan: Kumpulkan hasil latihan anda, tidak boleh sama dengan teman ma-hasiswa lainnya.

Chapter 6

Struktur Kontrol

Pada dasarnya, struktur kontrol memungkinkan Anda untuk memasukkan beberapa “logika” ke dalam kode R Anda, daripada hanya mengeksekusi kode R yang sama setiap saat. Struktur kontrol memungkinkan Anda untuk merespons input atau fitur data dan mengeksekusi ekspresi R yang berbeda. Struktur kontrol yang umum digunakan adalah

- `if` dan `else` menguji suatu kondisi dan menindaklanjutinya
- `for` mengeksekusi loop beberapa kali
- `while` menjalankan loop saat kondisi benar
- `break` mengeksekusi fraagmen loop yang rusak
- `repeat` menjalankan loop tak terbatas (harus keluar dari itu untuk berhenti)
- `next` melewati interasi dari sebuah loop

Sebagian besar struktur kontrol tidak digunakan dalam sesi interaktif, melainkan saat menulis fungsi atau ekspresi yang lebih panjang. Namun, konstruksi ini tidak harus digunakan dalam fungsi dan ada baiknya Anda memahaminya sebelumnya.

6.1 Pengambilan Keputusan

Gambaran pengambilan Keputusan (*Decision Tree*) dalam dunia pemrograman adalah proses penentuan keputusan dengan pernyataan bersyarat. Dalam hal ini, programer menginstruksikan komputer untuk melakukan suatu aksi tertentu (X), hanya jika suatu kondisi Y terpenuhi.

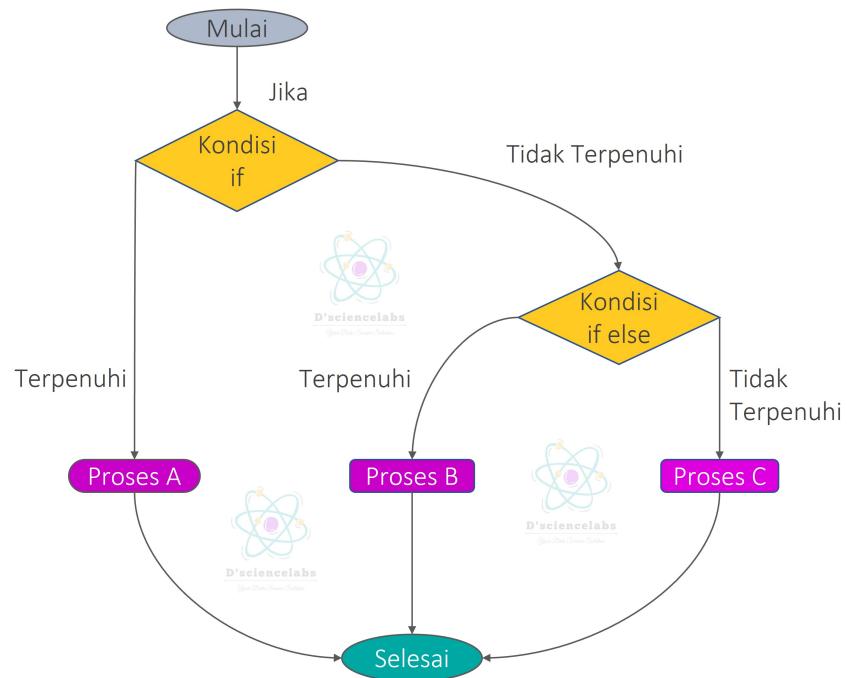


Figure 6.1: Struktur Kontrol if else


```

}

nilai(x)

## [1] "Peringkat C"

```

6.1.3 Program Hitung Faktorial

Untuk menambah pemahaman terkait pengambilan keputusan dalam pemrograman, mari kita perhatikan contoh berikut: fungsi factorial menghitung faktorial dari suatu bilangan n.

```

factorial <- function(n) {
  if (n == 0 || n == 1) {
    return(1)
  } else {
    return(n * factorial(n - 1))
  }
}

result <- factorial(5)
print(result) # Output: 120

```

```
## [1] 120
```

Faktorial dari n (dilambangkan sebagai $n!$) adalah hasil perkalian semua bilangan bulat positif dari 1 hingga n. Dalam implementasi tersebut, fungsi factorial memanggil dirinya sendiri dengan argumen yang lebih kecil hingga mencapai kasus dasar ($n = 0$ atau $n = 1$), di mana hasil faktorialnya adalah 1.

Namun, perlu diingat bahwa pengulangan rekursif dapat menyebabkan pemanggilan berulang yang dalam beberapa kasus dapat mengakibatkan penumpukan memori (stack overflow) atau kinerja yang buruk. Oleh karena itu, perlu dipastikan bahwa ada kondisi berhenti yang jelas untuk menghindari pengulangan tak terbatas. Selain itu, dalam beberapa situasi, solusi non-rekursif atau pendekatan lain mungkin lebih efisien.

6.2 Pengulangan Rekursif

Pada saat melakukan analisis data, terkadang seorang data analis atau data scientist perlu menggunakan fungsi pengulangan dalam proses pembentukan, perhitungan, manipulasi struktur data seperti halnya vektor, matriks, list, data frame, atau objek lainnya.

6.2.1 Sintaks for

Dalam poros perulangan ini digunakan fungsi kontrol ‘for’, dimana setiap iterasi pada beberapa perintah akan dievaluasi melalui perulangan yang dinginkan.

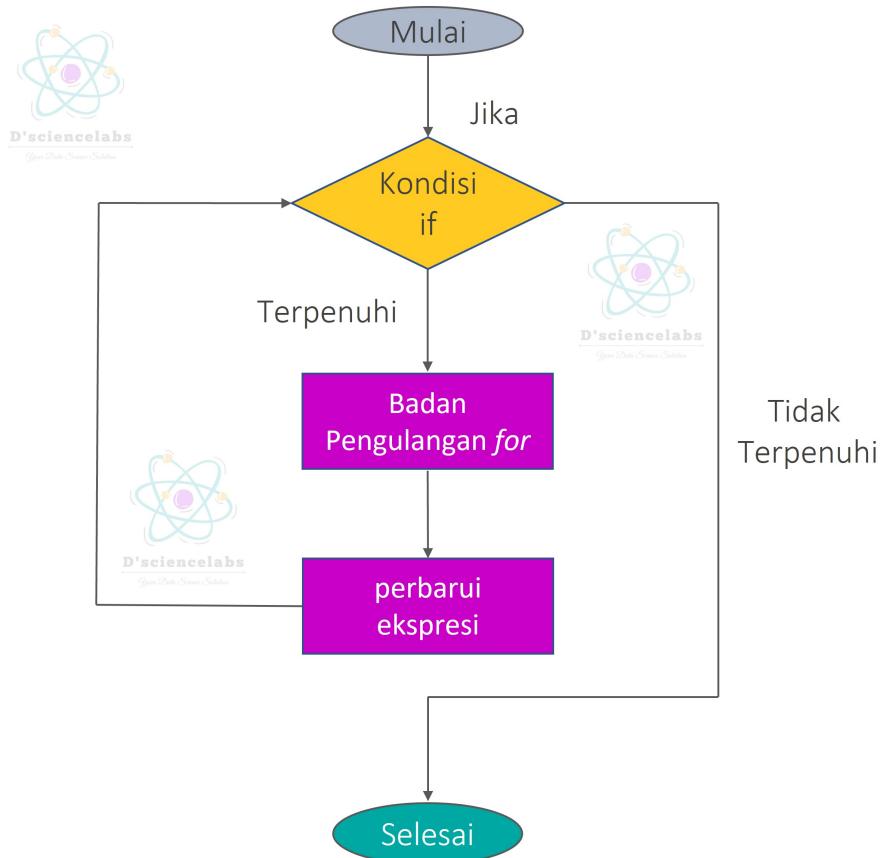


Figure 6.2: Struktur Kontrol for

Pada gambar di atas bisa dilihat bahwa perulangan juga memerlukan tes kondisi. Bila hasil tes kondisi terpenuhi, maka blok kode kembali dieksekusi. Tapi jika tidak terpenuhi, maka keluar dari perulangan. Sintaks pengulangan for di R sangat sederhana, dapat diperlihatkan sebagai berikut:

```
For (i in vector) {  
  lakukan sesuatu  
}
```

