Belajar Bahasa Pemrograman R

**(Dilengkapi Cara Membuat Aplikasi Olah Data
Sederhana dengan R Shiny)**

**Prana Ugiana Gio
Adhitya Ronnie Effendie**

USU Press

Art Design, Publishing & Printing

Gedung F, Pusat Sistem Informasi (PSI) Kampus USU Jl. Universitas No. 9 Medan 20155, Indonesia

Telp. 061-8213737; Fax 061-8213737

usupress.usu.ac.id

© USU Press 2017

Hak cipta dilindungi oleh undang-undang; dilarang memperbanyak menyalin, merekam sebagian atau seluruh bagian buku ini dalam bahasa atau bentuk apapun tanpa izin tertulis dari penerbit.

ISBN

Perpustakaan Nasional: Katalog Dalam Terbitan (KDT)

Belajar Bahasa Pemrograman R / Prana Ugiana Gio [dan] Adhitya Ronnie Effendie --Medan:
USU Press 2017.

iv, 199 p. ; illus.: 29 cm

Bibliografi

ISBN: 979-458-..

KATA PENGANTAR

Alhamdulillah, puji syukur atas kehadiran Allah SWT, karena atas izin-Nya, penulis dapat terus mempertahankan semangat untuk menulis, dan akhirnya dapat menyelesaikan buku ini. Hadirnya buku ini, tidak semata-mata atas usaha penulis sendiri, melainkan atas izin-Nya. Sungguh suatu kebahagiaan bagi penulis bisa berbagi sebagian kecil ilmu pengetahuan milik-Nya melalui buku yang berjudul **“Belajar Bahasa Pemrograman R”**.

Ucapan terima kasih penulis sampaikan kepada semua pihak yang telah membantu dalam rangka penyelesaian buku ini. Penulis menyadari bahwa buku ini tentunya masih perlu perbaikan, sehingga penulis mengharapkan kritik dan saran yang membangun dari para pembaca agar buku ini dapat menjadi lebih baik. Kritik dan saran dapat ditujukan ke alamat email gioprana89@gmail.com.

Medan, 10 April 2017

Prana Ugiana Gio
Adhitya Ronnie Effendie

DAFTAR ISI

BAB 1	
SEKILAS R.....	1-16
BAB 2	
VEKTOR.....	17-28
BAB 3	
FUNGSI DASAR R	29-46
BAB 4	
OPERATOR PEMBANDING DAN OPERATOR LOGIKA	47-52
BAB 5	
MATRIKS.....	53-74
BAB 6	
LIST.....	75-83
BAB 7	
DATA FRAME.....	84-108
BAB 8	
FACTOR.....	109-112
BAB 9	
FUNGSI TABLE DAN FTABLE.....	113-120
BAB 10	
STRUKTUR PEMROGRAMAN R.....	121-123
BAB 11	
FUNGSI.....	124-129
BAB 12	
MEMANIPULASI DATA DENGAN FUNGSI DPLYR	130-177
BAB 13	
SEKILAS MEMBUAT APLIKASI OLAH DATA SEDERHANA	
DENGAN SHINY	178-215

BAB 1

SEKILAS R

Berikut diberikan kutipan penjelasan singkat terkait R.

- ⇒ Norman Matloff dalam bukunya yang berjudul *“The Art of R Programming”* menyatakan sebagai berikut (2009).

“R is a scripting language for statistical data manipulation and analysis. It was inspired by, and is mostly compatible with, the statistical language S developed by AT&T. The name S, obviously standing for statistics, was an allusion to another programming language developed at AT&T with a one-letter name, C. S later was sold to a small firm, which added a GUI interface and named the result S-Plus.

R has become more popular than S/S-Plus, both because it’s free and because more people are contributing to it. R is sometimes called “GNU S.”

- ⇒ Lebih lanjut, Norman Matloff dalam bukunya yang berjudul *“The Art of R Programming”* menyatakan sebagai berikut (2009).

“Why use anything else? As the Cantonese say, yauh peng, yauh leng –“both inexpensive and beautiful. Its virtues:

➔ *A public-domain implementation of the widely-regarded S statistical language; R/S is the de facto standard among professional statisticians.*

➔ *Comparable, and often superior, in power to commercial products in most sense.*

➔ *Available for Windows, Macs, Linux.*

➔ *In addition to enabling statistical operations, it’s a general programming language, so that you can automate your analyses and create new functions.*

➔ *Object-oriented and functional programming structure.*

➔ *Your data sets are saved between sessions, so you don’t have to reload each time.*

➔ *Open-software nature means it’s easy to get help from the user community, and lots of new functions get contributed by users, many of which are prominent statisticians.*

I should warn you that one submits commands to R via text, rather than mouse clicks in a Graphical User Interface (GUI). If you can’t live without GUIs, you should consider using one of the free GUIs that have been develop for R, e.g. R commander or JGR. Note that R definitely does have graphics – tons of it. But the graphics are for the output, e.g. plots, not for the input.

The term object-oriented can be explained by example, say statistical regression. When you perform a regression analysis with other statistical package, say SAS or SPSS, you get a mountain of output. By

contrast, if you call `lm()` regression function in R, the function return an object containing all the results – estimated coefficients, their standard errors, residual, etc. You then pick and choose which parts of that object to extract, as you wish.”

⇒ Andrie de Vries dan Joris Meys dalam bukunya yang berjudul “*R for Dummies*” menyatakan sebagai berikut (2012).

“R is more a programming language than application. When you download R, you automatically download a console application that’s suitable for your operating system.

The R Development Core Team has put a lot of effort into making R available for different types of hardware and software. This means that R is available for Windows, Unix systems (such as Linux), and the Mac.

R itself is a powerful language that performs a wide variety of functions, such as data manipulation, statistical modeling, and graphics.”

⇒ Lebih lanjut, Andrie de Vries dan Joris Meys dalam bukunya yang berjudul “*R for Dummies*” menyatakan sebagai berikut (2012).

“In order to start working in R, you need to use an editing tool. Which editing tool you use depends to some extent on your operating system, because R does not provide a single graphical editor for all operating systems. The basic R install gives you the following:

→Windows: *A basic editor called RGui.*

→Mac OS X: *A basic R editor called R.app.*

→Linux: *There is no specific R editor on Linux, but you can use any editor (like Vim or Emacs) to edit your R code.*

Fortunately, there is an alternative called RStudio, third-party software that provides a consistent user interface regardless of operating system. In addition to demonstrating how to work with the Windows RGui, we also illustrate how to use RStudio.

The one thing that R is not is an application, which means that you have the freedom of selecting your own editing tools to interact with R. In this section we discuss the Windows R editor, RGui (short for R graphical user interface). Since the standard, basic R editors are so, well, basic, we also introduce you to RStudio. RStudio offers a richer editing environment than RGui and makes some common tasks easier and more fun.

As part of the process of downloading and installing R, you get the standard graphical user interface (GUI), called RGui. RGui gives you some tools to manage your R environment – most important, a console window. The console is where you type instructions, or scripts, and generally get R to do useful things for you.

⇒ Garrett Golemund dalam bukunya “*Hands-On Programming with R*” menyatakan sebagai berikut(2014).

*"Learning to program is important if you're serious about understanding data. There's no argument that data science must be performed on a computer, but you have a choice between learning a **graphical user interface (GUI)** or a **programming language**. Both Garrett and I strongly believe **that programming is a vital skill for everyone who works intensely with data**. While convenient, a **GUI is ultimately limiting**, because it hampers three properties essential for good data analysis:*

→ Reproducibility, The ability to-recreate a past analysis, which is crucial for good science.

→ Automation, The ability to rapidly re-create an analysis when data changes (as it always does).

→ Communication, Code is just text, it is easy to communicate. When learning, this makes it easy to get help - whether it's with email, Google, Stack Overflow, or elsewhere.

*Don't be afraid of programming! Anyone can learn to program with the right motivation, and this book is organized to keep you motivated. This is not a reference book; instead, it's structured around three hands-on challenges. Mastering these challenges will lead you through the basics of R programming and even into some intermediate topics, such as vectorized code, scoping, and S3 methods. Real challenges are a great way to learn, because you're not memorizing functions void of context; instead, you're learning functions as you need them to solve a real problem. **You'll learn by doing, not by reading.**"*

⇒ Lebih lanjut, Garrett Golemund dalam bukunya "*Hands-On Programming with R*" menyatakan sebagai berikut (2014).

"Not every programmer needs to be a data scientist, so not every programmer will find this book useful. You will find this book helpful if you're in one of the following categories:

- 1. You already use R as a statistical tool but would like to learn how to write your own functions and simulations with R.*
- 2. You would like to teach yourself how to program, and you see the sense of learning a language related to data science.*

*One of the biggest surprises in this book is that **I do not cover traditional applications of R, such as models and graphs; instead, I treat R purely as a programming language**. Why this narrow focus? R is designed to be a tool that helps scientists analyze data. It has many excellent functions that make plots and fit models to data. As a result, many statisticians learn to use R as if it were a piece of software—they learn which functions do what they want, and they ignore the rest.*

Greg compares R to SPSS, but he assumes that you use the full powers of R; in other words, that you learn how to program in R. If you only use functions that preexist in R, you are using R like SPSS: it is a bus that can only take you to certain places."

⇒ Garrett Golemund dalam bukunya "*Hands-On Programming with R*" menyatakan sebagai berikut (2014).

*R isn't a program that you can open and start using, like Microsoft Word or Internet Explorer. Instead, R is a computer language, like C, C++, or UNIX. **You use R by writing commands in the R language and asking your computer to interpret them**. In the old days, people ran R code in a UNIX terminal window—as if they were hackers in a movie from the 1980s. Now almost everyone uses R with an application called **RStudio**, and I recommend that you do, too.*

RStudio is an application like Microsoft Word—except that instead of helping you write in English, RStudio helps you write in R. I use RStudio throughout the book because it makes using R much easier. Also, the RStudio interface looks the same for Windows, Mac OS, and Linux. That will help me match the book to your personal experience. You can download RStudio for free. Just click the “Download RStudio” button and follow the simple instructions that follow. Once you’ve installed RStudio, you can open it like any other program on your computer—usually by clicking an icon on your desktop.”

W. John Braum dan Duncan J. Murdoch dalam bukunya "A First Course in Statistical Programming with R" menyatakan sebagai berikut (2007, 1:4).

“R is an open-source computing package which has seen a huge growth in popularity in the last few years. Being open source, it is easily obtainable by students and economical to install in our computing lab. One of us (Murdoch) is a member of the R core development team, and the other (Braun) is a co-author of a book on data analysis using R. These facts made it easy for us to choose R, but we are both strong believers in the idea that there are certain universals of programming, and in this text we try to emphasize those: it is not a manual about programming in R, it is a course in statistical programming that uses R.

This book uses R, which is an open-source package for statistical computing. “Open source” has a number of different meanings; here the important one is that R is freely available, and its users are free to see how it is written, and to improve it. R is based on the computer language S, developed by John Chambers and others at Bell Laboratories in 1976. In 1993 Robert Gentleman and Ross Ihaka at the University of Auckland wanted to experiment with the language, so they developed an implementation, and named it R. They made it open source in 1995, and hundreds of people around the world have contributed to its development. S-PLUS is a commercial implementation of the S language. Because both R and S-PLUS are based on the S language, much of what is described in what follows will apply without change to S-PLUS.

R can be downloaded from <http://cran.r-project.org/>!. Most users should download and install a binary version. This is a version that has been translated (by compilers) into machine language for execution on a particular type of computer with a particular operating system. R is designed to be very portable: it will run on Microsoft Windows, Linux, Solaris, Mac OSX, and other operating systems, but different binary versions are required for each. In this book most of what we do would be the same on any system, but when we write system-specific instructions, we will assume that readers are using Microsoft Windows.”

Berdasarkan pemaparan sebelumnya terkait R, dapat ditarik informasi bahwa R merupakan bahasa pemrograman statistika yang dapat digunakan untuk analisis dan manipulasi data statistika (pemodelan statistika), dan grafik. R diciptakan (*was created*) oleh Ross Ihaka dan Robert Gentleman (nama depan sama-sama diawali dengan huruf R & R, **Ross** dan **Robert**) dari departemen statistika, di Universitas Auckland, New Zealand. Saat ini R dikembangkan oleh *R Development Core Team*. Munculnya R terinspirasi oleh bahasa statistika dengan nama S (*statistical language S*). Bahasa statistika S diciptakan oleh John Chambers dan rekan, di *Bell Labs* atau *AT&T Bell Laboratories*. Selain bahasa statistika S, bahasa pemrograman C juga dikembangkan di *AT&T Bell Laboratories*.

Beberapa *software* seperti SPSS, Minitab, SAS, EViews, dan sebagainya, perlu membayar sejumlah uang, untuk mendapatkan izin menggunakan, sementara R gratis. Di samping itu, R juga tersedia untuk di-install di sistem operasi *Windows*, *Mac OS X*, dan *Linux*. Gambar 1.1 merupakan tampilan

website <https://cran.r-project.org/>. Di website tersebut, R dapat di-download sesuai dengan sistem operasi. Jika sistem operasi yang digunakan adalah *Windows*, maka pilih *Download R for Windows*.

Gambar 1.1

Setelah *software R for Windows* di-download, di-install, dan dijalankan, maka akan muncul tampilan seperti pada Gambar 1.3. Pada Gambar 1.3, merupakan tampilan editor dasar R (*basic R editor*) ketika R dijalankan. Tampilan editor dasar R tersebut dikenal dengan nama *RGui*. *RGui* merupakan nama editor dasar R ketika *R for Windows* dijalankan, sementara nama editor dasar R ketika *R for Mac OS X* dijalankan adalah *R.app*.

Gambar 1.2 Lambang R Versi 3.3.2

Gambar 1.3 Editor Dasar *R for Windows* (RGui)

```
R Console

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> A <- 1
> A
[1] 1
> B <- 5
> A+B
[1] 6
> C <- c(1,2,3)
> C
[1] 1 2 3
> |
```

Gambar 1.4

Pada Gambar 1.3, secara otomatis muncul tampilan *R Console*. Pada bagian *R Console*, perintah R dibuat. Terdapat tanda “>” pada *R Console*. Tanda tersebut disebut *prompt*. Di depan *prompt* perintah R ditulis. Gambar 1.4 menampilkan beberapa perintah R, beserta hasil dari eksekusi perintah tersebut. Pada Gambar 1.4, di depan tanda “[]” merupakan hasil dari eksekusi perintah R.

Selain editor R *RGui* (Gambar 1.3), terdapat alternatif editor R lain, yakni RStudio (Gambar 1.5 dan Gambar 1.6).

Gambar 1.5 Tampilan RStudio

Gambar 1.6 Tampilan RStudio

Pada RStudio menawarkan lebih banyak *editing environment* dibandingkan RGui. Gambar 1.7 menyajikan *website* <https://www.rstudio.com/products/rstudio/download/> di mana RStudio dapat di *download*.

Gambar 1.7

Dalam *website* <https://www.r-project.org/> juga menyajikan informasi seperti para kontributor R (Gambar 1.8), konferensi (Gambar 1.9), publikasi (Gambar 1.10), dan sebagainya.

[\[Home\]](#)

Download

[CRAN](#)

R Project

[About R](#)

[Logo](#)

[Contributors](#)

[What's New?](#)

[Reporting Bugs](#)

[Development Site](#)

[Conferences](#)

[Search](#)

R Foundation

[Foundation](#)

[Board](#)

[Members](#)

[Donors](#)

[Donate](#)

Help With R

Contributors

The current R is the result of a collaborative effort with contributions from all over the world. R was initially written by Robert Gentleman and Ross Ihaka—also known as “R & R” of the Statistics Department of the University of Auckland. Since mid-1997 there has been a core group with write access to the R source, currently consisting of

- Douglas Bates
- John Chambers
- Peter Dalgaard
- Robert Gentleman
- Kurt Hornik
- Ross Ihaka
- Michael Lawrence
- Friedrich Leisch
- Uwe Ligges
- Thomas Lumley
- Martin Maechler
- Martin Morgan
- Duncan Murdoch
- Paul Murrell
- Martyn Plummer
- Brian Ripley

Gambar 1.8

[\[Home\]](#)

Download

[CRAN](#)

R Project

[About R](#)

[Logo](#)

[Contributors](#)

[What's New?](#)

[Reporting Bugs](#)

[Development Site](#)

[Conferences](#)

[Search](#)

R Foundation

[Foundation](#)

[Board](#)

[Members](#)

[Donors](#)

[Donate](#)

Help With R

Conferences

The R project actively supports two conference series, organized regularly by members from the R community:

- useR!, providing a forum to the R user community.
- DSC, a platform for developers of statistical software.

A rough outline of the format for the respective conferences is given below; both conference formats were developed by the [Austrian Association for Statistical Computing \(AASC\)](#) in cooperation with the [R Foundation for Statistical Computing](#).

R-related conferences are coordinated by the “*R Foundation Conference Committee*” (RFCC) currently consisting of [Torsten Hothorn](#) ([Universität Zürich](#)) and [Achim Zeileis](#) ([Universität Innsbruck](#)) plus ex officio secretary and treasurer of the R Foundation. The RFCC can be contacted by email at R-conferences@r-project.org.

The R Foundation has a policy requiring [codes of conduct at R conferences](#).

useR! — International R User Conference

This is the main meeting of the R user and developer community, its program consisting of both invited and user-contributed presentations:

- The invited keynote lectures cover a broad spectrum of topics ranging from technical and R-related

Gambar 1.9

Home	About The R Journal
Current Issue	<i>The R Journal</i> is the open access, refereed journal of the R project for statistical computing. It features short to medium length articles covering topics that might be of interest to users or developers of R, including
Accepted Articles	Add-on packages: short introductions to R extension packages. Programmer's Niche: hints for programming in R. Help Desk: hints for newcomers explaining aspects of R that might not be so obvious from reading the manuals and FAQs. Applications: demonstrating how a new or existing technique can be applied in an area of current interest using R, providing a fresh view of such analyses in R that is of benefit beyond the specific application.
Archive	
Submissions	<i>The R Journal</i> intends to reach a wide audience and have a fast-track but thorough review process. Papers are expected to be reasonably short, clearly written, not too technical, and of course focused on R. Authors of refereed articles should take care to:
Editorial Board	<ul style="list-style-type: none"> • put their contribution in context, in particular discuss related R functions or packages; • explain the motivation for their contribution; • provide code examples that are reproducible.
	Continuing from R News , <i>The R Journal</i> also has a news section, including information on:
	Changes in R: new features of the latest release. Changes on CRAN: new add-on packages, manuals, binary distributions, mirrors,... Upcoming conferences: announcements of conferences related to R. Conference reports: overviews of past conferences related to R.

Gambar 1.10

R menjadi lebih populer dibandingkan S/S-Plus, dikarenakan R gratis dan orang-orang bisa berkontribusi dalam R. R sering juga disebut dengan “GNU S”. R atau S merupakan *de facto standard* di antara ahli statistika (*among professional statisticians*). R merupakan *open-software nature* yang berarti mudah untuk mendapatkan bantuan dari *user community*, dan terdapat banyak fungsi baru dari hasil kontribusi pengguna R, banyak di antaranya adalah para ahli statistika terkemuka (*prominent statisticians*).

Para pengembang (*developers*) dapat dengan mudah menuliskan *software* mereka sendiri (*write their own software*) dan mendistribusikannya dalam bentuk *add-on packages*. Karena begitu mudahnya membuat (*create*) suatu *package* maka terdapat ribuan *packages* yang telah tersedia (Gambar 1.11).

Faktanya banyak yang baru (tidak begitu baru) metode statistika yang dipublikasi pada suatu *package* R (Andrie de Vries dan Joris Meys, 2012). Sebagai contoh *package* MASS dan DiscriMiner, yang mana memuat metode statistika analisis diskriminan linear.