Jika anda ingin mencetak $i + 1$, menjadi $i = 1, \dots, 10$, pada setiap iterasi pengulangan (loop). Anda dapat menggunakan sintaks `for` untuk mencetak angka dengan yang dimulai $i = 0$ dan berakhir pada $i = 10$, sebagai berikut:

```
for (i in 0:10) {
    print(i + 1)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
```

Selain dapat digunakan pada data numerik, pengulangan `for` ini dapat juga diterapkan pada data karakter sebagai berikut:

```
mapel = c('Data Science', 'Statistika', 'Algoritma dan Pemrograman')

for (i in mapel){
    print(paste0("Saya suka: ", i))
}

## [1] "Saya suka: Data Science"
## [1] "Saya suka: Statistika"
## [1] "Saya suka: Algoritma dan Pemrograman"
```

Selain itu, anda juga dapat melakukan pengulangan bersarang untuk siklus tertentu. Sintaksnya direpresentasikan sebagai berikut:

```
nama = c("Bakti: ", "Alexander: ", "Siregar: ")
mapel = c('Data Science', 'Statistika', 'Algoritma dan Pemrograman')

for (x in nama)
    for (y in mapel){
        print(paste0(x, y))
    }
```

```
## [1] "Bakti: Data Science"
## [1] "Bakti: Statistika"
## [1] "Bakti: Algoritma dan Pemrograman"
## [1] "Alexander: Data Science"
## [1] "Alexander: Statistika"
## [1] "Alexander: Algoritma dan Pemrograman"
## [1] "Siregar: Data Science"
## [1] "Siregar: Statistika"
## [1] "Siregar: Algoritma dan Pemrograman"
```

6.2.2 Sintaks `while`

Terkadang anda bahkan tidak tahu berapa lama urutan input harus dijalankan. Hal ini biasa terjadi saat melakukan simulasi. Misalnya, Anda mungkin ingin mengulang sampai anda mendapatkan tiga kepala berturut-turut. Anda tidak dapat melakukan iterasi semacam itu dengan pengulangan `for`. Sebagai gantinya, Anda dapat menggunakan pengulangan `while`. Perulangan `while` lebih sederhana daripada perulangan `for` karena hanya memiliki dua komponen, kondisi, dan badan.

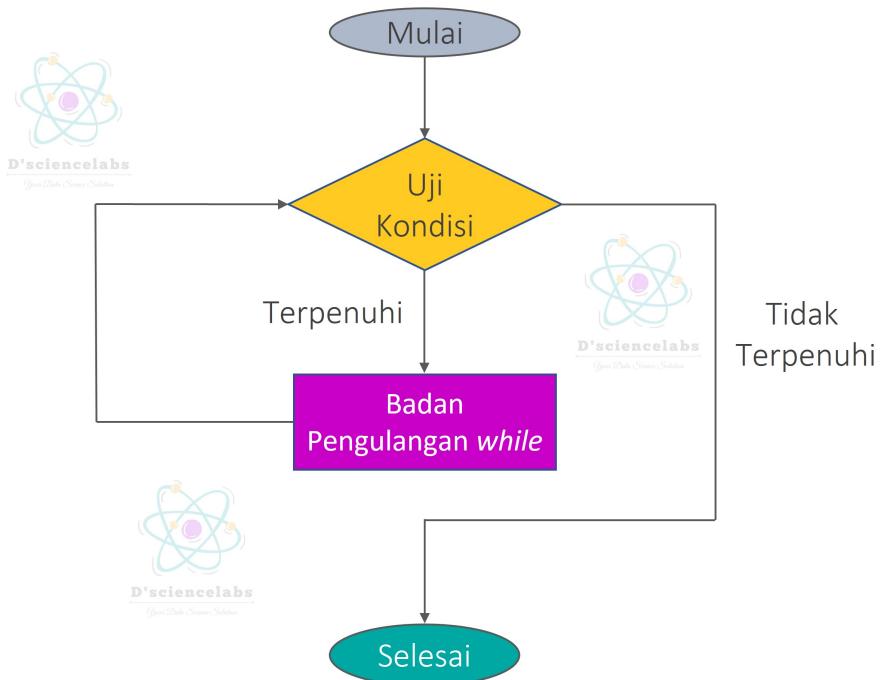


Figure 6.3: Struktur Kontrol `while`

Perulangan `while` dimulai dengan menguji suatu kondisi. Jika benar, maka mereka mengeksekusi badan perulangan. Setelah tubuh loop dieksekusi, kondisinya diuji lagi, dan seterusnya, sampai kondisinya salah, setelah itu loop akan melakukan eksekusi pada kodisi lainnya jika ada. Sintaks untuk `while` loop adalah sebagai berikut:

```
while (Kondisi) {
    lakukan sesuatu
}
```

Pengulangan while dalam bahasa pemrograman R digunakan untuk menjalankan serangkaian pernyataan selama kondisi tertentu tetap terpenuhi. Berikut adalah contoh penggunaan pengulangan while dalam R:

```
x <- 1

while (x <= 5) {
    print(x)
    x <- x + 1
}
```

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

Dalam contoh di atas, loop while akan menjalankan blok pernyataan selama nilai x masih kurang dari atau sama dengan 5. Setiap kali loop dijalankan, nilai x akan ditambah 1, sehingga pada akhirnya, kondisi $x \leq 5$ akan menjadi salah dan loop akan berhenti.

Anda perlu memastikan bahwa ada pernyataan di dalam blok while yang mengubah nilai yang digunakan dalam kondisi. Ini penting untuk menghindari terjebak dalam loop tak berakhir.

Berikut adalah contoh lain dengan menggunakan pengulangan while untuk menghitung jumlah bilangan genap antara 1 dan 10:

```
count <- 1
even_sum <- 0

while (count <= 10) {
    if (count %% 2 == 0) {
        even_sum <- even_sum + count
    }
}
```

```

    }
    count <- count + 1
}

print(even_sum) # Output: 30 (2 + 4 + 6 + 8 + 10)

## [1] 30

```

Dalam contoh ini, kita menggunakan loop while untuk menghitung jumlah bilangan genap antara 1 dan 10. Kita memeriksa apakah count (bilangan saat ini) adalah bilangan genap dengan menggunakan operator modulo (%%). Jika ya, kita menambahkan bilangan tersebut ke even_sum.

Penting untuk selalu memastikan bahwa kondisi dalam loop while pada suatu titik akan menjadi salah, sehingga loop akan berhenti dan program tidak terjebak dalam perulangan tak berakhir.

6.3 Interupsi Pengulangan

Interupsi Pengulangan dalam bahasa pemrograman dilakukan dengan pernyataan `break` pada suatu fungsi. Biasanya digunakan untuk melewati/menghentikan iterasi dan mengalirkan perintah pengulangan seperti (for, while, repeat). Meskipun, ini tidak umum digunakan dalam aplikasi statistik atau analisis data tetapi mereka memiliki peran penting dalam proses penyederhanaan program atau algoritma.

```

if (Test Kondisi) {
break
}

```

Andaikan kita ingin melakukan pengulangan suatu nilai pada vektor x, yang memiliki angka berurutan dari 1 hingga 5. Di dalam pengulangan `for`, kita telah menggunakan kondisi `if` untuk memutuskan pengulangan pada saat ditemukan nilai sama dengan 3. Dalam hal ini, loop akan berakhir ketika sesat pernyataan `break` terpenuhi.

```

x <- 1:100
for (val in x) {
if (val == 50){
break
}
print(val)
}

```

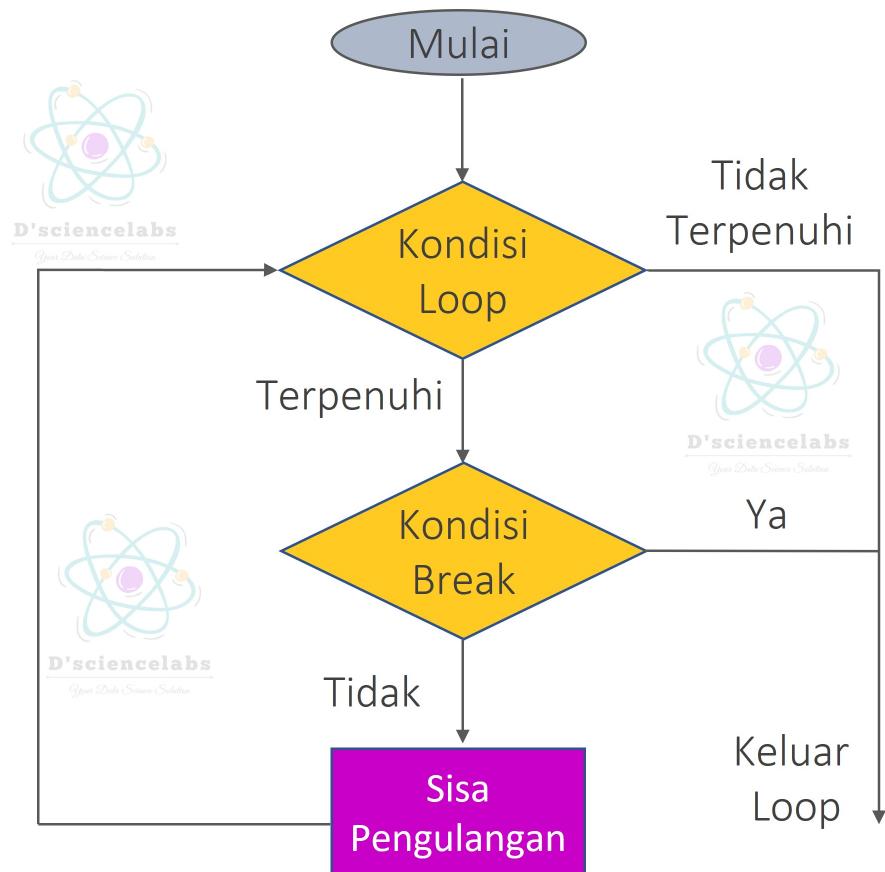


Figure 6.4: Struktur Kontrol break

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 31
## [1] 32
## [1] 33
## [1] 34
## [1] 35
## [1] 36
## [1] 37
## [1] 38
## [1] 39
## [1] 40
## [1] 41
## [1] 42
## [1] 43
## [1] 44
## [1] 45
## [1] 46
```

```
## [1] 47
## [1] 48
## [1] 49
```

6.4 Pengulangan Berkala

Kondisi perulangan yang digunakan adalah `repeat` untuk mengulangi perintah beberapa kali. Tidak ada pemeriksaan kondisi di loop untuk keluar dalam fungsi pengulangan ini. Kita sendiri harus menempatkan kondisi secara eksplisit di dalam tubuh loop.

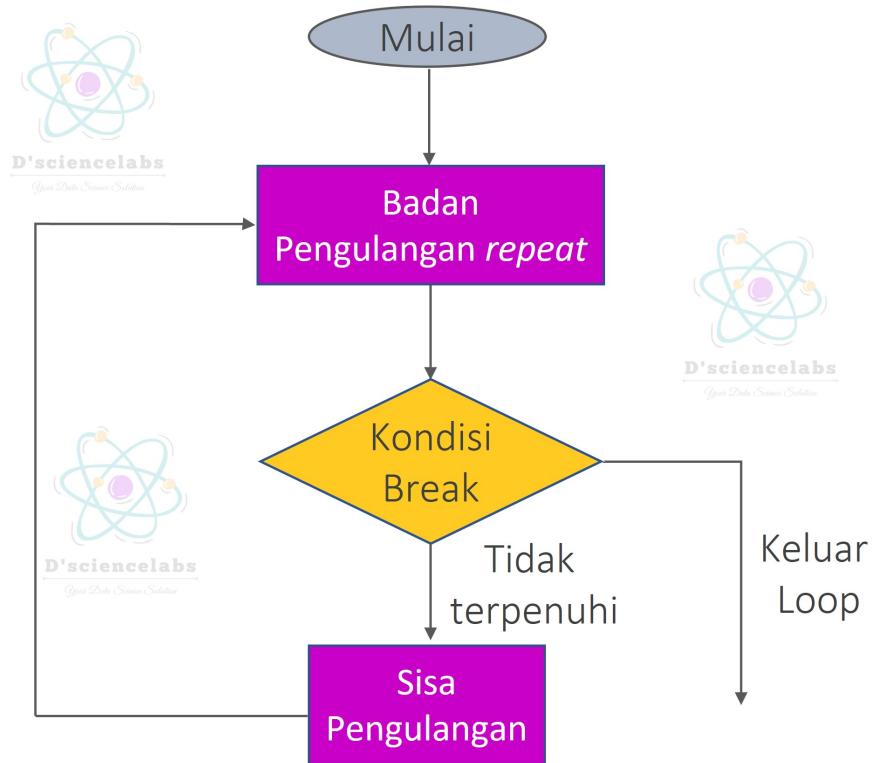


Figure 6.5: Struktur Kontrol repeat

Adapun sintak yang digunakan untuk melakukan pengulangan dengan `repeat` adalah sebagai berikut:

```
repeat {
  lakukan sesuatu
}
```

Mari kita gunakan kondisi ini untuk memeriksa dan keluar dari loop ketika x mengambil nilai 6. Oleh karena itu, kita melihat dalam output bahwa hanya ada nilai dari 1 hingga 5 yang akan dicetak.

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

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

6.5 Skip Iterasi Pengulangan

Penyataan skip iterasi digunakan untuk melewati sisa kode di dalam satu lingkaran untuk iterasi saat ini saja. Artinya, tidak seperti pernyataan break, pengulangan tidak berhenti tetapi terus berlanjut dengan iterasi berikutnya.

```
for pengulangan {
  if (kondisi){
    next
  }
  print(val)
}
```

Gunakan pernyataan `next`, misalnya untuk memeriksa apakah nilainya sama dengan 5. Jika nilainya sama dengan 5, evaluasi saat ini berhenti (nilai tidak dicetak) tetapi perulangan berlanjut dengan iterasi berikutnya.

```
x <- 1:11
for (val in x) {
  if (val == 5){
    next
  }
  print(val)
}
```

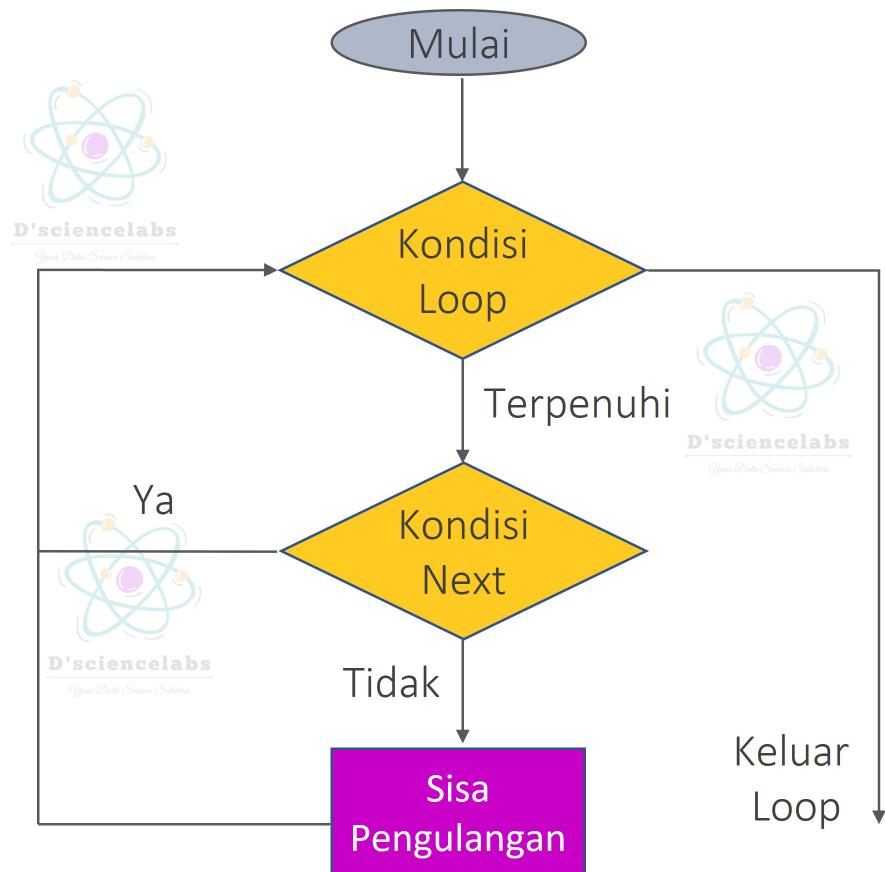


Figure 6.6: Struktur Kontrol next

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
```