Gambar 1.11 Tampilan *Packages*

Sebagai contoh fungsi *ggplot()* yang terdapat dalam *package ggplot2*. Fungsi *ggplot()* dapat digunakan untuk menyajikan data seperti dalam bentuk diagram batang, histogram, diagram lingkaran, diagram pencar, dan sebagainya. Gambar 1.12 merupakan contoh terapan dari fungsi *ggplot*.

```
ggplot(simpan, aes(pendapatan, pengeluaran)) + geom_point(aes(color = jenis, shape = jenis))
```

Gambar 1.12

Sebagai contoh salah satu kelebihan dari R, ketika melakukan analisis regresi dalam R, misalkan digunakan menggunakan fungsi *lm()*, fungsi tersebut akan mengembalikan sebuah objek yang mengandung seluruh hasil (*containing all the results*), seperti *estimated coefficients*, *standard errors*, *residuals*, dan sebagainya. Kemudian dari hasil tersebut, dapat diambil atau dipilih, bagian mana dari objek tersebut untuk dipilih untuk ditampilkan, sesuai yang diinginkan.

Berbeda dengan, ketika melakukan analisis regresi dalam *software*, katakanlah SPSS atau MINITAB, akan diperoleh atau ditampilkan banyak *output*, seperti *correlation*, *coefficient of determination*, dan sebagainya (*mountain of output*). Perhatikan ilustrasi Gambar 1.13. Pada Gambar 1.13 merupakan hasil SPSS untuk analisis regresi.

***Output1 [Document1] - SPSS Statistics Viewer**

File Edit View Data Transform Insert Format Analyze Graphs Utilities Add-ons Window Help

Model R R Square Adjusted R Square Std. Error of the Estimate

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.853 ^a	.728	.694	1.06967

a. Predictors: (Constant), X

ANOVA^b

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	24.446	1	24.446	21.366	.002 ^a
	Residual	9.154	8	1.144		
	Total	33.600	9			

a. Predictors: (Constant), X
b. Dependent Variable: Y

Coefficients^a

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	1.896	.791		2.398	.043
	X	.661	.143	.853	4.622	.002

a. Dependent Variable: Y

Gambar 1.13

Sementara pada Gambar 1.14 dan Gambar 1.15 merupakan hasil dengan menggunakan R untuk analisis regresi.

RStudio

File Edit Code View Plots Session Build Debug Tools Help

Go to file/function

Console ~/ ↻

```
> x <- c(1,2,3,4,5,6,7,6,7,9)
> y <- c(2,3,4,5,4,6,7,7,8,6)
> regresi <- lm(formula = y~x)
> summary(regresi)
```

Call:
lm(formula = y ~ x)

Residuals:

Min	1Q	Median	3Q	Max
-1.8429	-0.4723	0.1304	0.4741	1.4786

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	1.8964	0.7907	2.398	0.04329 *
x	0.6607	0.1429	4.622	0.00171 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.07 on 8 degrees of freedom
Multiple R-squared: 0.7276, Adjusted R-squared: 0.6935
F-statistic: 21.37 on 1 and 8 DF, p-value: 0.001705

Gambar 1.14


```

RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function

Console ~/
Call:
lm(formula = y ~ x)

Residuals:
    Min       1Q   Median       3Q      Max
-1.8429 -0.4723  0.1304  0.4741  1.4786

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.8964     0.7907   2.398  0.04329 *
x            0.6607     0.1429   4.622  0.00171 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.07 on 8 degrees of freedom
Multiple R-squared:  0.7276,    Adjusted R-squared:  0.6935
F-statistic: 21.37 on 1 and 8 DF,  p-value: 0.001705

> regres$coefficient[1] #Memilih untuk menampilkan intersep
(Intercept)
1.896429
> regres$coefficient[2] #Memilih untuk menampilkan koefisien regresi x
x
0.6607143
> regres$residual #Memilih untuk menampilkan nilai residual
      1      2      3      4      5      6      7      8
-0.5571429 -0.2178571  0.1214286  0.4607143 -1.2000000  0.1392857  0.4785714  1.1392857
      9     10
 1.4785714 -1.8428571
>

```

Gambar 1.15

Hasil analisis regresi dalam SPSS (Gambar 1.13), secara umum menampilkan tumpukan *output* seperti korelasi, koefisien determinasi, dan sebagainya, sedangkan dalam R, dapat dipilih hasil mana saja yang hendak diperlihatkan (Gambar 1.15).

R dapat dikatakan sebagai bahasa berbasis vektor (*vectorized-based language*). Suatu vektor dapat dipandang sebagai suatu baris atau kolom dari bilangan atau *text*. Sebagai contoh daftar dari bilangan {1,2,3,4,5} dapat dipandang sebagai vektor (Andrie de Vries dan Joris Meys, 2012). Perhatikan ilustrasi berikut (Gambar 1.16).

```

RStudio
File Edit Code View Plots Session Build
Go to file/function

Console ~/
> x <- 1:5
> x
[1] 1 2 3 4 5
>

```

Gambar 1.16

Berdasarkan Gambar 1.16, perintah R pada baris pertama `x <- 1:5` dapat diartikan menugaskan bilangan 1,2,3,4,5 ke vektor `x`, kemudian perintah R pada baris kedua `x` menampilkan seluruh elemen dalam vektor `x`. Perhatikan lagi ilustrasi Gambar 1.17.

Gambar 1.17

Berdasarkan Gambar 1.17, perintah R `x+2` berarti menjumlahkan nilai 2 terhadap masing-masing elemen vektor `x`. Untuk melakukan hal tersebut dalam bahasa pemrograman yang tidak berbasis vektor (*that aren't vectorized*), diperlukan menggunakan *explicit loop*, untuk mencapai hasil yang sama. Sebagai contoh diberikan ilustrasi dalam bahasa pemrograman C (Gambar 1.18).

Gambar 1.18

Oleh karena itu R sangat ampuh (*this feature of R extremely powerful*) dikarenakan R dapat melakukan banyak operasi (*many operation*) hanya dengan satu langkah, seperti yang telah diperlihatkan dari Gambar 1.16 hingga Gambar 1.18 (Andrie de Vries dan Joris Meys, 2012).

Norman Matloff (2009) menyatakan R memiliki 2 *mode*, yakni *interactive mode* dan *batch mode*. Gambar 1.19 merupakan contoh tampilan R dalam *interactive mode*. Pada *interactive mode* dijumpai tanda *prompt* “>”. Pada *interactive mode*, perintah R ditulis di depan tanda “>”, seperti pada Gambar 1.16 dan Gambar 1.17.

Gambar 1.19

Sementara pada *batch mode*, seluruh perintah R ditulis terlebih dahulu, seperti pada Gambar 1.21. Perintah R pada Gambar 1.21 bertujuan untuk membuat histogram, kemudian tampilan histogram tersebut disimpan dalam file dengan format **pdf**, dengan nama **membuat_histogram.pdf** (Gambar 1.23 dan Gambar 1.24). Setelah seluruh perintah R ditulis, seperti pada Gambar 1.21, selanjutnya, seluruh perintah R dieksekusi (perhatikan Gambar 1.22).

Gambar 1.20

Gambar 1.21

Gambar 1.22

Pada Gambar 1.23, telah terbentuk file dengan nama **membuat_histogram.pdf**, yang apabila dibuka, seperti pada Gambar 1.24.

Gambar 1.23

Gambar 1.24

BAB 2

VEKTOR

2.1 Membuat Vektor

Pertama-tama, aktifkan *software* RStudio terlebih dahulu (Gambar 2.1). Lambang RStudio diperlihatkan pada Gambar 2.2. Pada Gambar 2.1 sedang berada pada bagian *Console*. Pada bagian *R Console*, perintah R dibuat. Terdapat tanda “>” pada *R Console*. Tanda tersebut disebut *prompt*. Di depan *prompt* perintah R ditulis.

Gambar 2.1 Tampilan RStudio pada Bagian *Console*

Gambar 2.2 Lambang RStudio

Berdasarkan Gambar 2.3, perintah R pada baris pertama `x <- c(4,8,1,2)` dapat diartikan menugaskan (*assign*) bilangan 4,8,1,2 ke vektor dengan nama `x`, kemudian perintah R pada baris kedua `x` menampilkan seluruh elemen dalam vektor `x`. Dalam membuat vektor, melibatkan fungsi `c()`. Suatu fungsi ditandai dengan tanda buka kurung “(” dan tutup kurung “)”. Huruf “c” merupakan singkatan dari “*concatenate*”, dalam bahasa Indonesia dapat berarti “menggabungkan”.

Gambar 2.3

Pada Gambar 2.4, perintah R `y <- c(1,1,2,1)` dapat diartikan menugaskan (*assign*) bilangan 1,1,2,1 ke vektor dengan nama `y`, kemudian perintah R `y` menampilkan seluruh elemen dalam vektor `y`.

```
Console ~/
> x <- c(4,8,1,2)
> x
[1] 4 8 1 2
> y<- c(1,1,2,1)
> y
[1] 1 1 2 1
> |
```

Gambar 2.4

2.2 Hasil Eksekusi Perintah R

```
Console ~/
> x <- c(4,8,1,2)
> x
[1] 4 8 1 2
> y<- c(1,1,2,1)
> y
[1] 1 1 2 1
> z <- c(4,5,3,2,3,4,5,6,3,4,5,5,6,7,7,4,3,
2,2,3,4,5,6,7,8,4,3,2,5)
> z
[1] 4 5 3 2 3 4 5 6 3 4 5 5 6 7 7 4 3 2
[19] 2 3 4 5 6 7 8 4 3 2 5
>
```

Gambar 2.5

Perhatikan Gambar 2.5. Pada Gambar 2.5, terdapat tanda “[]”, yang merupakan hasil dari eksekusi perintah R. Perhatikan bahwa “[1]” berarti hasil atau *output* yang ditampilkan dimulai dari *output* yang ke-1. Sementara “[19]” berarti hasil atau *output* yang ditampilkan dimulai dari *output* yang ke-19.

2.3 Indeks Vektor

Indeks vektor dimulai dari 1. Berdasarkan Gambar 2.5, pada vektor `x`, diketahui:

- ⇒ Nilai 4 berada pada indeks ke-1 dari vektor `x`.
- ⇒ Nilai 8 berada pada indeks ke-2 dari vektor `x`.
- ⇒ Nilai 1 berada pada indeks ke-3 dari vektor `x`.
- ⇒ Nilai 2 berada pada indeks ke-4 dari vektor `x`.

2.4 Menampilkan Elemen Vektor berdasarkan Indeks

```
Console ~/
> x <- c(4,8,1,2)
> x[1] #Menampilkan elemen ke-1 pada vektor x
[1] 4
> x[2] #Menampilkan elemen ke-2 pada vektor x
[1] 8
> x[4] #Menampilkan elemen ke-4 pada vektor x
[1] 2
>
```

Gambar 2.6

Perhatikan Gambar 2.6. Pada Gambar 2.6, perintah R **x[1]** berarti menampilkan elemen pada indeks ke-1 dari vektor **x**, perintah R **x[4]** berarti menampilkan elemen pada indeks ke-4 pada vektor **x**, dan sebagainya. Pada Gambar 2.6, terdapat tanda "#". Kalimat di depan tanda "#" dimaksudkan sebagai pemberi keterangan (*comment*).

2.5 Mengganti Elemen Vektor

```
Console ~/
> x
[1] 4 8 1 2
> x[1]
[1] 4
> x[1] <- 100000
> x[1]
[1] 1e+05
> x[2]
[1] 8
> x[2] <- 12
> x[2]
[1] 12
> |
```

Gambar 2.7

Pada Gambar 2.7, diketahui nilai 4 berada pada indeks ke-1 di vektor **x**. Andaikan nilai 4 akan diganti dengan 100000. Perintah R untuk mengganti nilai 4 menjadi nilai 100000 adalah **x[1] <- 100000** (Perhatikan Gambar 2.7). Pada Gambar 2.7, *output* 1e+05 berarti $10^5 = 100000$. Selanjutnya diketahui nilai 8 berada pada indeks ke-2 di vektor **x**. Andaikan nilai 8 akan diganti dengan 12. Perintah R untuk mengganti nilai 8 menjadi nilai 12 adalah **x[2] <- 12** (Perhatikan Gambar 2.7).

2.6 Operator Penugasan

Perhatikan Gambar 2.8. Perintah R **x <- 1,2,3,4,5** memberikan pesan kesalahan. Sementara perintah R **x <- c(1,2,3,4,5)** tidak memberikan pesan kesalahan, begitu juga perintah R **y <- c(2,3,4,5,6)** tidak memberikan pesan kesalahan. Namun perintah R **z <- 1,2,3,4,5** memberikan pesan kesalahan. Perintah R **z <- y** dapat diartikan menugaskan seluruh elemen pada vektor **y** ke vektor **z**. Jadi, seluruh elemen

dari vektor **z** sekarang adalah seluruh elemen dari vektor **y**. Perhatikan bahwa perintah **R x <- 1** tidak memberikan pesan kesalahan, namun perintah **R x <- 1, 2** memberikan pesan kesalahan.

```
> x <- 1,2,3,4,5
Error: unexpected ',' in "x <- 1,"
> x <- c(1,2,3,4,5)
> y <- c(2,3,4,5,6)
> z <- 1,2,3,4,5
Error: unexpected ',' in "z <- 1,"
> z <- x
> z
[1] 1 2 3 4 5
> z <- c(y)
> z
[1] 2 3 4 5 6
> z = x
> z
[1] 1 2 3 4 5
> x <- 1
> x
[1] 1
> x <- 1, 2
Error: unexpected ',' in "x <- 1,"
> x <- 1 2
Error: unexpected numeric constant in "x <- 1 2"
> y <-5
> y
[1] 5
> y=4
> y
[1] 4
>
```

Gambar 2.8

Operator standar penugasan (*standard assignment operator*) dalam R adalah “<-”, meskipun terdapat beberapa operator penugasan seperti “->”, “=”, dan fungsi **assign()**. Norman Matloff (2009) menyatakan sebagai berikut.

“The standard assignment operator in R is <-. However, there are also ->, =, and even the assign() function.”

Perhatikan Gambar 2.9. Gambar 2.9 diberikan ilustrasi penggunaan operator penugasan “<-”, “->”, “=”, dan fungsi **assign()**. Perintah **R x <- c(1,2,3)** dapat diartikan menugaskan bilangan 1,2,3 ke vektor **x**. Perintah **R x -> y** dapat diartikan menugaskan bilangan-bilangan pada vektor **x** ke vektor **y**. Dengan kata lain, seluruh elemen dari vektor **y** adalah seluruh elemen dari vektor **x**. Perintah **R assign(“z”, y)** dapat diartikan menugaskan bilangan-bilangan pada vektor **y** ke vektor **z**. Dengan kata lain, seluruh elemen dari vektor **z** adalah seluruh elemen dari vektor **y**. Perintah **R assign(“a”,c(1,2,3,4,5,6,7,8))** dapat diartikan menugaskan bilangan 1,2,3,4,5,6,7,8 ke vektor **a**.


```

Source
Console ~/
> x <- c(1,2,3)
> x
[1] 1 2 3
> x -> y
> y
[1] 1 2 3
> assign("z",y)
> z
[1] 1 2 3
> assign("a",c(1,2,3,4,5,6,7,8))
> a
[1] 1 2 3 4 5 6 7 8
> b=a
> b
[1] 1 2 3 4 5 6 7 8
> |

```

Gambar 2.9

2.7 Berbagai Cara Menampilkan Elemen Vektor berdasarkan Indeks

Gambar 2.10 diberikan ilustrasi berbagai cara untuk menampilkan elemen suatu vektor. Perintah R `x <- c(4,5,2,1,9,8,5,6,10,13)` berarti menugaskan bilangan 4,5,2,1,9,8,5,6,10,13 ke vektor `x`. Sebagai contoh diketahui:

- ⇒ Elemen pada indeks ke-1 dari vektor `x` adalah 4.
- ⇒ Elemen pada indeks ke-2 dari vektor `x` adalah 5.
- ⇒ Elemen pada indeks ke-3 dari vektor `x` adalah 2.
- ⇒ Elemen pada indeks ke-10 dari vektor `x` adalah 13.

```

Console ~/
> x <- c(4,5,2,1,9,8,5,6,10,13)
> y <- c(1,2,6)
> z <- c(9,3,4,5)
> x
[1] 4 5 2 1 9 8 5 6 10 13
> x[4]
[1] 1
> x[c(1,3)]
[1] 4 2
> x[c(8,10)]
[1] 6 13
> x[-c(1,2,3,4,5)]
[1] 8 5 6 10 13
> x[3:7]
[1] 2 1 9 8 5
> x[-3]
[1] 4 5 1 9 8 5 6 10 13
> x[c(-3)]
[1] 4 5 1 9 8 5 6 10 13
> x[-3,1]
Error in x[-3, 1] : incorrect number of dimensions
> x[3,1]
Error in x[3, 1] : incorrect number of dimensions
> x[y]
[1] 4 5 8
> x[-y]
[1] 2 1 9 5 6 10 13
> x[z]
[1] 10 2 1 9
> z[-z]
[1] 9 3
>

```

Gambar 2.10

Perhatikan bahwa:

- ⇒ Perintah R `x[4]` berarti menampilkan elemen dari vektor `x` pada indeks ke-4, yakni 1.
- ⇒ Perintah R `x[c(1,3)]` berarti menampilkan elemen dari vektor `x` pada indeks ke-1 dan indeks ke-3, yakni 4 dan 2.
- ⇒ Perintah R `x[c(8,10)]` berarti menampilkan elemen dari vektor `x` pada indeks ke-8 dan indeks ke-10, yakni 6 dan 13.
- ⇒ Perintah R `x[-c(1,2,3,4,5)]` berarti menampilkan elemen dari vektor `x` yang bukan pada indeks ke-1,2,3,4,5. Dengan kata lain, menampilkan elemen dari vektor `x` pada indeks ke-6,7,8,9,10, yakni 8,5,6,10,13.
- ⇒ Perintah R `x[3:7]` berarti menampilkan elemen dari vektor `x` mulai dari indeks ke-3 sampai dengan indeks ke-7. Dengan kata lain, menampilkan elemen dari vektor `x` pada indeks ke-3,4,5,6,7, yakni 2,1,9,8,5.
- ⇒ Perintah R `x[-3]` berarti menampilkan elemen dari vektor `x` yang bukan pada indeks ke-3. Dengan kata lain, menampilkan elemen dari vektor `x` pada indeks ke-1,2,4,5,6,7,8,9,10. Perhatikan bahwa perintah R `x[-3]` memberikan hasil yang sama dengan perintah R `x[-c(3)]`. Namun perintah R `x[-3,1]` dan perintah R `x[3,1]` memberikan pesan kesalahan.
- ⇒ Perintah R `x[y]` sama saja dengan `x[c(1,2,6)]`. Perhatikan bahwa 1, 2, dan 6 merupakan elemen dari vektor `y`.
- ⇒ Perintah R `x[-y]` sama saja dengan `x[-c(1,2,6)]` atau sama saja dengan `x[c(3,4,5,7,8,9,10)]`.
- ⇒ Perintah R `x[z]` sama saja dengan `x[c(9,3,4,5)]`. Perhatikan bahwa 9, 3, 4, dan 5 merupakan elemen dari vektor `z`. Perintah R `x[z]` berarti menampilkan elemen dari vektor `x` pada indeks ke-9,3,4,5.

2.8 Mengetahui Panjang Vektor dengan Fungsi `length()`

Untuk mengetahui panjang (*length*) suatu vektor, dapat digunakan fungsi `length()`. Perhatikan Gambar 2.11. Berdasarkan Gambar 2.11, diketahui panjang dari vektor `x` adalah 5, yang mana elemennya adalah 1, 5, 3, 6, dan 7. Panjang dari vektor `y` adalah 4, yang mana elemennya adalah 4, 4, 5, dan 4. Pada Gambar 2.11, digunakan fungsi `length()` untuk mengetahui panjang dari suatu vektor.

Gambar 2.11

2.9 Membuat Vektor dengan Fungsi `vector()`

Fungsi `vector()` digunakan untuk membuat suatu vektor. Perhatikan Gambar 2.12. Berdasarkan Gambar 2.12, perintah R `C <- vector(length=2)` berarti membuat vektor `C` dengan panjang 2. Perintah R `length(C)` untuk mengetahui panjang vektor `C`, yang mana panjang vektor `C` adalah 2. Jika elemen dari vektor `C` ditampilkan, maka kedua elemen tersebut adalah `FALSE FALSE`. Secara bawaan (*default*), ketika suatu vektor dibuat dengan fungsi `vector()` dengan panjang tertentu, vektor tersebut berjenis logika dengan nilai elemen seluruhnya `FALSE`. Fungsi `mode()` digunakan untuk menentukan jenis data yang tersimpan dalam suatu vektor. Setelah vektor `C` dibuat dengan panjang 2, selanjutnya mengisi bilangan 4 pada indeks ke-1 dari vektor `C`, mengisi bilangan 12 pada indeks ke-2 dari vektor `C`. Setelah itu dicek kembali jenis dari vektor `C` dengan fungsi `mode()`. Jenis vektor `C` sekarang adalah *numeric*, karena elemen dari vektor `C` saat ini adalah angka, yakni 4 dan 12.

```
> C <- vector(length=2)
> length(C)
[1] 2
> C
[1] FALSE FALSE
> mode(C)
[1] "logical"
> C[1] <-4
> C[2] <-12
> C
[1] 4 12
> mode(C)
[1] "numeric"
```

Gambar 2.12

```
> A <- vector(length=5)
> length(A)
[1] 5
> A
[1] FALSE FALSE FALSE FALSE FALSE
> mode(A)
[1] "logical"
> A[2]=40000
> A
[1] 0 40000 0 0 0
> mode(A)
[1] "numeric"
> |
```

Gambar 2.13

Berdasarkan Gambar 2.13, perintah R `A<- vector(length=5)` berarti membuat vektor `A` dengan panjang 5. Perintah R `length(A)` untuk mengetahui panjang vektor `A`, yang mana panjang vektor `A` adalah 5. Jika elemen dari vektor `A` ditampilkan, maka kelima elemen tersebut adalah `FALSE FALSE FALSE FALSE FALSE`. Secara bawaan (*default*), ketika suatu vektor dibuat dengan fungsi `vector()` dengan

panjang tertentu, vektor tersebut berjenis logika dengan nilai elemen seluruhnya FALSE. Fungsi *mode()* digunakan untuk menentukan jenis data yang tersimpan dalam suatu vektor. Setelah vektor **A** dibuat dengan panjang 5, selanjutnya mengisi bilangan 40000 pada indeks ke-2 dari vektor **A**. Untuk indeks ke-1,3,4,5 tidak diisi bilangan. Selanjutnya ditampilkan seluruh nilai dari vektor **A**, yakni 0, 40000, 0, 0, dan 0. Secara bawaan (*default*) nilai vektor **A** pada indeks ke-1,3,4,5, diisi dengan 0. Selanjutnya diperiksa kembali jenis dari vektor **A** dengan fungsi *mode()*. Jenis vektor **A** sekarang adalah *numeric*, karena elemen dari vektor **A** saat ini adalah angka, yakni 0, 40000, 0, 0, dan 0.

2.9 Fungsi *mode()* Untuk Mengetahui Jenis Data Suatu Vektor

Untuk memeriksa jenis data suatu vektor, dapat digunakan fungsi *mode()*. Perhatikan ilustrasi Gambar 2.14. Berdasarkan Gambar 2.14, diketahui jenis data dari vektor **A** adalah *numeric*. Data dari vektor **A** adalah angka, yakni 4, 5, dan 3. Jenis data dari vektor **B** adalah *character*, dengan data “Ugi”, “Egi”, dan “Ronnie”. Setiap elemen dari vektor berjenis *character* diapit tanda petik ganda “ ”. Jenis data dari vektor **C** adalah *logical*. Data pada jenis *logical* hanya ada dua, yakni TRUE dan FALSE.

```

RStudio
File Edit Code View Plots Session Build Debug Tools Help
Source
Console ~/
> A <- c(4,5,3)
> A
[1] 4 5 3
> mode(A) #Untuk mengetahui jenis vektor A
[1] "numeric"
> B <- c("Ugi","Egi","Ronnie")
> B
[1] "Ugi" "Egi" "Ronnie"
> mode(B)
[1] "character"
> C <- c(TRUE, FALSE, TRUE)
> C
[1] TRUE FALSE TRUE
> mode(C)
[1] "logical"
>

```

Gambar 2.14

2.10 Operator Penjumlahan +, Pengurangan -, Perkalian *, Pembagian /, Pangkat ^, dan Sisa %%

Perhatikan Gambar 2.15. Pada Gambar 2.15, diketahui elemen-elemen dari vektor **z** adalah 3, 5, 7, 5, yang diperoleh dari perintah `R z <- x+y`. Diketahui:

- ⇒ Elemen pertama dari vektor **z** adalah 3, yang diperoleh dari hasil jumlah antara elemen pertama dari vektor **x**, yakni 1, dan elemen pertama dari vektor **y**, yakni 2.
- ⇒ Elemen kedua dari vektor **z** adalah 5, yang diperoleh dari hasil jumlah antara elemen kedua dari vektor **x**, yakni 2, dan elemen kedua dari vektor **y**, yakni 3.

- ⇒ Elemen ketiga dari vektor **z** adalah 7, yang diperoleh dari hasil jumlah antara elemen ketiga dari vektor **x**, yakni 3, dan elemen ketiga dari vektor **y**, yakni 4.
- ⇒ Elemen keempat dari vektor **z** adalah 5, yang diperoleh dari hasil jumlah antara elemen keempat dari vektor **x**, yakni 4, dan elemen keempat dari vektor **y**, yakni 1.

```

RStudio
File Edit Code View Plots Session Build Debug Tools Help
Source
Console
> x <- c(1,2,3,4)
> x
[1] 1 2 3 4
> y <- c(2,3,4,1)
> y
[1] 2 3 4 1
> z <- x+y
> z
[1] 3 5 7 5
> y <- c(z)
> y
[1] 3 5 7 5
> x*y
[1] 3 10 21 20
>

```

Gambar 2.15

Berdasarkan Gambar 2.15, perintah R **y <- c(z)** dapat diartikan menugaskan seluruh elemen dari vektor **z** ke vektor **y**. Dengan kata lain, sekarang seluruh elemen pada vektor **y** merupakan seluruh elemen pada vektor **z**. Perintah R **x*y** dapat diartikan:

- ⇒ Mengalikan elemen pertama pada vektor **x** dengan elemen pertama pada vektor **y** ($1 \times 3 = 3$).
- ⇒ Mengalikan elemen kedua pada vektor **x** dengan elemen kedua pada vektor **y** ($2 \times 5 = 10$).
- ⇒ Mengalikan elemen ketiga pada vektor **x** dengan elemen ketiga pada vektor **y** ($3 \times 7 = 21$).
- ⇒ Mengalikan elemen keempat pada vektor **x** dengan elemen keempat pada vektor **y** ($4 \times 5 = 20$).

Gambar 2.16 diberikan berbagai contoh perhitungan. Berdasarkan Gambar 2.16, diketahui:

- ⇒ Elemen dari vektor **X** adalah 1, 2, dan 3. Elemen dari vektor **Y** adalah 3, 2, dan 1.
- ⇒ Pada Gambar 2.16, terdapat perintah R **X^Y**, yang berarti $1^3, 2^2, 3^1$ atau 1, 4, 3.

Hasil dari perintah R **9%%4** adalah 1 ($9 = 4 * 2 + 1$), hasil dari perintah R **9%%5** adalah 4 ($9 = 5 * 1 + 4$), hasil dari perintah R **-5%%3** adalah 1 ($-5 = 3 * -2 + 1$), dan hasil dari perintah R **-5%%4** adalah 3 ($-5 = 4 * -2 + 3$).

```
> X <- c(1,2,3)
> Y <- c(3,2,1)
> X+Y #Operasi penjumlahan
[1] 4 4 4
> X-Y #Operasi pengurangan
[1] -2 0 2
> X*Y #Operasi perkalian
[1] 3 4 3
> X/Y #Operasi pembagian
[1] 0.3333333 1.0000000 3.0000000
> X^Y #Operasi Pangkat
[1] 1 4 3
> X%%Y #Operasi sisa
[1] 1 0 0
> 5^2
[1] 25
> 9%%4
[1] 1
> 9%%5
[1] 4
> 9%%6
[1] 3
> 2*X
[1] 2 4 6
> 2*Y
[1] 6 4 2
> -5%%3 # -5 = 3*(-2) + 1
[1] 1
> -5%%4 # -5 = 4*(-2) + 3
[1] 3
>
```

Gambar 2.16

2.11 Memberi Nama pada Elemen Vektor dengan Fungsi *names()*

Elemen dalam vektor juga dapat diberi nama dengan fungsi *names()*. Perhatikan ilustrasi pada Gambar 2.17. Berdasarkan Gambar 2.17, digunakan fungsi *names()* untuk memberi nama pada elemen vektor **X**. Elemen vektor **X** pada indeks ke-1diberi nama “Sangat Tidak Puas”, elemen vektor **X** pada indeks ke-2diberi nama “Tidak Puas”, elemen vektor **X** pada indeks ke-3diberi nama “Biasa Saja”, elemen vektor **X** pada indeks ke-4diberi nama “Puas”, dan elemen vektor **X** pada indeks ke-5diberi nama “Sangat Puas”. Kemudian perhatikan perintah R **names(X) <- NULL** bermaksud untuk menghapus nama pada elemen di vektor **X**.

Gambar 2.17

2.12 Tidak Tersedia (*Not Available* (NA))

Perhatikan Gambar 2.18. Perintah R `X <-c(5,4,3,5)` berarti menugaskan bilangan 5, 4, 3 dan 5 ke vektor `X`. Nilai pada indeks ke-1 dari vektor `X` adalah 5, nilai pada indeks ke-2 dari vektor `X` adalah 4, nilai pada indeks ke-3 dari vektor `X` adalah 3, nilai pada indeks ke-4 dari vektor `X` adalah 5, nilai pada indeks ke-5 dari vektor `X` adalah tidak ada atau tidak tersedia atau *not available* (disingkat “NA”), nilai pada indeks ke-6 dari vektor `X` adalah tidak ada atau tidak tersedia atau *not available* (disingkat “NA”).

Gambar 2.18

2.13 Menghapus Elemen Vektor

Perhatikan Gambar 2.19. Perintah R `X <-c(5,4,3,5)` berarti menugaskan bilangan 5, 4, 3 dan 5 ke vektor `X`. Andaikan nilai 4 pada indeks ke-2 dari vektor `X` ingin dihapus, sehingga elemen dari vektor `X` sekarang adalah 5, 3, dan 5. Maka perintah R untuk menghapus nilai 4 pada indeks ke-2 dari vektor `X` adalah `X <- X[-2]`. Perintah R `X <- X[-2]` dapat juga diartikan menugaskan bilangan selain pada indeks ke-2, ke vektor `X`. Sehingga elemen pada vektor `X` saat ini adalah 5, 3, dan 5. Selanjutnya perintah R `Y <- Y[-c(3,6,7)]` berarti menghapus elemen dari vektor `Y` pada indeks ke-3,6, dan 7.

The image shows a screenshot of the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Tools, and Help. Below the menu bar is a toolbar with icons for creating a new file, opening a file, saving, printing, and a search bar labeled 'Go to file/function'. The main window is divided into two panes. The top pane is labeled 'Source' and is currently empty. The bottom pane is labeled 'Console' and contains the following R code and its output:

```
> X <-c(5,4,3,5)
> X
[1] 5 4 3 5
> X <- X[-2] #Menghapus elemen pada indeks ke-2, dari vektor X
> X
[1] 5 3 5
> Y <-c(3,4,5,7,8,2,10)
> Y
[1] 3 4 5 7 8 2 10
> Y <-Y[-c(3,6,7)] #Menghapus elemen pada indeks ke-3,6, dan 7, dari vektor Y
> Y
[1] 3 4 7 8
>
```

Gambar 2.19

BAB 3

FUNGSI DASAR R

3.1 Mengurutkan Elemen Vektor dengan Fungsi *sort()*

Fungsi *sort()* dapat digunakan untuk mengurutkan elemen vektor. Gambar 3.1 diberikan ilustrasi mengurutkan bilangan dari yang terkecil ke terbesar, dan sebaliknya dari yang terbesar ke terkecil. Sementara pada Gambar 3.2, mengurutkan elemen vektor berjenis *character*, dimulai dari huruf abjad yang terkecil ke terbesar ($A \rightarrow Z$), dan sebaliknya dari huruf abjad yang terbesar ke terkecil ($Z \rightarrow A$).

Gambar 3.1

Gambar 3.2

3.2 Menghitung Selisih dari Data Berurutan dalam Vektor dengan Fungsi *diff()*

Misalkan diberikan data 100, 50, 70, 80, 60. Misalkan dilakukan perhitungan sebagai berikut.

$$\Rightarrow 50 - 100 = -50$$

$$\Rightarrow 70 - 50 = 20$$

$$\Rightarrow 80 - 70 = 10$$

$$\Rightarrow 60 - 80 = -20$$

Sehingga hasil akhirnya adalah -50,20,10,-20. Perhatikan Gambar 3.3. Perintah R **diff(X)*-1** dimaksudkan untuk melakukan perhitungan sebagai berikut.

- ⇒ $100 - 50 = 50$
- ⇒ $50 - 70 = -20$
- ⇒ $70 - 80 = -10$
- ⇒ $80 - 60 = 20$

```

Console ~/
> x <- c(100,50,70,80,60)
> diff(x)
[1] -50  20  10 -20
> diff(x)*-1
[1]  50 -20 -10  20
>

```

Gambar 3.3

3.3 Menjumlahkan Seluruh Bilangan dalam Vektor dengan Fungsi *sum()*

Fungsi *sum()* dapat digunakan untuk menjumlahkan seluruh bilangan dalam suatu vektor. Misalkan bilangan 50, 60, 70, dan 80 ditugaskan ke vektor **A**. Maka jumlah dari seluruh bilangan dalam vektor **A** adalah 260.

$$50 + 60 + 70 + 80 = 260$$

Perhatikan Gambar 3.4. Pada Gambar 3.4, perintah R *sum(A[c(2,4)])* berarti menjumlahkan bilangan dalam vektor **A** pada indeks ke-2 dan indeks ke-4.

```

~/
> A=c(50,60,70,80)
> sum(A)
[1] 260
> sum(A[c(2,4)]) #Menjumlahkan bilangan pada vektor x pada indeks ke-2 dan indeks ke-4
[1] 140
>

```

Gambar 3.4

3.4 Menghitung Akar Pangkat Dua dalam Vektor dengan Fungsi *sqrt()*

Fungsi *sqrt()* dapat digunakan untuk menghitung akar pangkat dua dari suatu bilangan dalam vektor. Perhatikan ilustrasi pada Gambar 3.5. Perintah *sqrt(B)* berarti $\sqrt{4}, \sqrt{9}, \sqrt{16}, \sqrt{25}$ atau 2,3,4,5. Selanjutnya perintah R *C^(1/3)* berarti menghitung akar pangkat tiga dari bilangan-bilangan pada vektor **C**, yakni $\sqrt[3]{8}, \sqrt[3]{27}, \sqrt[3]{64}$ atau 2, 3, 4.

```

> B <- c(4,9,16,25)
> sqrt(B) #Menghitung akar pangkat dua untuk setiap bilangan di vektor B
[1] 2 3 4 5
> C <- c(8,27,64)
> C^(1/3) #Menghitung akar pangkat tiga untuk setiap bilangan di vektor C
[1] 2 3 4
>

```

Gambar 3.5

3.5 Fungsi *max()* dan *min()* untuk Menentukan Nilai Maksimum dan Minimum dalam Vektor

Fungsi *max()* dan *min()* dapat digunakan untuk menentukan nilai maksimum dan minimum dalam vektor. Misalkan bilangan 10,25,90, 75, 95, 57 ditugaskan ke vektor **A**. Nilai maksimum bilangan di vektor **A** adalah 95, sementara nilai minimum bilangan di vektor **A** adalah 10. Perhatikan Gambar 3.6.

```

> A=c(10,25,90,75,95,57)
> max(A)
[1] 95
> min(A)
[1] 10
>

```

Gambar 3.6

Fungsi *max()* dan *min()* juga dapat digunakan untuk vektor berjenis *character*. Perhatikan ilustrasi pada Gambar 3.7.

```

> B <- c("Nanas","Jeruk","Anggur","Stroberi","Markisa")
> max(B) #Hasilnya adalah stroberi, dikarenakan huruf "s" adalah huruf awal kata paling besar dari nama buah di vektor B
[1] "stroberi"
> min(B) #Hasilnya adalah anggur, dikarenakan huruf "A" adalah huruf awal kata paling kecil dari nama buah di vektor B
[1] "Anggur"
>

```

Gambar 3.7

3.6 Fungsi *exp()* (Eksponensial)

Exp merupakan singkatan dari *exponential* atau eksponensial. Nilai dari eksponensial adalah 2,71828182845...

$$\exp = 2,71828182845$$

$$\exp^1 = 2,71828182845^1 = 2,71828182845$$

$$\exp^2 = 2,71828182845^2 = 7,389056096$$

Berikut diberikan contoh penggunaan fungsi *exp()* dalam R (Gambar 3.8).

```
Console R Markdown x
~/
> exp(1)
[1] 2.718282
> exp(2)
[1] 7.389056
>
```

Gambar 3.8

3.7 *pi* atau π

Pi atau π bernilai 3,141593 ... Berikut diberikan contoh penggunaan dari *pi* dalam R (Gambar 3.9).

```
Source
Console R Markdown x
~/
> pi
[1] 3.141593
> pi^2
[1] 9.869604
>
```

Gambar 3.9

3.8 Menentukan Jumlah Angka di belakang Koma dengan Fungsi *options()*

Nilai pi adalah 3.141593... Misalkan hanya ingin ditampilkan 2 digit angka di belakang koma dari nilai pi. Gambar 3.10 ilustrasi dalam R menggunakan fungsi *options()* untuk menampilkan hanya 2 digit angka di belakang koma dari bilangan pi.

```
Console R Markdown x
~/
> pi
[1] 3.141592654
> options(digits=3) #Menetapkan 2 angka di belakang koma
> pi
[1] 3.14
> options(digits=5) #Menetapkan 4 angka di belakang koma
> pi
[1] 3.1416
>
```

Gambar 3.10

3.9 Menampilkan Serangkaian Bilangan dengan Fungsi *seq()* dan Pengulangan Serangkaian Bilangan dengan Fungsi *rep()*

Fungsi *seq()* ("sequence" atau serangkaian) dapat digunakan untuk menampilkan serangkaian bilangan. Misalkan ingin ditampilkan bilangan dari 1 sampai 11, dengan jarak 2, yakni 1, 3, 5, 7, 9, 11. Gambar 3.11 digunakan fungsi *seq()* untuk menampilkan bilangan tersebut.

```

Console ~/
> seq(from=1, to=11, by=2)
[1] 1 3 5 7 9 11
> |

```

Gambar 3.11

Selanjutnya misalkan ingin ditampilkan bilangan dari 1 sampai 2, dengan jarak 0,2, yakni 1, 1,2, 1,4, 1,6, 1,8, dan 2. Gambar 3.12 digunakan fungsi *seq()* untuk menampilkan bilangan tersebut.

```

Console ~/
> seq(from=1, to=2, by=0.2)
[1] 1.0 1.2 1.4 1.6 1.8 2.0
>