6.6 Latihan

1. Pernyataan Kondisional (if-else): Buatlah sebuah program yang memeriksa apakah suatu bilangan adalah positif, negatif, atau nol, dan mencetak pesan yang sesuai.
2. Pengulangan (for): Hitunglah jumlah dari semua bilangan bulat genap antara 1 dan 50 menggunakan loop for.
3. Pengulangan (while): Buatlah program yang meminta pengguna untuk menebak suatu angka acak antara 1 dan 100. Berikan petunjuk apakah angka yang ditebak terlalu besar atau terlalu kecil, dan berhenti ketika pengguna berhasil menebak angka tersebut.
4. Pernyataan Switch: Buatlah program yang mengonversi nama hari dalam bahasa Inggris menjadi nama hari dalam bahasa Indonesia menggunakan pernyataan switch.
5. Pengulangan dan Pernyataan Kondisional Gabungan: Hitunglah jumlah bilangan bulat positif yang dapat dibagi habis oleh 3 atau 5 di antara 1 dan 1000, lalu cetak hasilnya.
6. Nested Loop: Buatlah pola segitiga angka seperti berikut menggunakan nested loop:

```
1
12
123
1234
12345
```

7. Penggunaan break: Buatlah sebuah program yang mencari bilangan prima pertama yang lebih dari 100.

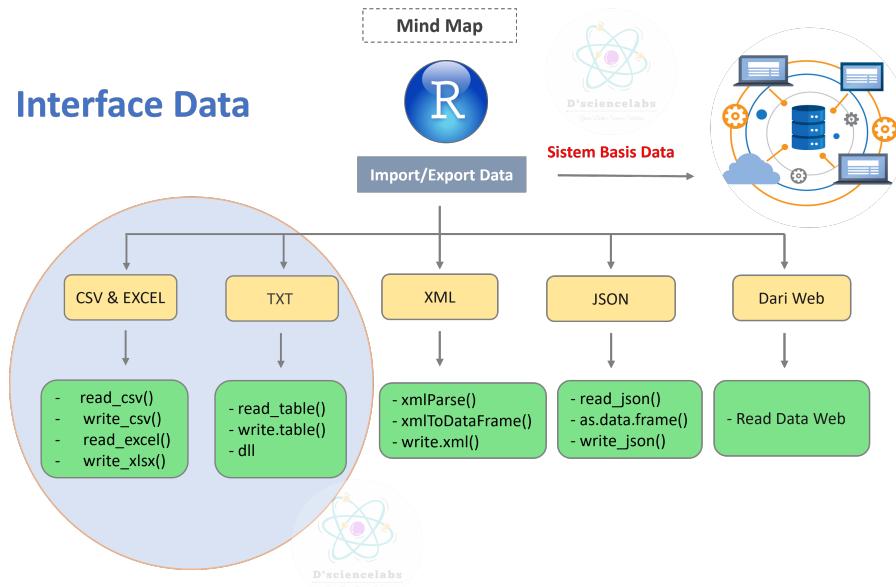
8. Fungsi Rekursif: Buatlah fungsi rekursif untuk menghitung bilangan Fibonacci ke-n.

Catatan: Kumpulkan hasil latihan anda, tidak boleh sama dengan teman mahasiswa lainnya.

Chapter 7

Interface Data

Terhubung dengan data adalah hal paling mendasar dalam proses olah data (Data Science). Pada bagian ini akan dibahas bagaimana proses antarmuka data yang paling sederhana dengan menggunakan R. Secara garis besar R dapat membaca data dari file yang disimpan di dalam maupun luar direktoriya. Selain itu, sistem operasi R juga dapat menyimpan dan membaca data dalam berbagai format file seperti csv, excel, txt, rds, xml, json, dll.



Catatan: Interface Data dapat dilakukan pada working direktori local (PC) maupun Basis website

Figure 7.1: Interface Data

Catatan: Data yang digunakan dalam pembelajaran ini, dapat download pada link berikut!

- input1.csv
- input2.csv
- input3.xls
- input4.xlsx
- input5.txt
- input6.Rdmpd
- input7.rds
- input8.ascii
- input9.xml
- input10.json

7.1 Impor/Ekspor CSV

Terdapat beberapa cara yang dapat dilakukan untuk impor/expor data dengan R. Berikut ini, diperlihatkan 3 cara yang paling sering digunakan dalam proses antarmuka data.

7.1.1 Cara 1

Pertama-tama pastikan direktori kerja anda dalam koneksi yang benar, periksalah dengan menggunakan fungsi `getwd()`. Kemudian, anda dapat mengatur direktori kerja baru menggunakan fungsi `setwd()`.

```
print(getwd())                                # cetak direktori kerja R
getwd()                                         # cetak direktori kerja R
#setwd("C:/Users/Bakti/Desktop/")            # atur direktori (ini contoh)
#setwd("C:\\Users\\Bakti\\Desktop\\")          # atau dengan cara ini
```

Selanjutnya, proses impor data CSV dengan menggunakan fungsi `read.csv()`.

```
setwd(getwd())                                # setting direktori
df1 <-read.csv("Data/Input/input1.csv",sep = ",") # format pemisah koma
df2 <-read.csv("Data/Input/input2.csv",sep = ";") # format pemisah titik koma
```

Jika anda ingin melakukan expor data (simpan) ke direktori yang diinginkan maka dapat dilakukan dengan menggunakan fungsi `write.csv()` untuk format data dengan pemisah koma (",") dan fungsi `write.csv2()` untuk format data dengan pemisah semicolon (";").

```
write.csv(df1,"Data/Output/output1.csv", row.names = FALSE)
write.csv2(df2,"Data/Output/output2.csv", row.names = FALSE)
```

7.1.2 Cara 2

Bagi anda yang lebih menyukai koneksi direktori secara automatis, cara ini adalah pilihan yang yang lebih bijak dan praktis.

```
# setting direktori
(WD <- getwd())
if (!is.null(WD)) setwd(WD)

# impor data
df1 <- read.csv(file.path(WD, 'Data/Input', 'input1.csv'))
df2 <- read.csv(file.path(WD, 'Data/Input', 'input2.csv'))

# ekspor data
write.csv(df1, file.path(WD, 'Data/Output', 'output1.csv'),row.names = FALSE)
write.csv(df2, file.path(WD, 'Data/Output', 'output2.csv'),row.names = FALSE)
```

7.1.3 Cara 3

Cara ini adalah langkah alternatif bagi anda yang lebih nyaman untuk memilih data yang tersimpan di komputer (PC) tanpa harus melakukan pengaturan direktori terlebih dahulu sebelum melakukan proses impor data.

```
df3 <- read.csv(file.choose())
```

Catatan: Proses impor/espor data selanjutnya akan dilakukan dengan cara 1 satu saja.