```

Gambar 3.12

Misalkan hanya ingin ditampilkan 4 bilangan, dimulai dari 2 kemudian 2,5, 3 dan 3,5, dengan jarak adalah 0,5. Maka perintah R untuk menampilkan bilangan-bilangan tersebut adalah **seq(from=2, by=0.5, length.out=4)**. Kemudian misalkan ingin ditampilkan 6 bilangan, dimulai dari 2 kemudian 2,5, 3, 3,5, 4, dan 4,5, dengan jarak adalah 0,5. Maka perintah R untuk menampilkan bilangan-bilangan tersebut adalah **seq(from=2, by=0.5, length.out=6)**. Perhatikan Gambar 3.13.

```

Console ~/
> seq(from=2, by=0.5, length.out=4)
[1] 2.0 2.5 3.0 3.5
> seq(from=2, by=0.5, length.out=6)
[1] 2.0 2.5 3.0 3.5 4.0 4.5
> length(seq(from=2, by=0.5, length.out=4))
[1] 4
> length(seq(from=2, by=0.5, length.out=6))
[1] 6
>

```

Gambar 3.13

Selanjutnya fungsi *seq()* dan *rep()* akan digunakan secara bersama, untuk menampilkan bilangan 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3. Fungsi *rep()* (“repeat” atau mengulangi) di sini digunakan untuk melakukan pengulangan rangkaian bilangan 1, 2, 3 sebanyak 3 kali. Perhatikan Gambar 3.14.

```

Console ~/
> rep(1:3,4) #Menampilkan bilangan 1, 2, 3, yang diulang sebanyak 4 kali
[1] 1 2 3 1 2 3 1 2 3 1 2 3
> sort(rep(1:3,4))
[1] 1 1 1 1 2 2 2 2 3 3 3 3
> sort(rep(1:3,4), decreasing=TRUE)
[1] 3 3 3 3 2 2 2 2 1 1 1 1
> |

```

Gambar 3.14

Misalkan akan ditampilkan bilangan 4, 2, 0, 4, 2, 0. Perhatikan Gambar 3.15.

```
Console ~/
> rep(seq(from=4, to=0, by=-2), 2)
[1] 4 2 0 4 2 0
> sort(rep(seq(from=4, to=0, by=-2), 2))
[1] 0 0 2 2 4 4
> sort(rep(seq(from=4, to=0, by=-2), 2), decreasing=TRUE)
[1] 4 4 2 2 0 0
>
> |
```

Gambar 3.15

Misalkan lagi akan ditampilkan bilangan 1, 1, 7, 7, 7, 7, 7, 3, 3. Perhatikan Gambar 3.16.

```
Console ~/
> rep(c(1,7,3), c(2,5,2))
[1] 1 1 7 7 7 7 7 3 3
> rep(c(1,7,3), times=c(2,5,2))
[1] 1 1 7 7 7 7 7 3 3
>
> |
```

Gambar 3.16

3.10 Membuat Distribusi Frekuensi dan Distribusi Probabilitas dengan Fungsi *table()*

Fungsi *table()* dapat digunakan untuk membuat suatu distribusi frekuensi. Misalkan suatu vektor bernama **A** menyimpan bilangan 10, 10, 30, 10, 30, 10, 10, 40, 40, 70, 90, 70, 80, 60, 60, 90. Diketahui nilai 10 sebanyak 5, nilai 30 sebanyak 2, nilai 40 sebanyak 2, nilai 60 sebanyak 2, nilai 70 sebanyak 2, nilai 80 sebanyak 1, dan nilai 90 sebanyak 2. Gambar 3.17 diperlihatkan penggunaan fungsi *table()* untuk menyajikan distribusi frekuensi dan probabilitas dari data pada vektor **A**. Perhatikan bahwa probabilitas dari nilai 10 adalah 0,3125 yang diperoleh $\frac{5}{16} = 0,3125$, probabilitas dari nilai 80 adalah 0,0625 yang diperoleh $\frac{1}{16} = 0,0625$, dan seterusnya.

```
Console ~/
> A <-c(10, 10, 30, 10, 30, 10, 10, 40, 40, 70, 90, 70, 80, 60, 60, 90)
> table(A)
A
10 30 40 60 70 80 90
 5  2  2  2  2  1  2
> table(A)/length(A) #Distribusi probabilitas
A
 10   30   40   60   70   80   90
0.3125 0.1250 0.1250 0.1250 0.1250 0.0625 0.1250
> length(A) #Mengetahui panjang vektor A
[1] 16
>
```

Gambar 3.17

Gambar 3.18 disajikan contoh distribusi frekuensi dan probabilitas untuk vektor berjenis *character*.

```

Console ~/
> hasil <- c("ya","tidak","ya","ya","ya","ya","tidak","ya")
> table(hasil)
hasil
tidak  ya
      2   6
> table(hasil)/length(hasil)
hasil
tidak  ya
0.25 0.75
> mode(hasil)
[1] "character"
> mode(table(hasil))
[1] "numeric"
>
> C <- table(hasil)
> C
hasil
tidak  ya
      2   6
> C <- as.numeric(C)
> C
[1] 2 6
>

```

Gambar 3.18

3.11 Fungsi *factor()* dan Fungsi *level()* untuk Mengetahui Keragaman Data

Fungsi *factor()* dalam R dapat digunakan untuk mengetahui keragaman *level* pada data vektor. Misalkan diberikan data sebagai berikut.

ikan, ikan, udang, ikan, udang, ikan, ikan, udang

Berdasarkan data tersebut, terdapat dua *level*, yakni ikan dan udang. Misalkan diberikan data sebagai berikut.

sarjana,diploma, sarjana, pengangguran, sarjana, diploma, diploma, pengangguran

Berdasarkan data tersebut, terdapat tiga *level*, yakni sarjana, diploma, dan pengangguran. Misalkan diberikan data sebagai berikut.

1, 2, 3, 2, 1, 4, 6, 3, 2, 6, 2, 4, 5, 3, 3, 2

Berdasarkan data tersebut, terdapat tiga *level*, yakni 1, 2, 3, 4, 5, dan 6. Ilustrasi dalam R disajikan pada Gambar 3.19 dan Gambar 3.20.

```

Console ~/
> A <- c("ikan","ikan","udang","ikan","udang","ikan","ikan","udang")
> factor(A)
[1] ikan ikan udang ikan udang ikan ikan udang
Levels: ikan udang
> levels(factor(A))
[1] "ikan" "udang"
> B <- levels(factor(A))
> B
[1] "ikan" "udang"
> length(B)
[1] 2
> mode(B)
[1] "character"
>
>
> C <- c("sarjana","diploma","sarjana","pengangguran","sarjana","diploma","diploma","pengangguran")
> factor(C)
[1] ikan udang
Levels: ikan udang
> levels(factor(C))
[1] "diploma" "pengangguran" "sarjana"
> D <- levels(factor(C))
> D
[1] "diploma" "pengangguran" "sarjana"
> length(D)
[1] 3
> mode(D)
[1] "character"
>
>

```

Gambar 3.19

```

Console ~/
> E <- c(1, 2, 3, 2, 1, 4, 6, 3, 2, 6, 2, 4, 5, 3, 3, 2)
> factor(E)
[1] 1 2 3 2 1 4 6 3 2 6 2 4 5 3 3 2
Levels: 1 2 3 4 5 6
> F <- levels(factor(E))
> F
[1] "1" "2" "3" "4" "5" "6"
> mode(F)
[1] "character"
> F <- as.numeric(F)
> F
[1] 1 2 3 4 5 6
> mode(F)
[1] "numeric"
> |

```

Gambar 3.20

3.12 Mengkonversi Vektor *Character* Menjadi Vektor *Numeric* dengan Fungsi *as.numeric()*

Fungsi *as.numeric()* dapat digunakan untuk mengkonversi vektor berjenis *character* menjadi vektor berjenis *numeric*. Ilustrasi diberikan pada Gambar 3.20.

3.13 Membuat Grafik Batang dengan Fungsi *barplot()*

Fungsi *barplot()* dalam R berfungsi untuk menyajikan data dalam bentuk grafik batang. Misalkan vektor **A** menyimpan data 10, 10, 10, 10, 20, 20, 30, 30, 30, 30, 30, 30. Gambar 3.21 diberikan ilustrasi dalam R untuk menyajikan data pada vektor **A** dalam bentuk grafik batang.

Gambar 3.21

Gambar 3.22

Berdasarkan Gambar 3.21, perhatikan bahwa untuk data dengan nilai 10 mempunyai frekuensi sebanyak 4, data dengan nilai 20 mempunyai frekuensi sebanyak 2, dan data dengan nilai 30 mempunyai frekuensi sebanyak 6. Grafik batang di atas dapat diatur agar disajikan berdasarkan probabilitas (Gambar 3.22). Perhatikan bahwa nilai 0,3333, 0,16667, dan 0,5 masing-masing merupakan probabilitas dari nilai 10, 20, dan 30.

3.14 Memplot Data dengan Fungsi *plot()*

Misalkan vektor bernama **A** menyimpan data 10,10,10,10,10,20,20,20,30,30,40. Diketahui nilai 10 muncul sebanyak 5, nilai 20 sebanyak 3, nilai 30 sebanyak 2, dan nilai 40 sebanyak 1. Gambar 3.23 digunakan fungsi *plot()* untuk memplot data yang tersimpan dalam vektor **A**.

Gambar 3.23

Alternatif lain disajikan pada Gambar 3.24.

Gambar 3.24

3.15 Membulatkan Bilangan dengan Fungsi *round()*, Fungsi *ceiling()*, Fungsi *floor()*, dan Fungsi *signif()*

Gambar 3.25 dan Gambar 3.26 diberikan contoh ilustrasi dalam R penggunaan fungsi *round()*, fungsi *ceiling()*, fungsi *floor()*, dan fungsi *signif()*. Pada penggunaan fungsi *round()*, 2,45 dibulatkan menjadi 2, namun 2,54 dibulatkan menjadi 3. Pada bilangan 2,45, satu angka di belakang koma adalah $4 \leq 5$, maka 2,45 dibulatkan menjadi 2. Pada bilangan 2,54, satu angka di belakang koma adalah $5 \geq 5$, maka 2,54 dibulatkan menjadi 3. Sementara pada penggunaan fungsi *ceiling()*, 2,12 dan 2,94 dibulatkan menjadi bilangan 3, meskipun pada bilangan 2,12, satu angka di belakang koma adalah $1 < 5$, bilangan 2,12 tetap dibulatkan menjadi 3. Namun pada fungsi *floor()*, 2,45 dan 2,656 dibulatkan menjadi 2.

```
Console ~/
> round(2.45)
[1] 2
> round(2.54)
[1] 3
> ceiling(2.12)
[1] 3
> ceiling(2.94545)
[1] 3
> floor(2.45)
[1] 2
> floor(2.656)
[1] 2
> round(2.17645, digits=2)
[1] 2.18
> round(2.172324, digits=2)
[1] 2.17
>
```

Gambar 3.25

```
Console ~/
> signif(2.34567,digits=2)
[1] 2.3
> signif(2.34567,digits=3)
[1] 2.35
> signif(2.34567,digits=1)
[1] 2
> round(2.34567,digits=1)
[1] 2.3
> round(2.34567,digits=2)
[1] 2.35
> round(2.34567,digits=3)
[1] 2.346
>
```

Gambar 3.26

3.16 Mengkonversi Vektor *Numeric* Menjadi Vektor *Character* dengan Fungsi *as.character()*

Fungsi *as.character()* dapat digunakan untuk mengkonversi vektor berjenis *numeric* menjadi vektor berjenis *character*. Ilustrasi diberikan pada Gambar 3.27.

```

Console ~/
> A <- c(1,2,3,4,2,4)
> A
[1] 1 2 3 4 2 4
> mode(A)
[1] "numeric"
> A <- as.character(A)
> A
[1] "1" "2" "3" "4" "2" "4"
> mode(A)
[1] "character"
> |

```

Gambar 3.27

3.17 Mengambil Beberapa Karakter dengan Fungsi *substr()* dan Fungsi *substring()*

Fungsi *substr()* dan fungsi *substring()* dapat digunakan untuk mengambil beberapa karakter dari serangkaian karakter. Ilustrasi diberikan pada Gambar 3.28 dan Gambar 3.29. Berdasarkan Gambar 3.28, diketahui vektor **Buah** menyimpan data “Mangga”, “Anggur”, “Pepaya”, “Semangka”. Selanjutnya perintah R **substr(Buah,1,1)** berarti mengambil satu huruf yang dimulai dari huruf pertama nama buah di vektor **Buah**, yang mana hasilnya “M”, “A”, “P”, dan “S”. Selanjutnya perintah R **substr(Buah,1,3)** berarti mengambil tiga huruf yang dimulai dari huruf pertama nama buah di vektor **Buah** (dari huruf pertama sampai huruf ketiga), yang mana hasilnya “Man”, “Ang”, “Pep”, dan “Sem”. Perintah R **substr(Buah,3,4)** berarti mengambil dua huruf yang dimulai dari huruf ketiga nama buah di vektor **Buah** (dari huruf ketiga sampai huruf keempat).

```

Console ~/
> Buah <- c("Mangga","Anggur","Pepaya","Semangka")
> substr(Buah,1,1) #Mengambil satu huruf yang dimulai dari huruf pertama nama buah
[1] "M" "A" "P" "S"
> substr(Buah,1,3) #Mengambil tiga huruf yang dimulai dari huruf pertama nama buah
[1] "Man" "Ang" "Pep" "Sem"
> substr(Buah,3,4) #Mengambil dua huruf yang dimulai dari huruf ketiga nama buah
[1] "ng" "gg" "pa" "ma"
>
> A <- c(6584, 1834, 5323, 9534)
> substr(A,1,1)
[1] "6" "1" "5" "9"
> substr(A,1,2)
[1] "65" "18" "53" "95"
>

```

Gambar 3.28

Pada Gambar 3.29 digunakan fungsi *substring()*. Perintah R **substring(Buah,3)** berarti mengambil seluruh huruf yang dimulai dari huruf ketiga nama buah di vektor **Buah**, yang mana hasilnya adalah “ngga”, “ggur”, “paya”, dan “mangka”.

```

Console ~/
> Buah <- c("Mangga","Anggur","Pepaya","Semangka")
> substring(Buah,3) #Mengambil seluruh huruf yang dimulai dari huruf ketiga nama buah
[1] "ngga" "ggur" "paya" "mangka"
>

```

Gambar 3.29

3.18 Mengkonversi Bilangan Negatif Menjadi Bilangan Tak Negatif dalam Vektor dengan Fungsi *abs()*

Misalkan bilangan -2, 4, -1, -5, 7 ditugaskan ke vektor bernama **A**. Andaikan diinginkan seluruh elemen di vektor **A** bernilai tak negatif, yakni menjadi 2, 4, 1, 5, dan 7. Fungsi *abs()* dapat digunakan untuk melakukan hal tersebut. Perhatikan Gambar 3.30.

```
Console ~/
> A <- c(-2, 4, -1, -5, 7)
> A
[1] -2  4 -1 -5  7
> abs(A)
[1] 2 4 1 5 7
>
```

Gambar 3.30

3.19 Fungsi *str_replace_all()* dan Fungsi *strsplit()*

Gambar 3.31 diberikan ilustrasi penggunaan fungsi *str_replace_all()*. Penggunaan fungsi *str_replace_all()* pada Gambar 3.31 dimaksudkan untuk menghapus spasi, sehingga “1 12 3 34 3 323 2” menjadi “11233433232”. Pada Gambar 3.32 diberikan ilustrasi penggunaan fungsi *strsplit()*. Diketahui elemen dari vektor **A** adalah “1 23 3234 43 4 534”.

Jenis data dari vektor **A** adalah *character*. Selanjutnya digunakan fungsi *strsplit()* untuk memisahkan angka (masih dalam jenis *character*) di antara spasi, sehingga menjadi “1” “23” “3234” “43” “4” “534”. Hasil dari perintah R `mode(strsplit(A, " "))` adalah *list*, yang kemudian dikonversi menjadi *character* dengan perintah R `unlist(strsplit(A, " "))`.

```
Console ~/
> library(stringr)
> A <- c("1 12 3 34 3 323 2")
> A
[1] "1 12 3 34 3 323 2"
> str_replace_all(string=A, pattern=" ", repl="")
[1] "11233433232"
>
```

Gambar 3.31

```

Source

Console ~/
> A <- c("1 23 3234 43 4 534")
> A
[1] "1 23 3234 43 4 534"
> mode(A)
[1] "character"
> strsplit(A, " ")
[[1]]
[1] "1" "23" "3234" "43" "4" "534"

> mode(strsplit(A, " "))
[1] "list"
> unlist(strsplit(A, " "))
[1] "1" "23" "3234" "43" "4" "534"
> unlist(strsplit(A, "[ ]"))
[1] "1" "23" "3234" "43" "4" "534"
> unlist(strsplit(A, " ", fixed=TRUE))
[1] "1" "23" "3234" "43" "4" "534"
> mode(unlist(strsplit(A, " ")))
[1] "character"
> B <- unlist(strsplit(A, " "))
> mode(B)
[1] "character"
> B <- as.numeric(B)
> B
[1] 1 23 3234 43 4 534
> mode(B)
[1] "numeric"
> |

```

Gambar 3.32

Gambar 3.33, Gambar 3.34, dan Gambar 3.35 diberikan ilustrasi tambahan mengenai penggunaan fungsi *strsplit()*.

```

Console ~/
> A <- c("1 23 3234 43 4 534 5 43434 32323")
> A
[1] "1\t23\t3234\t43\t4\t534\t5\t43434\t32323"
> mode(A)
[1] "character"
> strsplit(A, "\t")
[[1]]
[1] "1" "23" "3234" "43" "4" "534" "5" "43434" "32323"

> B <- unlist(strsplit(A, "\t"))
> mode(B)
[1] "character"
> B <- as.numeric(B)
> B
[1] 1 23 3234 43 4 534 5 43434 32323
> mode(B)
[1] "numeric"
>

```

Gambar 3.33

```

Console ~/
> A <- c("1 23 3234      43      4 34      5      43434      32323")
> A <- strsplit(A,"\\t")
> A <- unlist(A)
> A <- strsplit(A, " ")
> A
[[1]]
[1] "1"      "23"     "3234"

[[2]]
[1] "43"

[[3]]
[1] "4"      "34"

[[4]]
[1] "5"

[[5]]
[1] "43434"

[[6]]
[1] "32323"

> mode(A)
[1] "list"
> B <- unlist(A)
> mode(B)
[1] "character"
> B <- as.numeric(B)
> B
[1]      1      23     3234      43      4      34      5 43434 32323
> mode(B)
[1] "numeric"
>

```

Gambar 3.34

```

Console ~/
> A <- c("ddfdsfsdfdf5")
> strsplit(A, NULL)[[1]]
[1] "d" "d" "f" "d" "s" "f" "s" "d" "f" "d" "f" "5"
>
>

```

Gambar 3.35

3.20 ggplot2

Huruf "gg" pada *ggplot2* merupakan kependekan dari *grammar of graphics*. Ide dalam menggunakan package ini adalah kita bisa membuat *graph* dari komponent: **dataset**, **geom** yang merupakan tanda visual yang mewakili data dan **coordinate system**. Untuk menampilkan

Gambar 3.36

nilai datanya, gunakan variabel sebagai *aesthetic properties* dari **geom** seperti *size*, *color* dan lokasi **x** dan **y**.

Gambar 3.37

Untuk memudahkan pemahaman, ketikkan perintah berikut ,

```
ggplot(data = mpg, aes(x = cty, y = hwy))
```

maka akan menghasilkan plot yang tidak ada isinya atau bisa kita sebut sebagai layer pertama yang nantinya dapat kita tambahkan layer-layer berikutnya.

Gambar 3.38

Untuk menambahkan layer baru ke plot, kita dapat menggunakan **geom_*()** atau **stat_*()**. Seperti di bawah ini,

```
ggplot(mpg, aes(hwy, cty)) +  
  geom_point(aes(color=cyl)) +  
  geom_smooth(method = "lm") +  
  coord_cartesian() +  
  scale_color_gradientn() +  
  theme_bw()
```

yang akan menghasilkan,

Gambar 3.39

3.21 Lanjutan ggplot2 (Satu Variabel)

Gambar 3.40

3.22 Lanjutan ggplot2 (Dua Variabel)

Discrete X, Continuous Y

```
g <- ggplot(mpg, aes(class, hwy))
```

```
g + geom_bar(stat = "identity")
```

x, y, alpha, color, fill, linetype, size, weight

```
g + geom_boxplot()
```

lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight

```
g + geom_dotplot(binaxis = "y", stackdir = "center")
```

x, y, alpha, color, fill

```
g + geom_violin(scale = "area")
```

x, y, alpha, color, fill, linetype, size, weight

Discrete X, Discrete Y

```
h <- ggplot(diamonds, aes(cut, color))
```

```
h + geom_jitter()
```

x, y, alpha, color, fill, shape, size

Gambar 3.41

3.23 Stats

Selain menggunakan **geom**, untuk membuat layer dapat digunakan **stat**. Tak jarang dijumpai adanya keinginan untuk membuat plot yang merupakan transformasi dari asli.

Gambar 3.42

BAB 4

OPERATOR PEMBANDING DAN OPERATOR LOGIKA

4.1 Operator Pembandingan atau Operator Relasional

Operator pembandingan, atau disebut juga dengan istilah operator relasional, biasa digunakan untuk membandingkan dua bilangan. Tabel 4.1 disajikan berbagai macam operator pembandingan.

Tabel 4.1 Operator Pembandingan

Operator Pembandingan	Keterangan
==	Sama dengan (bukan penugasan)
!=	Tidak sama dengan
>	Lebih dari
<	Kurang dari
>=	Lebih dari atau sama dengan
<=	Kurang dari atau sama dengan

Gambar 4.1 diberikan ilustrasi penggunaan operator pembandingan.

```
Console ~/ ↗
> A <- c(1,2,3,4)
> B <- c(2,2,2,2)
> A>B
[1] FALSE FALSE TRUE TRUE
> A<B
[1] TRUE FALSE FALSE FALSE
> B>A
[1] TRUE FALSE FALSE FALSE
> B<A
[1] FALSE FALSE TRUE TRUE
> A==B
[1] FALSE TRUE FALSE FALSE
> A!=B
[1] TRUE FALSE TRUE TRUE
> A>=B
[1] FALSE TRUE TRUE TRUE
> 2>3
[1] FALSE
> 2<3
[1] TRUE
> 2!=3
[1] TRUE
> 2==3
[1] FALSE
> 2==2
[1] TRUE
>
```

Gambar 4.1

Berdasarkan Gambar 4.1, vektor **A** ditugaskan untuk menyimpan bilangan 1, 2, 3, dan 4, sementara vektor **B** ditugaskan untuk menyimpan bilangan 2, 2, 2, dan 2. Perintah R **A>B** berarti:

- ⇒ $1 > 2$ (salah atau FALSE). 1 merupakan elemen pertama dari vektor **A**, sementara 2 merupakan elemen pertama dari vektor **B**.
- ⇒ $2 > 2$ (salah atau FALSE). 2 merupakan elemen kedua dari vektor **A**, sementara 2 merupakan elemen kedua dari vektor **B**.
- ⇒ $3 > 2$ (benar atau TRUE). 3 merupakan elemen ketiga dari vektor **A**, sementara 2 merupakan elemen ketiga dari vektor **B**.
- ⇒ $4 > 2$ (benar atau TRUE). 4 merupakan elemen keempat dari vektor **A**, sementara 2 merupakan elemen keempat dari vektor **B**.

Perhatikan bahwa hasil dari perintah R $\mathbf{A} > \mathbf{B}$ adalah FALSE, FALSE, TRUE, dan TRUE (perhatikan Gambar 4.1). Perhatikan juga bahwa perintah R $\mathbf{2} \neq \mathbf{3}$ (2 tidak sama dengan 3) memberikan hasil TRUE. Perintah R $\mathbf{2} == \mathbf{3}$ (2 sama dengan 3) memberikan hasil FALSE. Hasil dari penggunaan operator pembandingan memberikan hasil nilai logika, yakni TRUE atau FALSE. Gambar 4.2 diberikan contoh penggunaan operator pembandingan lebih dari 1 kali dalam satu perintah R.

```

Console ~/
> 1<2>3
Error: unexpected '>' in "1<2>"
> 1<2
[1] TRUE
> (1<2)>3
[1] FALSE
> 1<2>3<4
Error: unexpected '>' in "1<2>"
> (((1<2)>3)<4)
[1] TRUE
> |

```

Gambar 4.2

4.2 Operator Logika

Operator logika pada umumnya digunakan untuk membuat keadaan logika dengan menggunakan dua buah kondisi atau sebuah kondisi, yang mana bergantung pada operator logika yang digunakan. Suatu kondisi dapat bernilai benar, yakni **TRUE**, atau dapat bernilai salah, yakni **FALSE**.

Operator logika “dan” yang dilambangkan “&&” menggunakan dua buah kondisi, yakni

(kondisi1)&&(kondisi2)

Suatu kondisi dapat bernilai benar atau salah. Tabel 4.2 disajikan nilai kebenaran untuk operasi logika “dan”.

Tabel 4.2 Nilai Kebenaran untuk Operasi “dan”

Kondisi Pertama	Kondisi Kedua	Hasil &&
Salah	Salah	Salah
Salah	Benar	Salah
Benar	Salah	Salah
Benar	Benar	Benar

Pada operasi “dan” akan menghasilkan nilai benar, jika kedua kondisi bernilai benar. Selain dari pada itu, bernilai salah. Gambar 4.3 diberikan ilustrasi dalam R terkait operasi “dan”.

```
Console ~/
> (2>3)&&(1>3) #salah && salah maka salah
[1] FALSE
> (2>3)&&(4>3) #salah && benar maka salah
[1] FALSE
> (5>3)&&(2>3) #benar && salah maka salah
[1] FALSE
> (5>3)&&(3>=3) #benar && benar maka benar
[1] TRUE
> (5>3)&&(3>=3)&&(3<2)
[1] FALSE
> (5>3)&&(3>=3)&&(1<2)
[1] TRUE
> |
```

Gambar 4.3

Operator logika “atau” yang dilambangkan dengan “||” juga menggunakan dua buah kondisi, yakni

(kondisi1)|| (kondisi2)

Suatu kondisi dapat bernilai benar atau salah. Tabel 4.3 disajikan nilai kebenaran untuk operasi logika “atau”.

Tabel 4.3 Nilai Kebenaran untuk Operasi “atau”

Kondisi Pertama	Kondisi Kedua	Hasil
Salah	Salah	Salah
Salah	Benar	Benar
Benar	Salah	Benar
Benar	Benar	Benar

Pada operasi “||” akan menghasilkan nilai salah, jika kedua kondisi bernilai salah. Selain dari pada itu, bernilai benar. Gambar 4.4 diberikan ilustrasi dalam R terkait operasi “||”.

```

Console ~/
> (2>3)|| (1>3) #salah || salah maka salah
[1] FALSE
> (2>3)|| (4>3) #salah || benar maka benar
[1] TRUE
> (5>3)|| (2>3) #benar || salah maka salah
[1] TRUE
> (5>3)|| (3>=3) #benar || benar maka benar
[1] TRUE
> (5>3)|| (3>=3)|| (3<2)
[1] TRUE
> (5>3)|| (3>=3)|| (1<2)
[1] TRUE
>

```

Gambar 4.4

Operator logika “bukan” yang dilambangkan dengan “!” berguna untuk membalikkan suatu nilai kebenaran dari suatu kondisi. Perintah R `!(2>3)` akan menghasilkan nilai kebenaran TRUE. Perhatikan Gambar 4.5.

```

Console ~/
> 2>5
[1] FALSE
> !(2>5)
[1] TRUE
> 3==3
[1] TRUE
> !(3==3)
[1] FALSE
>

```

Gambar 4.5

4.3 Lebih Lanjut Penggunaan Operator Pembandingan dan Operator Logika

Perhatikan Gambar 4.6. Perintah R `A[A>5]` berarti menampilkan elemen dari vektor **A** dengan syarat lebih besar dari 5, yakni 8 (pada indeks ke-1), 7 (pada indeks ke-4), dan 6 (pada indeks ke-8).

```

Console ~/
> A <- c(8,4,5,7,2,3,1,6)
> A
[1] 8 4 5 7 2 3 1 6
> A[A>5]
[1] 8 7 6
> |

```

Gambar 4.6

Perintah R `A[B[B>3]]` pada Gambar 4.7 sama saja dengan `A[4]`, yang akan menghasilkan 7. Perintah R `B[B>3]` akan menghasilkan 4 (Perhatikan Gambar 4.8).

```

Console ~/
> A <- c(8,4,5,7,2,3,1,6)
> B <- c(4,3,1,2)
> A[B[B>3]]
[1] 7
> |

```

Gambar 4.7

```

Console ~/
> A <- c(8,4,5,7,2,3,1,6)
> B <- c(4,3,1,2)
> A[B[B>3]]
[1] 7
> B[B>3]
[1] 4
> |

```

Gambar 4.8

Perhatikan Gambar 4.9. Perintah R pada Gambar 4.9 `A[A>=2 & A<7]` berarti menampilkan bilangan dari vektor `A` dengan syarat bilangan tersebut lebih besar atau sama dengan 2 dan lebih kecil dari 7, yang mana bilangan-bilangan tersebut adalah 4, 5, 2, 3, 6, 5, 2, 2, dan 6.

```

Console ~/
> A <- c(8,4,5,7,2,3,1,6,5,2,8,2,9,6)
> A[A>=2 & A<7]
[1] 4 5 2 3 6 5 2 2 6
> table(A[A>=2 & A<7])

 2  3  4  5  6 
 3  1  1  2  2 
> |

```

Gambar 4.9

Namun jika perintah R nya adalah `A[A>=2 | A<7]` akan menampilkan seluruh elemen dari vektor `A` (Gambar 4.10).

```

Console ~/
> A <- c(8,4,5,7,2,3,1,6,5,2,8,2,9,6)
> A[A>=2 & A<7]
[1] 4 5 2 3 6 5 2 2 6
> table(A[A>=2 & A<7])

 2  3  4  5  6 
 3  1  1  2  2 
> A[A>=2 | A<7]
[1] 8 4 5 7 2 3 1 6 5 2 8 2 9 6
> |

```

Gambar 4.10

Perintah R `A[A<2 | A>9]` menampilkan hasil 1. Elemen dari vektor **A** yang lebih kecil dari 2 adalah 1, sementara elemen dari vektor **A** yang lebih besar dari 9 tidak ada. Perintah R `A[A<2 | A>=9]` menampilkan hasil 1 dan 9, sementara perintah R `A[A<=2 | A>=9]` menampilkan hasil 2 (pada indeks ke-5), 1 (pada indeks ke-7), 2 (pada indeks ke-10), 2 (pada indeks ke-12), dan 9 (pada indeks ke-13).

```

Console ~/
> A <- c(8,4,5,7,2,3,1,6,5,2,8,2,9,6)
> A[A>=2 & A<7]
[1] 4 5 2 3 6 5 2 2 6
> table(A[A>=2 & A<7])

 2 3 4 5 6 
3 1 1 2 2 
> A[A>=2 | A<7]
[1] 8 4 5 7 2 3 1 6 5 2 8 2 9 6
> A[A<2 | A>9]
[1] 1
> A[A<2 | A>=9]
[1] 1 9
> A[A<=2 | A>=9]
[1] 2 1 2 2 9
>

```

Gambar 4.11

BAB 5

MATRIKS

5.1 Membuat Matriks dengan Fungsi *matrix()*

Norman Matloff (2009) menyatakan sebagai berikut.

“Matrix is a vector with two additional attributes, the number of rows and number of columns.”

Multidimensional vectors in R are called arrays. A two-dimensional array is also called a matrix, and is eligible for the usual matrix mathematical operations.

*Matrix row and column subscripts begin with 1, so for instance the upper-left corner of the matrix **a** is denoted **a[1,1]**. The internal linear storage of a matrix is in column-major-order, meaning that first all of column 1 is stored, then all of column 2, etc.”*

Berdasarkan uraian di atas dapat ditarik informasi bahwa suatu matriks merupakan suatu vektor dengan dua atribut tambahan (*two additional attributes*), yakni jumlah baris dan jumlah kolom. Vektor multidimensi (*multidimensional vectors*) dalam R disebut *arrays*. Array dua dimensi (*two-dimensional array*) juga disebut matriks. Perhatikan Gambar 5.1.

```
Console ~/
> y <- matrix(c(1,2,3,4), nrow=2, ncol=2)
> y
      [,1] [,2]
[1,]    1    3
[2,]    2    4
>
```

Gambar 5.1

Pada Gambar 5.1 digunakan fungsi *matrix()* untuk membuat matriks. Perintah R **y <- matrix(c(1,2,3,4), nrow=2, ncol=2)** dapat diartikan membuat suatu matriks dengan nama “y”, jumlah baris sebanyak 2, dan jumlah kolom sebanyak 2. Elemen-elemen pada matriks y adalah 1, 3, 2, dan 4. Diketahui:

- ⇒ Nilai 1 menempati posisi baris ke-1 dan kolom ke-1.
- ⇒ Nilai 2 menempati posisi baris ke-2 dan kolom ke-1.
- ⇒ Nilai 3 menempati posisi baris ke-1 dan kolom ke-2.
- ⇒ Nilai 4 menempati posisi baris ke-2 dan kolom ke-2.

Perintah R **y[1,1]** berarti menampilkan nilai pada matriks **y**, pada posisi baris ke-1 dan kolom ke-1, yakni 1. Perintah R **y[1,2]** berarti menampilkan nilai pada matriks **y**, pada posisi baris ke-1 dan kolom ke-2, yakni 3, dan seterusnya. Perhatikan Gambar 5.2.

```
> y <- matrix(c(1,2,3,4), nrow=2, ncol=2)
> y
      [,1] [,2]
[1,]    1    3
[2,]    2    4
> y[1,1]
[1] 1
> y[1,2]
[1] 3
>
```

Gambar 5.2

Pada Gambar 5.3, perintah R **z <- matrix(c(1,2,3,4,5,6), nrow=2)** dapat diartikan membuat suatu matriks bernama **z**, yang jumlah barisnya sebanyak 2. Elemen-elemen pada matriks **z** adalah 1, 2, 3, 4, 5, dan 6. Perhatikan bahwa:

- ⇒ Nilai 1 menempati posisi baris ke-1 dan kolom ke-1.
- ⇒ Nilai 2 menempati posisi baris ke-2 dan kolom ke-1.
- ⇒ Nilai 3 menempati posisi baris ke-1 dan kolom ke-2.
- ⇒ Nilai 4 menempati posisi baris ke-2 dan kolom ke-2.
- ⇒ Nilai 5 menempati posisi baris ke-1 dan kolom ke-3.
- ⇒ Nilai 6 menempati posisi baris ke-2 dan kolom ke-3.

```
Console ~/
> z <- matrix(c(1,2,3,4,5,6), nrow=2)
> z
      [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
>
```

Gambar 5.3

```
Console ~/
> z <- matrix(c(1,2,3,4,5,6), ncol=2)
> z
      [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> |
```

Gambar 5.4

Pada Gambar 5.4, perintah R `z <- matrix(c(1,2,3,4,5,6), ncol=2)` dapat diartikan membentuk suatu matriks bernama **z**, yang jumlah kolomnya sebanyak 2. Elemen-elemen pada matriks **z** adalah 1, 2, 3, 4, 5, dan 6. Perhatikan bahwa:

- ⇒ Nilai 1 menempati posisi baris ke-1 dan kolom ke-1.
- ⇒ Nilai 2 menempati posisi baris ke-2 dan kolom ke-1.
- ⇒ Nilai 3 menempati posisi baris ke-3 dan kolom ke-1.
- ⇒ Nilai 4 menempati posisi baris ke-1 dan kolom ke-2.
- ⇒ Nilai 5 menempati posisi baris ke-2 dan kolom ke-2.
- ⇒ Nilai 6 menempati posisi baris ke-3 dan kolom ke-2.

Namun perhatikan Gambar 5.5 dan Gambar 5.6.

```

Console ~/
> z <- matrix(c(1,2,3,4,5,6), ncol=2) #Membuat matriks bernama z, dengan jumlah kolom sebanyak 2
> z
      [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
>
> z <- matrix(c(1,2,3,4,5,6), ncol=2, byrow=TRUE) #Membuat matriks bernama z, dengan jumlah kolom sebanyak 2
> z
      [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
>

```

Gambar 5.5

```

Console ~/
> z
      [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
> z[1,] #Menampilkan elemen matriks z pada baris ke-1, yakni 1 dan 2
[1] 1 2
> z[2,] #Menampilkan elemen matriks z pada baris ke-2, yakni 3 dan 4
[1] 3 4
> z[3,] #Menampilkan elemen matriks z pada baris ke-3, yakni 5 dan 6
[1] 5 6
> z[,1] #Menampilkan elemen matriks z pada kolom ke-1, yakni 1, 3, dan 5
[1] 1 3 5
> z[,2] #Menampilkan elemen matriks z pada kolom ke-2, yakni 2, 4, dan 6
[1] 2 4 6
> |

```

Gambar 5.6

Berdasarkan Gambar 5.6, pada perintah R `z <- matrix(x(1,2,3,4,5,6), ncol=2, byrow=TRUE)` berarti membuat matriks bernama **z** dengan jumlah kolom sebanyak 2. Perhatikan bahwa:

- ⇒ Nilai 1 menempati posisi baris ke-1 dan kolom ke-1.
- ⇒ Nilai 2 menempati posisi baris ke-1 dan kolom ke-2.
- ⇒ Nilai 3 menempati posisi baris ke-2 dan kolom ke-1.
- ⇒ Nilai 4 menempati posisi baris ke-2 dan kolom ke-2.
- ⇒ Nilai 5 menempati posisi baris ke-3 dan kolom ke-1.
- ⇒ Nilai 6 menempati posisi baris ke-3 dan kolom ke-2.

Pada Gambar 5.6, perintah `Rz[1,]` berarti menampilkan elemen-elemen dari matriks **z** pada baris ke-1, yakni 1 dan 2. Perintah `R z[2,]` berarti menampilkan elemen-elemen dari matriks **z** pada baris ke-2, yakni 3 dan 4. Perintah `Rz[,1]` berarti menampilkan elemen-elemen dari matriks **z** pada kolom ke-1, yakni 1, 3, dan 5.

5.2 Menugaskan Bilangan ke Matriks

Perhatikan Gambar 5.7. Pada Gambar 5.7 perintah `R a <- matrix(nrow=3, ncol=2)` berarti membentuk matriks bernama **a**, dengan jumlah baris sebanyak 3 dan jumlah kolom sebanyak 2. Pada awalnya, elemen dari matriks **a** adalah NA (*not available*).

```

Console ~/
> a <- matrix(nrow=3, ncol=2)
> a
      [,1] [,2]
[1,]    NA    NA
[2,]    NA    NA
[3,]    NA    NA
> a[1,1] <- 1
> a[2,1] <- 2
> a[3,1] <- 3
> a[1,2] <- 4
> a[2,2] <- 5
> a[3,2] <- 6
> a
      [,1] [,2]
[1,]     1     4
[2,]     2     5
[3,]     3     6
> |

```

Gambar 5.7

Perhatikan bahwa:

- ⇒ Perintah `R a[1,1] <- 1` berarti menugaskan bilangan 1 ke matriks **a** pada baris ke-1, kolom ke-1.
- ⇒ Perintah `R a[2,1] <- 2` berarti menugaskan bilangan 2 ke matriks **a** pada baris ke-2, kolom ke-1.