7.2 Impor/Ekspor Excel

Microsoft Excel adalah program spreadsheet yang paling banyak digunakan yang menyimpan data dalam format .xls atau .xlsx. R dapat membaca langsung dari file-file ini menggunakan paket `readxl`.

```
# install.packages(c("readxl", "writexl"))
pacman::p_load(readxl, writexl)

# impor data
```

```
df3<-read_excel("Data/Input/input3.xls")           # impor data xls (97-2003)
df4<-read_excel("Data/Input/input4.xlsx",sheet=1) # impor dataxlsx (2003-up)

# ekspor data
write_xlsx(df3,"Data/Output/output3.xls")
write_xlsx(df4,"Data/Output/output4.xlsx")
```

7.3 Impor/Ekspor TXT and RDS

Format data paling umum yang sering digunakan untuk impor/ekspor data adalah file CSV dan XLSX. Namun, ukuran file CSV atau XLSX cenderung lebih besar dibanding format yang lainnya seperti TXT atau file biner R (RDS). Sehingga, ini akan membutuhkan tambahan waktu pada saat anda melakukan impor/ekspor data. Tips, yang sangat saya rekomendasikan dalam hal ini adalah mengimpor dan ekspor data tersebut dengan format TXT atau binary.

```
# impor data
df5 <- read.table("Data/Input/input5.txt")          # format TXT (notepad)
df6 <- source("Data/Input/input6.Rdmpd")            # format TXT (Rdmpd)
df7 <- readRDS("Data/Input/input7.rds")             # format binary RDS
df8 <- readRDS("Data/Input/input8.ascii")            # format binary ASCII

# ekspor data
write.table(df5,"Data/Output/output5.txt")
dump("df6", "Data/Output/output6.Rdmpd")
saveRDS(df7, "Data/Output/output7.rds")
saveRDS(df8, "Data/Output/output8.ascii", ascii=TRUE)
```

Catatan: Format binary sangat lazim digunakan untuk meringkas file yang berukuran besar, terutama saat menggunakan sistem kontrol (API) seperti halnya Git.

7.4 Impor/Ekspor XML

XML adalah kumpulan berbagi format file dan data di World Wide Web, intranet, dan di tempat lain menggunakan teks ASCII standar. XML adalah singkatan dari eXtensible Markup Language. Untuk penjelasan lebih lengkap silahkan kunjungi

Sebenarnya XML sangat mirip dengan HTML, dimana sintak-sintaknya berisi markup. Tetapi, perlu dicatat bahwa XML tidak sama dalam hal tag markup

HTML yang menjelaskan laman struktur datanya terdapat dalam file tersebut. Silahkan perhatikan video penjelasan mengenai HTML;

Selanjutnya, mari kita lakukan impor/ekspor data dengan format XML.

```
library("XML")                                # load paket XML
library("kulife")                            # load paket kulife
library("methods")                           # paket kondisional

df9 <- xmlParse("Data/Input/input9.xml")      # impor data XML
xml_df <- xmlToDataFrame(df9)                 # konversi ke Dataframe
write.xml(xml_df, "Data/Output/output9.xml")  # ekspor data XML
```

7.5 Impor/Eksport JSON

File JSON menyimpan data sebagai teks dalam format yang dapat dibaca manusia. Json adalah singkatan dari JavaScript Object Notation. R dapat membaca file JSON menggunakan paket `jsonlite`. Selengkapnya tentang JSON

```
library("jsonlite")                          # load paket JSON

df10 <- fromJSON("Data/Input/input10.json")   # impor data
json_df <- as.data.frame(df10)                # konversi ke Dataframe
write_json(json_df, "Data/Output/output10.json") # ekspor data JSON
```

7.6 Impor Data dari Web

Ada banyak website menyediakan data untuk dikonsumsi oleh penggunanya. Dengan menggunakan program R, kita dapat mengekstrak data spesifik dari situs web tersebut secara terprogram. Di bagian ini, saya memberikan contoh cara mengimpor data dari repositori GitHub, tetapi Anda dapat melakukan hal yang sama ke situs web atau repositori lain.

7.6.1 CSV:

```
web_csv <- read.csv("https://raw.githubusercontent.com/dsciencelabs/Algoritma_Pemrograman_R/main/
```

7.6.2 XLSX:

```
library(rio)                      # impor data dari github
install_formats()                 # periksa auto paket yg disarankan
web_xlsx <- rio::import("https://github.com/dsciencelabs/Algoritma_Pemrograman_R/raw/mal...
```

7.7 Basis Data R

Sistem basis data adalah data relasional yang disimpan dalam format yang dinormalisasi. Jadi, untuk melakukan komputasi statistik kita akan membutuhkan query SQL yang sangat canggih dan kompleks. Tetapi R dapat terhubung dengan mudah ke banyak database relasional seperti MySql, Oracle, SQL Server, dll. Selanjutnya biasanya, kita akan mengubah basis data tersebut menjadi bingkai data (data frame). Setelah data tersedia di lingkungan R, kemudian dimanipulasi atau dilakukan analisis lebih lanjut.

Catatan: Kita akan mempelajari bagian ini di bagian khusus yang disebut “Sistem Basis Data dengan R” semester berikutnya.

7.8 Menambang Data Web

Pengikisan data dari web (Webscraping Data) adalah proses menggunakan bot untuk mengekstrak konten dan data dari situs web. Tidak seperti screen scraping, yang hanya menyalin piksel yang ditampilkan di layar, web scraping mengekstrak kode HTML yang mendasarinya yang pada dasarnya disimpan dalam database. Pengikisan data ini dapat mereplikasi seluruh konten dari berbagai situs web yang target.

Catatan: Kita akan mempelajari bagian ini di bagian khusus yang disebut “Ilmu Data Terapan”.

7.9 Latihan

Buatlah tutorial Baca dan Simpan data dengan mengikuti materi yang sudah dipelajari diatas. Lengkapi dengan Gambar dan penjelasan setiap prosesnya.

Catatan: Kumpulkan hasil latihan anda, tidak boleh sama dengan teman mahasiswa lainnya.

Chapter 8

Manipulasi Data

Salah satu keterampilan paling mendasar yang harus dimiliki seorang Data Scientist adalah memanipulasi data. Untuk menjadi seseorang yang sangat efektif, Anda harus ahli dalam memanipulasi data penting. Hal ini perlu diperhatikan karena sebagian besar pekerjaan Anda akan melibatkan pengambilan dan pembersihan data.

Data Scientist menghabiskan 60% waktunya untuk membersihkan dan mengatur data. Di urutan kedua, 19% waktunya untuk mengumpulkan dataset, yang berarti Data Scientist menghabiskan sekitar 80% waktunya untuk menyiapkan dan mengelola data untuk analisis.

Pada bagian ini, Anda akan mempelajari bagaimana memanipulasi data dengan mudah menggunakan R. Kita akan membahas fungsi-fungsi manipulasi data mendasar yang sebagian besar akan Anda gunakan untuk memanipulasi data Anda.

- `read_csv()` Mengimpor data (Anda bisa menggunakan fungsi lainnya)
- `str()` Struktur Data
- `apply()` Untuk mengecek dan mengganti data yang hilang.
- `select()` Memilih kolom yang akan disertakan.
- `filter()` Memilih subset yang ada di dalam data.
- `arrange()` Mengurutkan data, berdasarkan ukuran dari variabel kontinu, berdasarkan tanggal, atau menurut abjad.
- `rename()` Mengganti nama kolom.
- `mutate()` Membuat kolom baru di dalam data, atau mengganti kolom yang sudah ada.
- `bind_rows()` Menggabungkan dua data frame menjadi satu, menggabungkan data dari kolom-kolom dengan nama yang sama.
- `group_by()` Mengelompokkan data berdasarkan variabel kategorikal.
- `summarize()` Meringkas, atau mengagregat (untuk setiap kelompok jika mengikuti `group_by()`). Sering digunakan bersama fungsi sebagai berikut:

- `mean()` Menghitung rata-rata.
- `median()` Menghitung median.
- `max()` Mencari nilai maksimum.
- `min()` Mencari nilai minimum.
- `sum()` Menambahkan semua nilai secara bersamaan.
- `n()` Menghitung jumlah record.