- ⇒ Perintah R **a[3,1] <- 3** berarti menugaskan bilangan 3 ke matriks **a** pada baris ke-3, kolom ke-1.
- ⇒ Perintah R **a[1,2] <- 4** berarti menugaskan bilangan 4 ke matriks **a** pada baris ke-1, kolom ke-2.
- ⇒ Perintah R **a[2,2] <- 5** berarti menugaskan bilangan 5 ke matriks **a** pada baris ke-2, kolom ke-2.
- ⇒ Perintah R **a[3,2] <- 6** berarti menugaskan bilangan 6 ke matriks **a** pada baris ke-3, kolom ke-2.

5.3 Mengganti Elemen Matriks

```

> a
      [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> a[2,2]
[1] 5
> a[2,2] <- 1000 #Menggubah elemen dari matriks a pada baris ke-2, kolom ke-2 deng
an bilangan 1000
> a[2,2]
[1] 1000
> a
      [,1] [,2]
[1,]    1    4
[2,]    2 1000
[3,]    3    6
> |

```

Gambar 5.8

Pada Gambar 5.8, misalkan elemen matriks **a** pada baris ke-2 dan kolom ke-2, yakni nilai 5, ingin diganti menjadi 1000. Perintah R untuk mengganti elemen matriks **a** pada baris ke-2 dan kolom ke-2, yakni nilai 5, menjadi 1000 adalah **a[2,2] <- 1000**.

5.4 Berbagai Cara Menampilkan Elemen Matriks

Gambar 5.9 menampilkan berbagai cara untuk menampilkan elemen matriks. Berdasarkan Gambar 5.9, perintah **RA[,c(1)]** berarti menampilkan elemen-elemen matriks **A** pada kolom ke-1, yakni 1, 2, dan 3. Perintah **RA[,-c(2:4)]** berarti menampilkan elemen-elemen matriks **A**, selain dari pada kolom ke-2 sampai ke-4. Dengan kata lain, berarti menampilkan elemen-elemen matriks **A** pada kolom ke-1, yakni 1, 2, dan 3. Perintah **RA[,c(1,4)]** berarti menampilkan elemen-elemen matriks **A** pada kolom ke-1 dan ke-4. Perintah **RA[,c(2:4)]** berarti menampilkan elemen-elemen matriks **A** pada kolom ke-2 sampai ke-4. Perintah **RA[c(1,3),]** berarti menampilkan elemen-elemen matriks **A** pada baris ke-1 dan ke-3.

```

Console ~/
> A <- matrix(c(1,2,3,4,5,6,7,8,9,10,11,12), nrow=3)
> A
      [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> A[,c(1)] #Menampilkan elemen-elemen matriks A pada kolom ke-1
[1] 1 2 3
> A[,-c(2:4)] #Menampilkan elemen-elemen matriks A, selain dari kolom ke-2 sampai kolom ke-4
[1] 1 2 3
> A[,c(1,2)] #Menampilkan elemen-elemen matriks A pada kolom ke-1 dan ke-2
      [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> A[,c(1,4)] #Menampilkan elemen-elemen matriks A pada kolom ke-1 dan ke-4
      [,1] [,2]
[1,]    1   10
[2,]    2   11
[3,]    3   12
> A[,c(2:4)] #Menampilkan elemen-elemen matriks A dari kolom ke-2 sampai ke-4
      [,1] [,2] [,3]
[1,]    4    7   10
[2,]    5    8   11
[3,]    6    9   12
> A[c(1),] #Menampilkan elemen-elemen matriks A pada baris ke-1
[1] 1 4 7 10
> A[c(1,3),] #Menampilkan elemen-elemen matriks A pada baris ke-1 dan ke-3
      [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    3    6    9   12
> A[c(2:3),] #Menampilkan elemen-elemen matriks A pada baris ke-2 sampai ke-3
      [,1] [,2] [,3] [,4]
[1,]    2    5    8   11
[2,]    3    6    9   12
> A[-c(2:3),] #Menampilkan elemen-elemen matriks A, selain dari baris ke-2 sampai ke-3
[1] 1 4 7 10
>

```

Gambar 5.9

5.5 Menghapus Baris dan Kolom pada Matriks

Perhatikan Gambar 5.10. Pada perintah R `A <- A[-c(2), ]` bertujuan untuk menghapus baris ke-2 dari matriks A. Sementara perintah R `A <- A[, -c(4)]` bertujuan menghapus kolom ke-4 dari matriks A.

```

> A <- matrix(c(1,2,3,4,5,6,7,8,9,10,11,12), nrow=3)
> A
      [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> A <- A[-c(2),] #Menghapus baris ke-2 dari matriks A
> A
      [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    3    6    9   12
> A <- A[, -c(4)] #Menghapus kolom ke-4 dari matriks A
> A
      [,1] [,2] [,3]
[1,]    1    4    7
[2,]    3    6    9
>

```

Gambar 5.10

5.6 Mengganti Elemen Matriks (Bagian 2)

Pada Gambar 5.11 dan Gambar 5.12 diberikan berbagai contoh perintah R untuk mengganti elemen matriks. Berdasarkan Gambar 5.11, diberikan matriks **B**, **C**, dan **D**. Diketahui digunakan fungsi *rbind*(

) untuk membuat matriks **C**, sementara fungsi **cbind()** digunakan untuk membuat matriks **D**. Berdasarkan Gambar 5.11, Perintah **B[,1] <- C[,2]** berarti mengganti elemen-elemen matriks **B** pada kolom ke-1, dengan elemen-elemen matriks **C** pada kolom ke-2. Perintah **B[,1] <- D[,1]** berarti mengganti elemen-elemen matriks **B** pada kolom ke-1, dengan elemen-elemen matriks **D** pada kolom ke-1. Perintah **B[,c(2,3)] <- C** berarti mengganti elemen-elemen matriks **B** pada kolom ke-2 dan ke-3, dengan elemen-elemen pada matriks **C**. Perintah **B[c(1),] <- D[c(2),]** mengganti elemen-elemen matriks **B** pada baris ke-1, dengan elemen-elemen pada matriks **D**, baris ke-2. Namun **memberikan informasi kesalahan**

***Error in B[c(1),] = D[c(2),] :
number of items to replace is not a multiple of replacement length***

Hal ini karena jumlah elemen dari matriks **B** pada baris ke-1 sebanyak 3, sementara jumlah elemen dari matriks **D** pada baris ke-2 sebanyak 2. Selanjutnya, perintah **B[c(1),] <- D[c(2),]** berarti mengganti elemen-elemen matriks **C** pada baris ke-1, dengan elemen-elemen pada matriks **D**, baris ke-2.

```

Console ~/
> B <- matrix(c(1,2,3,4,5,6), nrow=2)
> B
      [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> C <- rbind(c(5,2), c(1,4))
> C
      [,1] [,2]
[1,]    5    2
[2,]    1    4
> D <- cbind(c(5,2), c(1,4))
> D
      [,1] [,2]
[1,]    5    1
[2,]    2    4
> B[,1] <- C[,2] #Mengganti elemen-elemen matriks B pada kolom ke-1, dengan elemen-elemen matriks C pada kolom ke-2
> B
      [,1] [,2] [,3]
[1,]    2    3    5
[2,]    4    4    6
> B[,1] <- D[,1] #Mengganti elemen-elemen matriks B pada kolom ke-1, dengan elemen-elemen matriks D pada kolom ke-1
> B
      [,1] [,2] [,3]
[1,]    5    3    5
[2,]    2    4    6
> B[,c(2,3)] <- C #Mengganti elemen-elemen matriks B pada kolom ke-2 dan ke-3, dengan elemen-elemen pada matriks C
> B
      [,1] [,2] [,3]
[1,]    5    5    2
[2,]    2    1    4

```

Gambar 5.11

```

[1,] 5 3 5
[2,] 2 4 6
> B[,c(2,3)] <- c #Mengganti elemen-elemen matriks B pada kolom ke-2 dan ke-3, dengan elemen-elemen pada matriks c
> B
      [,1] [,2] [,3]
[1,] 5    5    2
[2,] 2    1    4
> B[c(1),] <- D[c(2), ] #Mengganti elemen-elemen matriks B pada baris ke-1, dengan elemen-elemen pada matriks D, baris ke-2
Error in B[c(1), ] <- D[c(2), ] :
  number of items to replace is not a multiple of replacement length
> B
      [,1] [,2] [,3]
[1,] 5    5    2
[2,] 2    1    4
> B[,c(2,3)] <- c #Mengganti elemen-elemen matriks B pada kolom ke-2 dan ke-3, dengan elemen-elemen pada matriks D
> B
      [,1] [,2] [,3]
[1,] 5    5    2
[2,] 2    1    4
> B[c(1),] <- D[c(2),] #Mengganti elemen-elemen matriks B pada baris ke-1, dengan elemen-elemen matriks D, baris ke-2
Error in B[c(1), ] <- D[c(2), ] :
  number of items to replace is not a multiple of replacement length
> B
      [,1] [,2] [,3]
[1,] 5    5    2
[2,] 2    1    4
> C
      [,1] [,2]
[1,] 5    2
[2,] 1    4
> C[c(1),] <- D[c(2),] #Mengganti elemen-elemen matriks C pada baris ke-1, dengan elemen-elemen pada matriks D, baris ke-2
> C
      [,1] [,2]
[1,] 2    4
[2,] 1    4
>

```

Gambar 5.12

5.7 Operasi Matematika pada Matriks (Bagian 1)

Perhatikan Gambar 5.13. Perintah R **B+C** berarti

$$B = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}, \quad C = \begin{bmatrix} 2 & 4 \\ 1 & 1 \end{bmatrix}, \quad B + C = \begin{bmatrix} 1+2 & 3+4 \\ 2+1 & 2+1 \end{bmatrix} = \begin{bmatrix} 3 & 7 \\ 3 & 3 \end{bmatrix}$$

```

Console ~/
> A <- matrix(c(1,2,3,4,2,1), nrow=2)
> B <- matrix(c(1,2,3,2), nrow=2)
> C <- matrix(c(2,1,4,1), nrow=2)
> A
      [,1] [,2] [,3]
[1,] 1    3    2
[2,] 2    4    1
> B
      [,1] [,2]
[1,] 1    3
[2,] 2    2
> C
      [,1] [,2]
[1,] 2    4
[2,] 1    1
> B+C
      [,1] [,2]
[1,] 3    7
[2,] 3    3
>

```

Gambar 5.13

Perintah R **A+B** pada Gambar 5.14 memberi pesan kesalahan dikarenakan matriks A terdiri dari 2 baris dan 3 kolom, sementara matriks B terdiri dari 2 baris dan 2 kolom.


```

Console ~/
> A
      [,1] [,2] [,3]
[1,]    1    3    2
[2,]    2    4    1
> B
      [,1] [,2]
[1,]    1    3
[2,]    2    2
> C
      [,1] [,2]
[1,]    2    4
[2,]    1    1
> A+B
Error in A + B : non-conformable arrays
> |

```

Gambar 5.14

Perintah R **B/C** pada Gambar 5.15 berarti

$$B = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}, \quad C = \begin{bmatrix} 2 & 4 \\ 1 & 1 \end{bmatrix}, \quad B/C = \begin{bmatrix} 1/2 & 3/4 \\ 2/1 & 2/1 \end{bmatrix} = \begin{bmatrix} 0,5 & 0,75 \\ 2 & 2 \end{bmatrix}$$

```

Console ~/
> A
      [,1] [,2] [,3]
[1,]    1    3    2
[2,]    2    4    1
> B
      [,1] [,2]
[1,]    1    3
[2,]    2    2
> C
      [,1] [,2]
[1,]    2    4
[2,]    1    1
> B/C
      [,1] [,2]
[1,] 0.5 0.75
[2,] 2.0 2.00
> |

```

Gambar 5.15

Perintah R **B%%C** pada Gambar 5.16 berarti

$$B = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}, \quad C = \begin{bmatrix} 2 & 4 \\ 1 & 1 \end{bmatrix}, \quad B%%C = \begin{bmatrix} 1%%2 & 3%%4 \\ 2%%2 & 2%%1 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 0 & 0 \end{bmatrix}$$

```

Console ~/
> A
      [,1] [,2] [,3]
[1,]    1    3    2
[2,]    2    4    1
> B
      [,1] [,2]
[1,]    1    3
[2,]    2    2
> C
      [,1] [,2]
[1,]    2    4
[2,]    1    1
> B%%C
      [,1] [,2]
[1,]    1    3
[2,]    0    0
> |

```

Gambar 5.16

Perintah R **B^C** pada Gambar 5.17 berarti

$$B = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}, C = \begin{bmatrix} 2 & 4 \\ 1 & 1 \end{bmatrix}, B^C = \begin{bmatrix} 1^2 & 3^4 \\ 2^1 & 2^1 \end{bmatrix} = \begin{bmatrix} 1 & 81 \\ 2 & 2 \end{bmatrix}$$

```

Console ~/
> A
      [,1] [,2] [,3]
[1,]    1    3    2
[2,]    2    4    1
> B
      [,1] [,2]
[1,]    1    3
[2,]    2    2
> C
      [,1] [,2]
[1,]    2    4
[2,]    1    1
> B^C
      [,1] [,2]
[1,]    1   81
[2,]    2    2
> |

```

Gambar 5.17

Perintah R **B%*%C** pada Gambar 5.18 berarti

$$B = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}, C = \begin{bmatrix} 2 & 4 \\ 1 & 1 \end{bmatrix}, B \% * \% C = \begin{bmatrix} (1 \times 2) + (3 \times 1) & (1 \times 4) + (3 \times 1) \\ (2 \times 2) + (2 \times 1) & (2 \times 4) + (2 \times 1) \end{bmatrix} = \begin{bmatrix} 5 & 7 \\ 6 & 10 \end{bmatrix}$$

```

Console ~/
> A
      [,1] [,2] [,3]
[1,]    1    3    2
[2,]    2    4    1
> B
      [,1] [,2]
[1,]    1    3
[2,]    2    2
> C
      [,1] [,2]
[1,]    2    4
[2,]    1    1
> B%*%C
      [,1] [,2]
[1,]    5    7
[2,]    6   10
> |

```

Gambar 5.18

5.8 Fungsi *rowMeans()*, *colMeans()*, *rowSums()*, dan *colSums()*

Perhatikan Gambar 5.19. Perintah R **rowMeans(A)** berarti

$$\frac{1 + 3 + 2}{3}; \frac{2 + 4 + 1}{3} = 2; 2,333$$

Perintah R **colMeans(A)** berarti

$$\frac{1 + 2}{2}; \frac{3 + 4}{2}; \frac{2 + 1}{2} = 1,5; 3,5; 1,5$$

```

Console ~/
> A
      [,1] [,2] [,3]
[1,]    1    3    2
[2,]    2    4    1
> rowMeans(A)
[1] 2.000000 2.333333
> colMeans(A)
[1] 1.5 3.5 1.5
> rowSums(A)
[1] 6 7
> colSums(A)
[1] 3 7 3
>

```

Gambar 5.19

Perintah R **rowSums(A)** berarti

$$1 + 3 + 2 = 6$$

$$2 + 4 + 1 = 7$$

Perintah R **colSums(A)** berarti

$$1 + 2 = 3$$

$$3 + 4 = 7$$

$$2 + 1 = 3$$

5.9 Transpose Matriks

Gambar 5.20 memperlihatkan penggunaan fungsi *t()* untuk menentukan *tranpose* suatu matriks.

$$A = \begin{bmatrix} 1 & 3 & 2 \\ 2 & 4 & 1 \end{bmatrix}; A^T = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 2 & 1 \end{bmatrix}$$

```

Console ~/
> A
      [,1] [,2] [,3]
[1,]    1    3    2
[2,]    2    4    1
> t(A)
      [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    2    1
> |

```

Gambar 5.20

5.10 Determinan Matriks

Gambar 5.21 memperlihatkan penggunaan fungsi *det()* untuk menentukan determinan suatu matriks.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \det(A) = (4 \times 1) - (2 \times 3) = -2$$

```

Console ~/
> A
      [,1] [,2]
[1,]    1    2
[2,]    3    4
> det(A)
[1] -2
>

```

Gambar 5.21

5.11 Invers Matriks

Gambar 5.22 memperlihatkan penggunaan fungsi *solve()* untuk menentukan *invers* suatu matriks.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, A^{-1} = \frac{1}{(4 \times 1) - (3 \times 2)} \begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix} = \frac{1}{-2} \begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ 1.5 & -0.5 \end{bmatrix}$$

```

Console ~/
> A <- matrix(c(1,2,3,4), ncol=2, byrow=TRUE)
> A
      [,1] [,2]
[1,]    1    2
[2,]    3    4
> solve(A)
      [,1] [,2]
[1,] -2.0  1.0
[2,]  1.5 -0.5
>

```

Gambar 5.22

5.12 Fungsi *eigen()* untuk Menentukan Nilai Eigen dan Vektor Eigen dari Matriks

Matriks persegi (*square matrix*) A dikatakan memiliki nilai eigen λ , dengan vektor eigen-nya yang bersesuaian (*eigenvector*) $x \neq 0$, jika

$$Ax = \lambda x$$

Misalkan diberikan matriks

$$A = \begin{bmatrix} 1 & -5 \\ -5 & 1 \end{bmatrix}$$

Gambar 5.23 memperlihatkan nilai eigen dan vektor eigen dari matriks A. Perhatikan bahwa

$$\begin{bmatrix} 1 & -5 \\ -5 & 1 \end{bmatrix} \begin{bmatrix} -0.7071068 \\ 0.7071068 \end{bmatrix} = 6 \begin{bmatrix} -0.7071068 \\ 0.7071068 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -5 \\ -5 & 1 \end{bmatrix} \begin{bmatrix} -0.7071068 \\ -0.7071068 \end{bmatrix} = -4 \begin{bmatrix} -0.7071068 \\ -0.7071068 \end{bmatrix}$$

Sehingga matriks A memiliki 2 pasang nilai eigen dan vektor eigen, yakni

$$6, \begin{bmatrix} -0.7071068 \\ 0.7071068 \end{bmatrix} \text{ dan } -4, \begin{bmatrix} -0.7071068 \\ -0.7071068 \end{bmatrix}$$

```

Console ~/
> A
      [,1] [,2]
[1,]    1  -5
[2,]  -5    1
> simpan <- eigen(A)
> simpan
$values
[1] 6 -4

$vectors
      [,1]      [,2]
[1,] -0.7071068 -0.7071068
[2,]  0.7071068 -0.7071068

> eigenvalue1 <- simpan$values[1]
> eigenvalue2 <- simpan$values[2]
> eigenvector1 <- simpan$vectors[,1]
> eigenvector2 <- simpan$vectors[,2]
> eigenvalue1
[1] 6
> eigenvalue2
[1] -4
> eigenvector1
[1] -0.7071068  0.7071068
> eigenvector2
[1] -0.7071068 -0.7071068
> |

```

Gambar 5.23

```

> A%%eigenvector1
      [,1]
[1,] -4.242641
[2,]  4.242641
> eigenvalue1*eigenvector1
[1] -4.242641  4.242641
> A%%eigenvector2
      [,1]
[1,]  2.828427
[2,]  2.828427
> eigenvalue2*eigenvector2
[1]  2.828427  2.828427
> |

```

Gambar 5.24

5.13 Eliminasi Gauss-Jordan untuk Menentukan Solusi dari Sistem Persamaan Linear

Misalkan diberikan sistem persamaan linear sebagai berikut.

$$2x + 4y + 6z = 20$$

$$4x + 6y + 2z = 26$$

$$6x - 2y + 4z = 18$$

Gambar 5.25 dan Gambar 5.26 diberikan ilustrasi dalam R terkait eliminasi Gauss-Jordan, untuk memperoleh solusi dari sistem persamaan linear di atas.

```
> A <- matrix(c(2, 4, 6, 20, 4, 6, 2, 26, 6, -2, 4, 18), ncol=4, byrow=TRUE)
> A
      [,1] [,2] [,3] [,4]
[1,]    2    4    6   20
[2,]    4    6    2   26
[3,]    6   -2    4   18
> A[2,] <- A[2,]-2*A[1,]
> A
      [,1] [,2] [,3] [,4]
[1,]    2    4    6   20
[2,]    0   -2  -10  -14
[3,]    6   -2    4   18
> A[3,] <- A[3,]-3*A[1,]
> A
      [,1] [,2] [,3] [,4]
[1,]    2    4    6   20
[2,]    0   -2  -10  -14
[3,]    0  -14  -14  -42
> A[3,] <- A[3,]-7*A[2,]
> A
      [,1] [,2] [,3] [,4]
[1,]    2    4    6   20
[2,]    0   -2  -10  -14
[3,]    0    0   56   56
> A[1,] <- A[1,]+2*A[2,]
> A
      [,1] [,2] [,3] [,4]
[1,]    2    0  -14   -8
[2,]    0   -2  -10  -14
[3,]    0    0   56   56
> A[1,] <- 4*A[1,]+A[3,]
> A
      [,1] [,2] [,3] [,4]
[1,]    8    0    0   24
[2,]    0   -2  -10  -14
[3,]    0    0   56   56
```

Gambar 5.25

```
> A[2,] <- 56*A[2,]+10*A[3,]
> A
      [,1] [,2] [,3] [,4]
[1,]    8    0    0   24
[2,]    0 -112    0 -224
[3,]    0    0   56   56
> A[1,] <- (1/8)*A[1,]
> A
      [,1] [,2] [,3] [,4]
[1,]    1    0    0    3
[2,]    0 -112    0 -224
[3,]    0    0   56   56
> A[2,] <- (1/112)*A[2,]
> A
      [,1] [,2] [,3] [,4]
[1,]    1    0    0    3
[2,]    0  -1    0   -2
[3,]    0    0   56   56
> A[2,] <- -1*A[2,]
> A
      [,1] [,2] [,3] [,4]
[1,]    1    0    0    3
[2,]    0    1    0    2
[3,]    0    0   56   56
> A[3,] <- (1/56)*A[3,]
> A
      [,1] [,2] [,3] [,4]
[1,]    1    0    0    3
[2,]    0    1    0    2
[3,]    0    0    1    1
> |
```

Gambar 5.26

Berdasarkan Gambar 5.26, diperoleh solusi $x = 3$, $y = 2$, dan $z = 1$. Perhatikan bahwa

$$2x + 4y + 6z = 2(3) + 4(2) + 6(1) = 20$$

$$4x + 6y + 2z = 4(3) + 6(2) + 2(1) = 26$$

$$6x - 2y + 4z = 6(3) - 2(2) + 4(1) = 18$$

5.14 Fungsi *rref()* untuk Menentukan Solusi dari Sistem Persamaan Linear

Misalkan diberikan sistem persamaan linear sebagai berikut.

$$2x + 4y + 6z = 20$$

$$4x + 6y + 2z = 26$$

$$6x - 2y + 4z = 18$$

Gambar 5.27 diperlihatkan ilustrasi dalam R penggunaan fungsi *rref()* untuk menentukan solusi dari sistem persamaan di atas. Diketahui solusi dari sistem persamaan linear tersebut adalah $x = 3, y = 2$, dan $z = 1$.

```

Console ~/
> A
      [,1] [,2] [,3] [,4]
[1,]    2    4    6    20
[2,]    4    6    2    26
[3,]    6   -2    4    18
> library(pracma) #Untuk menggunakan fungsi rref()
> rref(A)
      [,1] [,2] [,3] [,4]
[1,]    1    0    0     3
[2,]    0    1    0     2
[3,]    0    0    1     1
> |

```

Gambar 5.27

5.15 Fungsi *LUsplit()* dan *lu.decomposition()* untuk Dekomposisi LU

Misalkan diberikan matriks A sebagai berikut.

$$A = \begin{bmatrix} 4 & 12 & 8 & 4 \\ 1 & 7 & 18 & 9 \\ 2 & 9 & 20 & 20 \\ 3 & 11 & 15 & 14 \end{bmatrix}$$

Gambar 5.28 diberikan ilustrasi dalam R penggunaan fungsi *LUsplit()* untuk memperoleh matriks segitiga bawah L dan matriks segitiga atas U , yang memenuhi $LU = A$.

```

> A
      [,1] [,2] [,3] [,4]
[1,]    4   12    8    4
[2,]    1    7   18    9
[3,]    2    9   20   20
[4,]    3   11   15   14

> library(optR)
> Z <- optR(A, method="LU")
> LUSplit(Z$U)
$U
      [,1] [,2] [,3] [,4]
[1,]    4   12    8    4
[2,]    0    2    9   11
[3,]    0    0   -2  -14
[4,]    0    0    0  -16

$L
      [,1] [,2] [,3] [,4]
[1,] 1.00 0.0 0.00 0
[2,] 0.75 1.0 0.00 0
[3,] 0.25 2.0 1.00 0
[4,] 0.50 1.5 -1.25 1

> simpan$L %>% simpan$U
      [,1] [,2] [,3] [,4]
[1,]    4   12    8    4
[2,]    3   11   15   14
[3,]    1    7   18    9
[4,]    2    9   20   20
>

> A
      [,1] [,2] [,3] [,4]
[1,]    4   12    8    4
[2,]    1    7   18    9
[3,]    2    9   20   20
[4,]    3   11   15   14

> library(matrixcalc)
> lu.decomposition(A)
$L
      [,1] [,2] [,3] [,4]
[1,] 1.00 0.00 0.00 0
[2,] 0.25 1.00 0.00 0
[3,] 0.50 0.75 1.00 0
[4,] 0.75 0.50 0.25 1

$U
      [,1] [,2] [,3] [,4]
[1,]    4   12    8    4
[2,]    0    4   16    8
[3,]    0    0    4   12
[4,]    0    0    0    4

```

Gambar 5.28

5.16 Fungsi *chol()* untuk Dekomposisi Cholesky

Misalkan diberikan matriks simetri A dengan dimensi 3×3 .

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Karena A matriks simetri, maka berlaku

$$A = A^T$$

Sehingga

$$a_{12} = a_{21}$$

$$a_{13} = a_{31}$$

$$a_{23} = a_{32}$$

Sebagai contoh misalkan diberikan matriks simetri A sebagai berikut.

$$A = \begin{bmatrix} 9 & -3 & 6 \\ -3 & 17 & -10 \\ 6 & -10 & 12 \end{bmatrix}$$

Perhatikan bahwa

$$\begin{aligned}A &= A^T \\ a_{12} &= -3 = a_{21} \\ a_{13} &= 6 = a_{31} \\ a_{23} &= -10 = a_{32}\end{aligned}$$

Matriks simetri A dapat dinyatakan sebagai perkalian matriks segitiga bawah L dan matriks segitiga atas U , di mana berlaku

$$L = U^T$$

Sehingga

$$A = LU$$

$$A = U^T U$$

Pada Gambar 5.29 diberikan ilustrasi dalam R penggunaan fungsi ***chol()*** untuk menentukan matriks U^T dan U , sehingga memenuhi $U^T U = A$.

```
Console ~/
> A <- rbind(c(9,-3,6), c(-3,17,-10), c(6,-10,12))
> A
      [,1] [,2] [,3]
[1,]    9   -3    6
[2,]   -3   17  -10
[3,]    6  -10   12
> chol(A)
      [,1] [,2] [,3]
[1,]    3   -1    2
[2,]    0    4   -2
[3,]    0    0    2
> U <- chol(A)
> U_transpose <- t(U)
> U
      [,1] [,2] [,3]
[1,]    3   -1    2
[2,]    0    4   -2
[3,]    0    0    2
> U_transpose
      [,1] [,2] [,3]
[1,]    3    0    0
[2,]   -1    4    0
[3,]    2   -2    2
> U_transpose %% U
      [,1] [,2] [,3]
[1,]    9   -3    6
[2,]   -3   17  -10
[3,]    6  -10   12
> |
```

Gambar 5.29

5.17 Fungsi ***qr()*** untuk Dekomposisi QR

Misalkan diberikan matriks A sebagai berikut.

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Gambar 5.31 diberikan ilustrasi dalam R penggunaan fungsi *qr()* terkait dekomposisi QR, untuk memperoleh matriks Q dan R, sehingga memenuhi $QR = A$. Berdasarkan Gambar 5.31, diperoleh

$$Q = \begin{bmatrix} -0,57735 & 0,81650 & -0,000000000000000055511 \\ -0,57735 & -0,40825 & -0,70711 \\ -0,57735 & -0,40825 & 0,70711 \end{bmatrix}$$

$$R = \begin{bmatrix} -1,7321 & -1,1547 & -0,57735 \\ 0 & -0,8165 & -0,40825 \\ 0 & 0 & 0,70711 \end{bmatrix}$$

$$QR = A$$

$$Q^T Q = I$$

```

> A <- rbind(c(1,0,0), c(1,1,0), c(1,1,1))
> A
      [,1] [,2] [,3]
[1,]    1    0    0
[2,]    1    1    0
[3,]    1    1    1
> y <- qr(A)
> Q <- qr.Q(y)
> R <- qr.R(y)
> y
$qr
      [,1] [,2] [,3]
[1,] -1.73205 -1.15470 -0.57735
[2,]  0.57735 -0.81650 -0.40825
[3,]  0.57735  0.70711  0.70711

$rank
[1] 3

$graux
[1] 1.57735 1.70711 0.70711

$pivot
[1] 1 2 3

attr(,"class")
[1] "qr"
> Q
      [,1] [,2] [,3]
[1,] -0.57735  0.81650 -5.5511e-17
[2,] -0.57735 -0.40825 -7.0711e-01
[3,] -0.57735 -0.40825  7.0711e-01
> R
      [,1] [,2] [,3]
[1,] -1.7321 -1.1547 -0.57735
[2,]  0.0000 -0.8165 -0.40825
[3,]  0.0000  0.0000  0.70711
> Q %*% R
      [,1] [,2] [,3]
[1,]  1 1.1102e-16 1.6259e-17
[2,]  1 1.0000e+00 0.0000e+00
[3,]  1 1.0000e+00 1.0000e+00
> t(Q) %*% Q
      [,1] [,2] [,3]
[1,] 1.0000e+00 -1.6653e-16 0.0000e+00
[2,] -1.6653e-16 1.0000e+00 -1.1102e-16
[3,]  0.0000e+00 -1.1102e-16 1.0000e+00
> |

```

Gambar 5.30

5.18 Memberi Nama Pada Kolom dan Baris dengan Fungsi `colnames()` dan `rownames()`

Gambar 5.31 diberikan ilustrasi dalam R penggunaan fungsi `colnames()` dan `rownames()` untuk memberi nama pada kolom dan baris.

```
Console ~/
> A <- rbind(c(1,2,3), c(4,5,6))
> A
      [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> colnames(A)
NULL
> rownames(A)
NULL
> colnames(A) <- c("Kolom Pertama", "Kolom Kedua", "Kolom Ketiga")
> A
      Kolom Pertama Kolom Kedua Kolom Ketiga
[1,]             1           2           3
[2,]             4           5           6
> rownames(A) <- c("Baris Pertama", "Baris Kedua")
> A
      Kolom Pertama Kolom Kedua Kolom Ketiga
Baris Pertama      1           2           3
Baris Kedua        4           5           6
> rownames(A)
[1] "Baris Pertama" "Baris Kedua"
> A[,1]
Baris Pertama      1
Baris Kedua        4
> A[1,]
Kolom Pertama      1
Kolom Kedua        2
Kolom Ketiga       3

Console ~/
> A["Baris Pertama",]
Kolom Pertama      1
Kolom Kedua        2
Kolom Ketiga       3
> A["Baris Kedua",]
Kolom Pertama      4
Kolom Kedua        5
Kolom Ketiga       6
> A[, "Kolom Kedua"]
Baris Pertama      2
Baris Kedua        5
> A["Baris Pertama", "Kolom Ketiga"]
[1] 3
> A["Baris Kedua", "Kolom Ketiga"]
[1] 6
>

Console ~/
> A
      Kolom Pertama Kolom Kedua Kolom Ketiga
Baris Pertama      1           2           3
Baris Kedua        4           5           6
> rowMeans(A)
Baris Pertama      2
Baris Kedua        5
> colMeans(A)
Kolom Pertama      2.5
Kolom Kedua        3.5
Kolom Ketiga       4.5
> rowSums(A)
Baris Pertama      6
Baris Kedua       15
> colSums(A)
Kolom Pertama      5
Kolom Kedua        7
Kolom Ketiga       9
> |
```

Gambar 5.31

5.19 Fungsi `apply()` untuk Menampilkan Berbagai Ukuran Deskriptif berdasarkan Baris atau Kolom dari Matriks

Gambar 5.32 diberikan ilustrasi dalam R penggunaan `apply()` untuk menampilkan berbagai ukuran deskriptif berdasarkan baris atau kolom dari matriks. Berdasarkan Gambar 5.32, diketahui:

- ⇒ Perintah R `apply(A, 1, sum)` #Jumlah dari masing-masing baris (nilai 1) berarti menjumlahkan bilangan-bilangan dari matriks A berdasarkan baris, yakni $1 + 2 + 3 = 6$ dan $4 + 5 + 6 = 15$. Pada perintah R `apply(A, 1, sum)`, angka “1” tersebut memerintahkan perhitungan berdasarkan baris.
- ⇒ Perintah R `apply(A, 2, sum)` #Jumlah dari masing-masing kolom (nilai 2) berarti menjumlahkan bilangan-bilangan dari matriks A berdasarkan kolom, yakni $1 + 4 = 5$, $2 +$

$5 = 7$ dan $3 + 6 = 9$. Pada perintah R **apply(A, 2, sum)**, angka “2” tersebut memerintahkan perhitungan berdasarkan kolom.

⇒ Perintah R **apply(A, sum)** memberikan pesan kesalahan, begitu juga dengan perintah R **apply(A, 3, sum)**.

```

Console ~/
> A <- rbind(c(1,2,3), c(4,5,6))
> A
      [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> apply(A, 1, sum) #Jumlah dari masing-masing baris (nilai 1)
[1]  6 15
> apply(A, 2, sum) #Jumlah dari masing-masing kolom (nilai 2)
[1]  5 7 9
> apply(A, sum) #Memberi pesan kesalahan
Error in match.fun(FUN) : argument "FUN" is missing, with no default
> apply(A, 3, sum) #Memberi pesan kesalahan
Error in if (d2 == 0L) { : missing value where TRUE/FALSE needed
> apply(A, 1, mean) #Rata-rata dari masing-masing baris (nilai 1)
[1]  2  5
> apply(A, 2, mean) #Rata-rata dari masing-masing kolom (nilai 2)
[1] 2.5 3.5 4.5
> apply(A, 1, max) #Maksimum dari masing-masing baris (nilai 1)
[1]  3  6
> apply(A, 2, max) #Maksimum dari masing-masing kolom (nilai 2)
[1]  4  5  6
> apply(A, 1, min) #Minimum dari masing-masing baris (nilai 1)
[1]  1  4
> apply(A, 2, min) #Minimum dari masing-masing kolom (nilai 2)
[1]  1  2  3
> apply(A, 1, sd) #Standar deviasi dari masing-masing baris (nilai 1)
[1]  1  1
> apply(A, 2, sd) #Standar deviasi dari masing-masing kolom (nilai 2)
[1] 2.1213 2.1213 2.1213
> apply(A, 1, function(x) length(x)) #Jumlah elemen dari masing-masing baris (nilai 1)
[1]  3  3
> apply(A, 2, function(x) length(x)) #Jumlah elemen dari masing-masing kolom (nilai 2)
[1]  2  2  2
> |

```

Gambar 5.32

Perhatikan Gambar 5.33. Perintah **RA <- matrix(nrow = 4, ncol = 5)** berarti membuat matriks bernama A dengan jumlah baris sebanyak 4 dan jumlah kolom sebanyak 5. Selanjutnya menampilkan elemen dari matriks A, yang ternyata seluruhnya adalah “NA” atau “*not available* (tidak tersedia)”. Selanjutnya:

⇒ Perintah R **A[1,1] <- 2** berarti menugaskan bilangan 2 ke matriks A pada posisi baris ke-1 dan kolom ke-1.

⇒ Perintah R **A[1,3] <- 5** berarti menugaskan bilangan 5 ke matriks A pada posisi baris ke-1 dan kolom ke-3, dan seterusnya.

```

Console ~/
> A <- matrix(nrow = 4, ncol = 5)
> A
      [,1] [,2] [,3] [,4] [,5]
[1,]    NA    NA    NA    NA    NA
[2,]    NA    NA    NA    NA    NA
[3,]    NA    NA    NA    NA    NA
[4,]    NA    NA    NA    NA    NA
> A[1,1] <- 2
> A
      [,1] [,2] [,3] [,4] [,5]
[1,]     2    NA    NA    NA    NA
[2,]    NA    NA    NA    NA    NA
[3,]    NA    NA    NA    NA    NA
[4,]    NA    NA    NA    NA    NA
> A[1,3] <- 5
> A
      [,1] [,2] [,3] [,4] [,5]
[1,]     2    NA     5    NA    NA
[2,]    NA    NA    NA    NA    NA
[3,]    NA    NA    NA    NA    NA
[4,]    NA    NA    NA    NA    NA
> A[2,2] <- 3
> A
      [,1] [,2] [,3] [,4] [,5]
[1,]     2    NA     5    NA    NA
[2,]    NA     3    NA    NA    NA
[3,]    NA    NA    NA    NA    NA
[4,]    NA    NA    NA    NA    NA
> A[1,4] <- 6
> A[1,5] <- 8
> A[2,4] <- 1
> A[3,1] <- 2
> A[3,4] <- 1
> A[3,3] <- 2
> A[4,1] <- 5
> A[4,5] <- 2
> A
      [,1] [,2] [,3] [,4] [,5]
[1,]     2    NA     5     6     8
[2,]    NA     3    NA     1    NA
[3,]     2    NA     2     1    NA
[4,]     5    NA    NA    NA     2
> rowSums(A)
[1] NA NA NA NA
> rowSums(A, na.rm=TRUE)
[1] 21  4  5  7
> colSums(A)
[1] NA NA NA NA NA
> colSums(A, na.rm=TRUE)
[1]  9  3  7  8 10
> apply(A, 1, sum) #jumlah dari masing-masing baris (nilai 1)
[1] NA NA NA NA
> apply(A, 1, sum, na.rm=T) #jumlah dari masing-masing baris (nilai 1)
[1] 21  4  5  7
> apply(A, 1, function(x) length(x), na.rm=T) #jumlah elemen dari masing-masing baris (nilai 1)
Error in FUN(newX[, i], ...) : unused argument (na.rm = TRUE)
> apply(A, 1, function(x) length(which(!is.na(x)))) #jumlah elemen dari masing-masing baris (nilai 1)
[1] 4 2 3 2
>
> apply(A, 2, function(x) length(x)) #jumlah elemen dari masing-masing kolom (nilai 2)
[1] 4 4 4 4 4
> apply(A, 2, function(x) length(x), na.rm=T) #jumlah elemen dari masing-masing kolom (nilai 2)
[1] 3 1 2 3 2
>

```

Gambar 5.33

- ⇒ Perintah R **rowSums(A)** bertujuan untuk menjumlahkan bilangan-bilangan berdasarkan baris. Namun hasil dari perintah **rowSums(A)** adalah NA NA NA NA.
- ⇒ Selanjutnya perintah R yang sebelumnya **rowSums(A)** diganti menjadi **rowSums(A, na.rm=TRUE)**. Perintah R **rowSums(A, na.rm=TRUE)** memberikan hasil 21 4 5 7.

$$2 + 5 + 6 + 8 = 21$$

$$3 + 1 = 4$$

$$2 + 2 + 1 = 5$$

$$5 + 2 = 7$$

- ⇒ Perintah R **apply(A, 1, function(x) length(which(!is.na(x))))** memberikan hasil 4 2 3 2. Nilai 4 berarti banyaknya bilangan pada baris pertama sebanyak 4, yakni 2, 5, 6, dan 8. Nilai 2 berarti banyaknya bilangan pada baris kedua sebanyak 2, yakni 3 dan 1.
- ⇒ Pada Gambar 5.34 perintah R **attributes(A)** memberikan informasi dimensi dari matriks A, yakni terdiri dari 4 baris dan 5 kolom.

```

Console ~/
> attributes(A)
$dim
[1] 4 5
> |
  
```

Gambar 5.34

BAB 6

LIST

6.1 Membuat List dengan Fungsi `list()`

Dalam R, fungsi `list()` digunakan untuk membuat suatu *list*. Gambar 6.1 diberikan contoh membuat *list* bernama **daftar**. Perhatikan bahwa pada *list* yang bernama **daftar** terdiri dari 4 anggota *list* (*list members*) atau 4 vektor, yakni **data1**, **data2**, **data3**, dan **data4**. Elemen dari vektor **data1** adalah 3, 4, dan 2. Elemen dari vektor **data2** adalah 4, 4, 7, dan 3, dan seterusnya. Perintah R `mode(daftar)` memperlihatkan bahwa **daftar** berjenis *list*.

```
Console ~/
> daftar <- list(data1 = c(3,4,2), data2 = c(4,4,7,3), data3 = c("Anggur", "Apel"), data4 = c("Ugi", "Egi"))
> daftar
$data1
[1] 3 4 2

$data2
[1] 4 4 7 3

$data3
[1] "Anggur" "Apel"

$data4
[1] "Ugi" "Egi"

> mode(daftar)
[1] "list"
> |
```

Perintah R `mode(daftar)` memperlihatkan bahwa **daftar** berjenis *list*.