Saya menyarankan Anda untuk menginstal package `tidyverse`. Karena inti dari `tidyverse` mencakup packages yang cendrung Anda gunakan dalam analisis data sehari-hari.

```
install.packages("tidyverse")
```

Kita sebagian besar akan bekerja dengan dua package yang sangat berguna yang dikembangkan oleh Hadley Wickham, kepala scientist di RStudio:

- `readr` Untuk membaca dan menulis CSV dan file teks lainnya.
- `dplyr` Untuk memproses dan memanipulasi data.

8.1 Impor Data

Data yang akan kita gunakan pada bagian ini adalah `pfizer.csv` dan `fda.csv`, silakan unduh dan tempatkan di desktop Anda. Sebagai opsional, Anda dapat memuat data ke sesi R saat ini dengan memilih Import Dataset>From Teks File... di tab Environment. Tapi, dalam hal ini kita akan menggunakan fungsi `read_csv()` dari package `readr`. Salinlah kode berikut ke skrip Anda dan jalankan:

```
suppressPackageStartupMessages(library(tidyverse))# memuat tidyverse
#setwd("C:/Users/Bakti/Desktop/")# ingatlah untuk mengatur working directory
pfizer <- read_csv("Data/pfizer.csv")# memuat data `pfizer`
```

```
fda <- read_csv("Data/fda.csv")
```

8.2 Struktur Data

Perhatikan bahwa Anda akan membutuhkan pemahaman yang kuat mengenai tipe-tipe data mendasar dan struktur data dan bagaimana cara mengoprasikannya. Fungsi `str()` akan memberi tahu lebih banyak mengenai kolom dalam data Anda, termasuk tipe datanya. Salinlah kode berikut ke skrip Anda dan jalankan:

```
str(pfizer)                                # melihat struktur dari data `pfizer`  
str(fda)                                    # melihat struktur dari data `fda`
```

Sangat penting untuk dipahami karena ini adalah objek yang akan Anda manipulasi di R setiap saat. Jika Anda perlu mengubah tipe data untuk kolom apa pun, gunakan fungsi-fungsi di bawah ini:

- `as.character()` mengubah ke teks string.
- `as.numeric()` mengubah ke angka.
- `as.factor()` mengubah ke variabel kategorikal.
- `as.integer()` mengubah ke bilangan bulat.
- `as.Date()` mengubah ke tanggal.
- `as.POSIXct()` mengubah ke tanggal dan waktu penuh.

Misalnya, tambahkan kode berikut ke skrip Anda untuk mengubah total konversi dalam data `pfizer` ke variabel numerik (yang memungkinkannya menyimpan nilai desimal, jika ada).

```
pfizer$total <- as.numeric(pfizer$total)          # konversi total ke variabel numerik  
str(pfizer$total)                                # mari periksa strukturnya lagi
```

8.3 Missing Value

Tidak seperti pemrograman biasa, ketika bekerja dengan data sesungguhnya, Anda mungkin menemukan **nilai yang hilang**: pengukuran yang tidak terekam/tersimpan/dll. R memiliki mekanisme yang cukup canggih untuk menangani nilai-nilai yang hilang. Ini membedakan antara jenis berikut:

- `NA` : Not Available (Nilai NA juga memiliki kelas, ada bilangan bulat NA, karakter NA, dll).
- `NaN` : Not a Number (Nilai NaN juga merupakan NA tetapi NA bukan merupakan NaN)

Temukan nilai yang hilang di kolom dataframe `pfizer`

```
is.na(pfizer)                                # cara untuk mengecek NA  
sum(is.na(pfizer))                          # menghitung jumlah NA  
apply(is.na(pfizer),2, which)                # indeks NA (hanya df)  
which(complete.cases(pfizer))                # mengidentifikasi nilai lengkap yang diamati
```

Mekanisme yang lebih umum adalah menghapusnya secara manual:

```

clean.vector<- na.omit(pfizer$first_name)           # bersihkan/hapus NA di vektor
clean.df <- na.omit(pfizer)                         # berishkan/hapus NA di dataframe
apply(is.na(clean.df),2, which)                     # pastikan jika ada nilai yang hilang

```

8.4 Mengganti Nilai yang Hilang

Kita juga dapat mengganti nilai yang hilang dengan rata-rata (median). Praktik yang baik adalah dengan membuat dua variabel terpisah untuk mean. Setelah dibuat, kita dapat mengganti nilai yang hilang dengan variabel yang baru dibentuk. Mari unggah dan memeriksa data yang hilang.

```

PATH <- "https://raw.githubusercontent.com/dsciencelabs/Algoritma_Pemrograman_R/main/Da
tanic <- read.csv(PATH, sep = ",")
list_na <- colnames(titanic)[ apply(titanic, 2, anyNA) ]
list_na

```

Dalam hal ini, kita tidak menghapus semua nilai yang hilang, tetapi kita menggunakan metode `apply()` untuk menghitung rata-rata kolom dengan `NA`. Pertama, kita perlu menghitung rata-rata dengan argumen `na.rm = TRUE`. Argumen ini wajib karena kolom memiliki data yang hilang dan ini memberi tahu R untuk mengabaikannya.

```

average_missing <- apply(titanic[,colnames(titanic) %in% list_na],
                           2,
                           mean,
                           na.rm = TRUE)
average_missing

```

Penjelasan Kode: Terdapat 4 argumen di metode `apply` yang kita jalankan.

- `df titanic[,colnames(titanic) %in% list_na]`. Kode ini akan mengembalikan nama kolom dari objek `list_na` (mis. “age” dan “fare”).
- `2` Menghitung fungsi pada kolom.
- `mean` Menghitung rata-rata.
- `na.rm = TRUE` Menolak nilai yang hilang.

Selanjutnya, kita dapat mengganti nilai-nilai `NA`. Fungsi `mutate` dari library `dplyr` berguna untuk membuat variabel baru. Kita tidak perlu mengubah kolom asli, jadi kita dapat membuat sebuah variabel baru tanpa `NA`. Fungsi `mutate` mudah digunakan, kita hanya perlu memilih nama variabel dan menentukan bagaimana cara membuat variabel tersebut. Berikut ini kode lengkapnya:

```
library(dplyr)
titanic_replace <- titanic %>%
  mutate(age = ifelse(is.na(Age), average_missing[1], Age),
        fare = ifelse(is.na(Fare), average_missing[2], Fare))
sum(is.na(titanic_replace$Age))
sum(is.na(titanic_replace$Fare))
```

Kolom usia asli memiliki 86 nilai yang hilang sementara variabel yang baru dibuat telah mengganti nilai yang hilang dengan rata-rata usia variabel. Anda dapat mencoba sendiri mengganti penelitian yang hilang dengan nilai median juga.

8.5 Memilih Data

Pada bagian ini, Anda akan mempelajari bagaimana cara memilih atau mengelompokkan kolom dataframe berdasarkan nama dan posisi menggunakan fungsi R yaitu `select()` dalam package `dplyr`. Anda akan belajar cara menggunakan fungsi berikut ini:

- `pull()` mengekstrak nilai kolom sebagai sebuah vektor. Kolom bunga dapat ditentukan berdasarkan nama atau indeks.
- `select()` mengekstrak satu atau beberapa kolom sebagai tabel data. Fungsi ini juga dapat menghapus kolom dari data frame.
- `select_if()` Memilih kolom berdasarkan kondisi tertentu. Seseorang dapat menggunakan fungsi ini, misalnya untuk memilih kolom jika numerik.
- **Fungsi Pembantu:** `starts_with()`, `ends_with()`, `contains()`, `matches()`: Pilihlah kolom/variabel berdasarkan namanya.

```
library(tidyverse)
pfizer %>% pull(state) %>% head()
pfizer %>% select(1:3)
pfizer %>% select(1,3)
pfizer %>% select(state:total)
pfizer %>% select(state,total)
pfizer %>% select_if(is.numeric)
pfizer %>% select_if(is.character)
pfizer %>% select(starts_with("first"))
pfizer %>% select(ends_with("name"))
pfizer %>% select(contains("rst"))
pfizer %>% select(matches("_"))
pfizer %>% select(-(state:total))
pfizer %>% select(-state, -total)
```

muat `tidyverse`, yang termasuk dalam `dplyr`
ekstrak nilai kolom `state` sebagai vektor
memilih kolom 1 sampai 3
memilih kolom 1 dan 3, tidak termasuk 2
memilih semua kolom dari `state` sampai `total`
memilih kolom berdasarkan nama variabel
hanya memilih kolom numerik
hanya memilih kolom karakter
hanya memilih kolom yang dimulai dengan `first`
hanya memilih kolom yang diakhiri dengan `name`
memilih kolom yang namanya mengandung `rst`
memilih kolom yang namanya cocok dengan regulir
menghapus semua kolom dari `state` sampai `total`
menghapus kolom `state` dan `total`

8.6 Menyaring dan Mengurutkan Data

Sekarang kita akan `filter()` dan `arrange()` data dengan cara tertentu. Untuk setiap contoh berikut, salin kode berikut ke dalam skrip Anda, dan lihat hasilnya. Perhatikan bagaimana kita membuat objek baru untuk menampung data yang diproses.

Contoh 1: Temukan dokter di California yang dibayar \$10.000 atau lebih oleh Pfizer untuk menjalankan “Professional Advising”!

```
ca_expert_10000 <- pfizer %>%
  filter(state == "CA" &
         total >= 10000 &
         category == "Professional Advising")
ca_expert_10000
```

memuat semua `pfizer`
memuat semua `pfizer` disaring ber
dan juga disaring berdasarkan `tot
kemudian disaring berdasarkan `cat
cetak hasilnya

Contoh 2: Sekarang tambahkan daftar urutan secara menurun pembayaran yang diterima oleh dokter di bagian akhir kode!

```
ca_expert_10000 <- pfizer %>%
  filter(state == "CA" &
         total >= 10000 &
         category == "Professional Advising") %>%
  arrange(desc(total))
ca_expert_10000
```

memuat semua `pfizer`
disaring berdasarkan `state` di Cal
dan juga disaring berdasarkan `tot
kemudian disaring berdasarkan `cat
mengurutkan secara menurun pembaya
cetak hasilnya

arrange((total))

mengurutkan secara menaik telah di

Contoh 3: Temukan dokter di California atau New York yang dibayar \$10.000 atau lebih oleh Pfizer untuk menjalankan “Professional Advising”!

Perhatikan bahwa, dalam kasus ini kita menggunakan `|` operator Boolean, dan tanda kurung di sekitar bagian kueri tersebut. Ini memastikan bahwa bagian kueri ini dijalankan terlebih dahulu. Lihat apa yang terjadi jika Anda mengecualikan mereka.

```
ca_ny_expert_10000 <- pfizer %>%
  filter((state == "CA" | state == "NY") &
         total >= 10000 &
         category == "Professional Advising") %>%
  arrange(desc(total))
ca_ny_expert_10000
```

memuat semua `pfizer`
disaring berdasarkan `state` Calif
dan juga disaring berdasarkan `tot
kemudian disaring berdasarkan `cat
mengurutkan secara menurun pembaya
cetak hasilnya

Contoh 4: Temukan dokter di negara bagian selain California yang dibayar \$10.000 atau lebih oleh Pfizer untuk menjalankan “Professional Advising”!

```
not_ca_expert_10000 <- pfizer %>%
  filter(state != "CA" &
         total >= 10000 &
         category == "Professional Advising") %>%
  arrange(desc(total))
not_ca_expert_10000
```

memuat semua `pfizer`
disaring berdasarkan `state` selain California
dan juga disaring berdasarkan `total` lebih besar atau sama dengan 10000
kemudian disaring berdasarkan `category`
mengurutkan secara menurun pembayaran yang diberikan
cetak hasilnya

Contoh 5: Temukan 20 dokter di empat negara bagian terbesar (CA, TX, FL, NY) yang dibayar paling tinggi untuk “Expert-Led Forums”!

```
ca_ny_tx_fl_prof_top20 <- pfizer %>%
  filter((state == "CA" |
          state == "NY" |
          state == "TX" |
          state == "FL") &
         category == "Expert-Led Forums") %>%
  arrange(desc(total)) %>%
  head(20)
ca_ny_tx_fl_prof_top20
```

Contoh 6: Saring data `pfizer` untuk semua pembayaran untuk menjalankan “Expert-Led Forums” atau untuk “Professional Advising”, dan urutkan nama dokter berdasarkan abjad (nama belakang, kemudian nama depan)

```
expert_professional_advice <- pfizer %>%
  filter(category == "Expert-Led Forums" |
         category == "Professional Advising") %>%
  arrange(last_name, first_name)
expert_professional_advice
```

8.7 Mengganti Nama dan Mengubah (Mutate)

Di bagian ini, Anda akan belajar bagaimana mengganti nama kolom dari sebuah data frame di R. Kemudian, Anda akan belajar bagaimana cara menghitung dan menambah variabel baru ke data frame di R. Anda akan belajar fungsi-fungsi R berikut ini dari package R yaitu `dplyr`:

- `rename()` Kode ini digunakan untuk mengganti nama kolom dari sebuah data frame di R.
- `mutate()` Menghitung dan menambah variabel baru ke dalam sebuah tabel data. Hal ini tidak menghilangkan variabel yang ada.
- `transmute()` Menghitung kolom baru tetapi menghilangkan variabel yang ada.

```

# Mengganti nama kolom dari data `pfizer` dengan fungsi-fungsi dasar R:
names(pfizer)[names(pfizer) == "org_indiv"] <- "rename1"
names(pfizer)[1] <- "rename2"
names(pfizer)[names(pfizer)==names(pfizer)] <- c("rename3",
                                                 "rename4",
                                                 "first_name",
                                                 "last_name",
                                                 "city",
                                                 "state",
                                                 "category",
                                                 "cash",
                                                 "other",
                                                 "total")
# Mengganti nama kolom dari data `pfizer` dengan `dplyr::rename()``:
pfizer %>%
  rename(
    org_indiv = rename3 ,
    first_plus = rename4
  )
# Menambahkan kolom baru (year*) dengan mempertahankan data `fd` yang ada:
letters_year <- fd %>%
  mutate(year = format(issued, "%Y")) %>%
  group_by(year)
letters_year
# Menambahkan kolom baru (year*) dan (last_name*) dengan menghilangkan data `fd` yang
fd %>%
  transmute(
    year = format(issued, "%Y"),
    last_name = name_last
  )

```

8.8 Menggabungkan Data

Ada juga beberapa fungsi gabungan di `dplyr` untuk menggabungkan data dari dua data frame. Berikut ini adalah fungsi yang paling berguna:

- `inner_join()` Mengembalikan nilai dari kedua tabel hanya jika ada kecocokan.
- `left_join()` Mengembalikan semua nilai dari tabel yang pertama disebutkan, ditambah nilai dari tabel kedua yang cocok.
- `semi_join()` Menyaring tabel yang pertama disebutkan untuk mendapatkan nilai yang memiliki kecocokan dengan tabel kedua.
- `anti_join()` Menyaring tabel yang pertama disebutkan untuk mendapatkan nilai yang tidak memiliki kecocokan dengan tabel kedua.

Sebagai ilustrasinya, gabungan ini akan menemukan dokter yang dibayar oleh **Pfizer** untuk menjalankan Expertp-Led Forums yang juga menerima surat peringatan dari **fda**:

```
expert_warned_inner <- inner_join(pfizer, fda,
                                     by=c("first_name" = "name_first",
                                         "last_name" = "name_last")) %>%
                                     filter(category=="Expert-Led Forums")

expert_warned_semi <- semi_join(pfizer, fda,
                                   by=c("first_name" = "name_first",
                                        "last_name" = "name_last")) %>%
                                   filter(category=="Expert-Led Forums")
```

Kode dalam `by=c()` menentukan bagaimana gabungan harus dibuat. Jika instruksi tentang bagaimana cara menggabungkan tabel tidak disediakan, `dplyr` akan mencari kolom dengan nama yang cocok, dan melakukan penggabungan berdasarkan hal tersebut. Perbedaan antara dua gabungan di atas adalah yang pertama berisi semua kolom dari kedua data frame, sedangkan yang kedua hanya memberikan kolom dari data frame **pfizer**.

Dalam praktiknya, mungkin Anda ingin `inner_join` dan kemudian menggunakan fungsi `select` dari `dplyr` untuk memilih kolom yang ingin Anda pertahankan, misalnya:

```
expert_warned <- inner_join(pfizer, fda,
                             by=c("first_name" = "name_first",
                                 "last_name" = "name_last")) %>%
                             filter(category=="Expert-Led Forums") %>%
                             select(last_name,
                                    city,
                                    state,
                                    total,
                                    issued)

expert_warned <- inner_join(pfizer, fda,
                            by=c("first_name" = "name_first",
                                "last_name" = "name_last")) %>%
                            filter(category=="Expert-Led Forums") %>%
                            select(2:5,10,12)
```

Klik di sini untuk referensi yang lebih berguna untuk mengelola gabungan dengan `dplyr`.

8.9 Kelompok dan Ringkasan

Bagian ini memperkenalkan cara menghitung ringkasan statistik dengan mudah di R menggunakan package `dplyr`. Anda akan belajar, bagaimana:

- Menghitung ringkasan statistik untuk data yang tidak dikelompokkan, serta untuk data yang dikelompokan menurut satu atau beberapa variabel. Fungsi R: `summarise()` dan `group_by()`.
- Meringkas beberapa kolom variabel. Fungsi R:
 - `summarise_all()` Menerapkan fungsi ringkasan ke setiap kolom dalam data frame.
 - `summarise_at()` Menerapkan fungsi ringkasan ke kolom tertentu yang dipilih dengan vektor karakter.
 - `summarise_if()` Menerapkan fungsi ringkasan ke kolom yang dipilih dengan fungsi yang menampilkan TRUE

Contoh 7: Hitunglah total pembayaran data `pfizer`, dengan urutan negara bagian secara menurun!

```
state_sum <- pfizer %>%
  group_by(state) %>%
  summarise(sum = sum(total)) %>%
  arrange(desc(sum))

state_sum
```

Contoh 8: Hitunglah beberapa ringkasan statistik tambahan dari data `pfizer`, dengan urutan negara bagian secara menurun!

```
state_summary <- pfizer %>%
  group_by(state) %>%
  summarise(sum = sum(total),
            average = mean(total),
            median = median(total),
            min = min(total),
            max = max(total),
            count = n()) %>%
  arrange(desc(sum))

state_summary
```

Contoh 9: Kelompokkan dan rangkum data `pfizer` untuk beberapa kategori naik!

```
state_summary <- pfizer %>%
  group_by(state, category) %>%
  summarise(sum = sum(total),
            average = mean(total),
            median = median(total),
            min = min(total),
            max = max(total),
            count = n()) %>%
  arrange(state, category)

state_summary
```

Contoh 10: Saring data fda untuk surat-surat yang telah dikirim dari awal tahun 2006 dan seterusnya dan rangkum!

```
year_summary <- fda %>%
  filter(issued >= "2005-01-01") %>%
  arrange(issued) %>%
  mutate(year = format(issued, "%Y")) %>%
  group_by(year) %>%
  summarise(letters=n())

year_summary
```

Waspada: Menggunakan kembali variabel dapat menyebabkan hasil yang tidak dapat diharapkan, tetapi jangan khawatir. R akan memberikan peringatan seperti `summarise()` ungrouping output (`override with .groups argument`).

8.10 Membagi Data

Seperti yang sudah saya sampaikan pada bab Pemrograman R tentang menulis fungsi untuk membagi data latihan dan data percobaan. Di sini, kita akan mempelajari lebih lanjut tentang bagaimana menggunakan beberapa packages untuk membagi data. Karena bagian ini sangat penting sebagai seorang data scientist khususnya ketika Anda menerapkan tugas Machine Learning untuk menganalisis data. Pada bagian ini biasanya kita perlu membagi dataset antara set latihan dan set percobaan. Set latihan memungkinkan algoritme untuk belajar dari data. Untuk menguji kinerja model kita, kita dapat menggunakan set percobaan untuk mengembalikan ukuran kinerja. Jadi, mari kita lihat beberapa packages yang dapat Anda gunakan untuk membagi data:

8.10.1 dplyr

Anda dapat menggunakan `dplyr` untuk ini, membuat jadi sangat sederhana. itu memang memerlukan variabel id dalam dataset Anda, yang mana ini merupakan ide yang bagus, tidak hanya untuk membuat set-set tetapi juga untuk keterlacakkan selama proyek Anda. Tambahkan jika belum ada.

```
library(dplyr) # memuat package `dplyr`  
data(mtcars) # menggunakan data dari environment  
set.seed(123) # untuk memastikan kita menghasilkan  
mtcars$id <- 1:nrow(mtcars) # menambahkan `id` jika belum ada  
train<-mtcars %>% dplyr::sample_frac(.75) # beri nama untuk set latihan  
test<-dplyr::anti_join(mtcars, train, by = 'id') # beri nama untuk set percobaan  
dim(train) # mengecek dimensi set latihan  
dim(test) # mengecek dimensi set percobaan
```

Catatan: Dataset ini hanya sebagai contoh, ini tidak cocok untuk tugas machine learning karena datanya terlalu sedikit.

8.10.2 caTools

Ada banyak pendekatan untuk mencapai partisi data. Untuk pendekatan yang lebih lengkap, lihat fungsi `createDataPartition` dalam package `caTools`.

```

library(caTools)
data(mtcars)
set.seed(123)
smp_size <- floor(0.75 * nrow(mtcars))
train_ind <- sample(seq_len(nrow(mtcars)),
                     size = smp_size)
train <- mtcars[train_ind, ]
test <- mtcars[-train_ind, ]
dim(train)
dim(test)
# memuat package `caTools`
# menggunakan data dari environment
# untuk memastikan kita mendapatkan
# 75% dari ukuran sampel
# memberi nama untuk set latihan
# memberi nama untuk set percobaan
# mengecek dimensi set latihan
# mengecek dimensi set percobaan

```

8.10.3 caret

Packages hebat lainnya yang dapat Anda gunakan untuk membagi dataset adalah `caret`.

```
library(caret) # memuat package `caret`  
data(mtcars) # menggunakan data dari environment
```


Chapter 9

Referensi

Berikut adalah beberapa referensi yang dapat Anda gunakan untuk mempelajari dasar-dasar pemrograman dalam bahasa R:

1. Venables, W.N. Smith D.M. and R Core Team. 2018. **An Introduction to R**: <https://cran.r-project.org/manuals.html>
2. R for Data Science: <https://r4ds.had.co.nz/>
3. Codecademy - Learn R : <https://www.codecademy.com/learn/learn-r>
4. DataCamp: <https://www.datacamp.com/courses/tech:r>
5. Primartha, R. 2018. **Belajar Machine Learning Teori dan Praktik**. Penerbit Informatika : Bandung
6. Rosadi,D. 2016. **Analisis Statistika dengan R**. Gadjah Mada University Press: Yogyakarta
7. STHDA. Running RStudio and Setting Up Your Working Directory - Easy R Programming .<http://www.sthda.com/english/wiki/running-rstudio-and-setting-up-your-working-directory-easy-r-programming#set-your-working-directory>
8. STDHA. **Getting Help With Functions In R Programming**.
<http://www.sthda.com/english/wiki/getting-help-with-functions-in-r-programming> .