Gambar 6.1

6.2 Memperoleh Nama Anggota List (Tags) dengan Fungsi `names()`

Norman Matloff dalam bukunya yang berjudul “*The Art of R Programming*” menyatakan sebagai berikut (2009).

“If the elements in a list do have names, e.g. with **name**, **salary** and **union** for *j* above, these names are called **tags**. The value associated with a tag is indeed called its **value**. You can obtain the tags via `names()`.”

Berdasarkan Gambar 6.1, **data1**, **data2**, **data3**, dan **data4** merupakan nama-nama dari anggota *list*. Pada Gambar 6.2, perintah R `names(daftar)` bertujuan untuk mengetahui nama-nama anggota dari *list* **daftar**.

```
Console ~/
> daftar <- list(data1 = c(3,4,2), data2 = c(4,4,7,3), data3 = c("Anggur", "Apel"), data4 = c("Ugi", "Egi"))
> names(daftar)
[1] "data1" "data2" "data3" "data4"
> |
```

Gambar 6.2

6.3 Mengetahui Jumlah Anggota List dengan Fungsi *length()*

Berdasarkan Gambar 6.2, diketahui jumlah anggota dari *list* **daftar** sebanyak 4. Untuk mengetahui jumlah anggota *list*, dapat digunakan fungsi *length()*. Berdasarkan Gambar 6.3, perintah R **length(daftar)** bertujuan untuk mengetahui banyaknya anggota dari *list* **daftar**, yakni sebanyak 4.

```
Console ~/
> daftar <- list(data1 = c(3,4,2), data2 = c(4,4,7,3), data3 = c("Anggur", "Apel"), data4 = c("Ugi", "Egi"))
> names(daftar)
[1] "data1" "data2" "data3" "data4"
> length(daftar)
[1] 4
> |
```

Gambar 6.3

6.4 Mengakses Anggota List

Gambar 6.4 diberikan ilustrasi untuk mengakses anggota *list* **daftar**. Berdasarkan Gambar 6.4, perintah R **daftar[1]** bertujuan mengakses anggota pertama dari *list* **daftar**, yakni **data1**, serta terlihat bahwa elemen dari **data1**, yakni 3, 4, dan 2. Perintah R **mode(daftar[1])** memberikan informasi bahwa **daftar[1]** berjenis *list*.

```
Console ~/
> daftar <- list(data1 = c(3,4,2), data2 = c(4,4,7,3), data3 = c("Anggur", "Apel"), data4 = c("Ugi", "Egi"))
> daftar[1]
$data1
[1] 3 4 2

> mode(daftar[1])
[1] "list"
> |
```

Gambar 6.4

Pada Gambar 6.5, perintah R **daftar[2]** bertujuan mengakses anggota kedua dari *list* **daftar**, yakni **data2**, serta terlihat bahwa elemen dari **data2**, yakni 4, 4, 7, dan 3. Perintah R **mode(daftar[2])** memberikan informasi bahwa **daftar[2]** berjenis *list*.

```
Console ~/
> daftar <- list(data1 = c(3,4,2), data2 = c(4,4,7,3), data3 = c("Anggur", "Apel"), data4 = c("Ugi", "Egi"))
> daftar[1]
$data1
[1] 3 4 2

> mode(daftar[1])
[1] "list"
> daftar[2]
$data2
[1] 4 4 7 3

> mode(daftar[2])
[1] "list"
> |
```

Gambar 6.5

Mengakses anggota dari *list* juga dapat menggunakan nama dari anggota *list*. Sebagai contoh perintah R **daftar[“data1”]** bertujuan mengakses anggota *list* **daftar** bernama **data1**. Perintah R **daftar[“data2”]** bertujuan mengakses anggota *list* **daftar** bernama **data2**. Perintah R **daftar[c(1,4)]** bertujuan mengakses anggota pertama dan keempat dari *list* **daftar**, yakni **data1** dan **data4**. Perhatikan Gambar 6.6.

```
Console ~/
> daftar["data1"]
$data1
[1] 3 4 2

> daftar["data2"]
$data2
[1] 4 4 7 3

> daftar[c(1,4)]
$data1
[1] 3 4 2

$data4
[1] "Ugi" "Egi"

> daftar[c("data1", "data4")]
$data1
[1] 3 4 2

$data4
[1] "Ugi" "Egi"
```

Gambar 6.6

6.5 Mengakses Elemen dari Anggota List

Pada Gambar 6.7, perhatikan hasil dari perintah R **daftar[1]** dan **daftar[[1]]**. Hasil dari perintah R **daftar[1]** adalah

```
$data1
[1] 3 4 2
```

Sementara hasil dari perintah R **daftar[[1]]** adalah

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

Jenis dari **daftar[1]** adalah *list*, sementara jenis **daftar[[1]]** adalah *numeric* (vektor berjenis numerik). Perintah R **daftar[[1]]** atau **daftar[["data1"]]** atau **daftar\$data1** mengakses elemen dari anggota *list* pertama.

```

Console ~/
> daftar
$data1
[1] 3 4 2

$data2
[1] 4 4 7 3

$data3
[1] "Anggur" "Ape1"

$data4
[1] "Ugi" "Egi"

> mode(daftar)
[1] "list"
> simpan1 <- daftar[1]
> simpan1
$data1
[1] 3 4 2

> mode(simpan1)
[1] "list"
> simpan2 <- daftar[[1]]
> simpan2
[1] 3 4 2
> mode(simpan2)
[1] "numeric"
> daftar[["data2"]]
[1] 4 4 7 3
> mode(daftar[["data2"]])
[1] "numeric"
> |

Console ~/
> daftar[[1]]
[1] 3 4 2
> mode(daftar[[1]])
[1] "numeric"
> daftar[["data1"]]
[1] 3 4 2
> mode(daftar[["data1"]])
[1] "numeric"
> daftar$data1
[1] 3 4 2
> mode(daftar$data1)
[1] "numeric"
> |

```

Gambar 6.7

6.6 Mengganti Nama dari Anggota List

Nama dari anggota *list* dapat diganti. Gambar 6.8 diberikan ilustrasi mengganti nama dari anggota *list* **daftar**. Pada awalnya, nama-nama anggota dari *list* **daftar** adalah **data1**, **data2**, **data3**, dan **data4**. Perintah R **names(daftar)[3] <- c("Buah")** bertujuan mengganti nama dari anggota *list* **daftar** yang ketiga (pada indeks ke-3) dengan nama “**Buah**”. Sementara perintah R **names(daftar)[c(1,4)] <- c("tes 1", "tes 4")** bertujuan untuk mengganti nama dari anggota *list* **daftar** yang pertama dan keempat (pada indeks ke-1 dan indeks ke-4) dengan nama “**tes 1**” dan “**tes 4**”.

```

Console ~/
> daftar
$data1
[1] 3 4 2

$data2
[1] 4 4 7 3

$data3
[1] "Anggur" "Ape1"

$data4
[1] "Ugi" "Egi"

> names(daftar)
[1] "data1" "data2" "data3" "data4"
> names(daftar)[3] <- c("Buah")
> names(daftar)
[1] "data1" "data2" "Buah" "data4"
> names(daftar)[c(1,4)] <- c("tes 1", "tes 4")
> names(daftar)
[1] "tes 1" "data2" "Buah" "tes 4"
> |

```

Gambar 6.8

6.7 Mengganti Elemen dari Anggota List

Di samping nama dari anggota *list* dapat diganti, elemen dari anggota *list* juga dapat diganti. Gambar 6.9 diberikan ilustrasi mengganti elemen dari anggota *list*.

- ⇒ Pertama, diketahui *list* **daftar** memiliki 2 anggota, yakni **data1** dan **data2**. Elemen dari **data1** adalah 2, 6, dan 3, sementara elemen dari **data2** adalah “Apel”, “Mangga”, dan “Melon”.
- ⇒ Selanjutnya elemen kedua dari **data1**, yakni 6, akan diubah menjadi 20. Diketahui nilai 6 terletak pada posisi atau indeks ke-2 pada **data1**. Perintah R untuk mengganti 6 menjadi 20 adalah **daftar\$data1[2] <- 20**.
- ⇒ Elemen pertama dan ketiga dari **data2**, yakni “Apel” dan “Melon”, akan diubah menjadi “Anggur” dan “Jeruk”. Diketahui “Apel” terletak pada posisi atau indeks ke-1 dari **data2**, sementara “Melon” terletak pada posisi atau indeks ke-3 dari **data2**. Perintah R untuk mengganti “Apel” menjadi “Anggur” dan “Melon” menjadi “Jeruk” adalah **daftar\$data2[c(1,3)] <- c("Anggur", "Jeruk")**.

```
Console ~/
> daftar <- list(data1 = c(2,6,3), data2 = c("Apel", "Mangga", "Melon"))
> daftar
$data1
[1] 2 6 3

$data2
[1] "Apel" "Mangga" "Melon"

> daftar$data1
[1] 2 6 3
> daftar$data1[2]
[1] 6
> daftar$data1[2] <- 20 #Diketahui list daftar memiliki 2 anggota, yakni data1
dan data2. Elemen dari data1 adalah 2, 6, dan 3. Elemen ke-2 dari data1, yakni
6, akan diubah menjadi 20.
> daftar
$data1
[1] 2 20 3

$data2
[1] "Apel" "Mangga" "Melon"

> daftar$data2
[1] "Apel" "Mangga" "Melon"
> daftar$data2[3]
[1] "Melon"
> daftar$data2[c(1,3)]
[1] "Apel" "Melon"
> daftar$data2[c(1,3)] <- c("Anggur", "Jeruk")
> daftar
$data2
[1] "Anggur" "Mangga" "Jeruk"
```

Gambar 6.9

6.8 Menghapus Nama dan Memberi Nama dari Anggota List

Nama dari anggota *list* juga dapat dihapus. Gambar 6.10 diberikan ilustrasi dalam R menghapus nama dari anggota *list* **daftar**.

```

Console ~/
> daftar
$data1
[1] 2 20 3

$data2
[1] "Anggur" "Mangga" "Jeruk"

> names(daftar)
[1] "data1" "data2"
> names(daftar) <- NULL
> daftar
[[1]]
[1] 2 20 3

[[2]]
[1] "Anggur" "Mangga" "Jeruk"

> names(daftar)
NULL
> mode(daftar)
[1] "list"

```

Gambar 6.10

Berdasarkan Gambar 6.10, diketahui anggota *list* **daftar** sebanyak 2, dengan nama **data1** dan **data2**. Pada Gambar 6.11, selanjutnya nama dari anggota *list* **daftar** dihapus dengan perintah R **names(daftar) <- NULL**, sehingga nama **data1** menjadi [1] dan nama **data2** menjadi [2]. Kemudian diberi nama kembali, yakni [1] menjadi **data pertama** dan [2] menjadi **data kedua**.

```

> daftar
$data1
[1] 2 20 3

$data2
[1] "Anggur" "Mangga" "Jeruk"

> names(daftar)
[1] "data1" "data2"
> names(daftar) <- NULL
> daftar
[[1]]
[1] 2 20 3

[[2]]
[1] "Anggur" "Mangga" "Jeruk"

> names(daftar)
NULL
> mode(daftar)
[1] "list"
> names(daftar) <- c("data pertama", "data kedua")
> daftar
$`data pertama`
[1] 2 20 3

$`data kedua`
[1] "Anggur" "Mangga" "Jeruk"

> names(daftar)
[1] "data pertama" "data kedua"

```

Gambar 6.11

6.9 Mengubah *List* Menjadi Vektor dengan Fungsi *unlist()*

Gambar 6.12 diberikan ilustrasi mengubah *list* menjadi vektor.

```

> daftar <- list(data1 = c(1,2), data2 = c(3,4), data3 = c(5,6))
> daftar
$data1
[1] 1 2

$data2
[1] 3 4

$data3
[1] 5 6

> mode(daftar)
[1] "list"
> daftar <- unlist(daftar)
> daftar
data11 data12 data21 data22 data31 data32
     1      2      3      4      5      6
> mode(daftar)
[1] "numeric"
> names(daftar) <- NULL
> daftar
[1] 1 2 3 4 5 6
> |

```

Gambar 6.12

Penggunaan fungsi **unlist()** mengembalikan suatu vektor. Berdasarkan Gambar 6.12, pada awalnya **daftar** adalah suatu *list*. Selanjutnya digunakan fungsi **unlist()**, yakni **daftar <- unlist(daftar)**. Hasilnya adalah **daftar** adalah suatu vektor berjenis numerik dengan nilai 1, 2, 3, 4, 5, dan 6.

6.10 Menambah dan Menghapus Anggota dari Suatu List

Anggota dari suatu *list* dapat ditambah atau dihapus. Gambar 6.13 diberikan ilustrasi menambah anggota dari *list* **daftar**. Pada awalnya anggota dari *list* **daftar** sebanyak 2, yakni **data1** dan **data2**. Selanjutnya perintah R **daftar\$data3 <- c("Medan", "Bandung", "Jogja")** bertujuan untuk menambah anggota dari *list* **daftar**, dengan nama anggota **data3**. Sehingga anggota *list* **daftar** sebanyak 3, yakni **data1**, **data2**, dan **data3**. Kemudian anggota *list* **daftar** yang kedua, yakni **data2** dihapus dengan perintah R **daftar <- daftar[-c(2)]**, sehingga anggota *list* **daftar** yang sekarang adalah **data1** dan **data3**.

```

Console ~/
> daftar <- list(data1 = c(1,2,3), data2 = c("Anggur", "Pepaya"))
> daftar
$data1
[1] 1 2 3

$data2
[1] "Anggur" "Pepaya"

> daftar$data3 <- c("Medan", "Bandung", "Jogja")
> daftar
$data1
[1] 1 2 3

$data2
[1] "Anggur" "Pepaya"

$data3
[1] "Medan" "Bandung" "Jogja"

> daftar <- daftar[-c(2)]
> daftar
$data1
[1] 1 2 3

$data3
[1] "Medan" "Bandung" "Jogja"

```

Gambar 6.13

6.11 Menambah dan Menghapus Elemen dari Anggota List

Elemen dari anggota *list* juga dapat ditambah atau dihapus. Pada Gambar 6.14, pada awalnya anggota dari *list* **daftar** adalah **data1** dan **data2**. Elemen dari **data1** adalah 5, 3, dan 2. Perintah R **daftar\$data1[c(4)] <- 21** bertujuan untuk menugaskan bilangan 21 disimpan ke dalam **data1**. Pada “c(4)” menyatakan indeks ke-4 dari **data1**. Sehingga elemen dari **data1** saat ini adalah 5, 3, 2, dan 21. Kemudian elemen dari **data1** akan ditambah lagi dengan bilangan 10 dan 11. Perintah R untuk menambah bilangan 10 dan 11 di **data1** adalah **daftar\$data1[c(5,6)] <- c(10,11)**. Perintah R **daftar\$data1[c(5,6)] <- c(10,11)** dapat juga diartikan menugaskan bilangan 10 dan 11 ke **data1** pada indeks ke-5 dan indeks ke-6. Sehingga elemen dari **data1** saat ini adalah 5, 3, 2, 21, 10, dan 11.

Diketahui elemen dari **data2** adalah “Jakarta” dan “Medan”. Perintah R **daftar\$data2[c(3,4)] <- c(“Bogor”, “Jogja”)** bertujuan menambah atau menugaskan “Bogor” dan “Jogja” ke **data2** pada indeks ke-3 dan indeks ke-4. Sehingga elemen dari **data2** adalah “Jakarta”, “Medan”, “Bogor”, dan “Jogja”. Selanjutnya misalkan ingin dihapus elemen dari **data1** pada indeks ke-5 dan indeks ke-6, yakni bilangan 10 dan 11. Perintah R untuk menghapus elemen dari **data1** pada indeks ke-5 dan indeks ke-6 adalah **daftar\$data1 <- daftar\$data1[-c(5,6)]**. Sehingga elemen dari **data1** saat ini menjadi 5, 3, 2, dan 21. Perintah R **daftar\$data1 <- daftar\$data1[-c(1,3)]** bertujuan untuk menghapus elemen dari **data1** pada indeks ke-1 dan indeks ke-3. Sehingga elemen dari **data1** saat ini menjadi 3 dan 21. Perintah R **daftar\$data2 <- daftar\$data2[-c(2,3)]** bertujuan untuk menghapus elemen dari **data2** pada indeks ke-2 dan indeks ke-3. Sehingga elemen dari **data2** saat ini menjadi “Jakarta” dan “Jogja”.

The image displays two sequential screenshots of an R console window, illustrating the step-by-step manipulation of a list object named **daftar**.

Left Screenshot (Initial State):

```
> daftar <- list(data1 = c(5,3,2), data2 = c("Jakarta","Medan"))
> daftar
$data1
[1] 5 3 2

$data2
[1] "Jakarta" "Medan"
```

Right Screenshot (Final State after all operations):

```
$data2
[1] "Jakarta" "Medan"

> daftar$data2[c(3,4)] <- c("Bogor","Jogja")
> daftar
$data1
[1] 5 3 2 21 10 11

$data2
[1] "Jakarta" "Medan" "Bogor" "Jogja"
```

The sequence of operations performed in the console is as follows:

- Initial creation of **daftar** with **data1** (5, 3, 2) and **data2** ("Jakarta", "Medan").
- Adding 21 to **data1** at index 4: **daftar\$data1[c(4)] <- 21**.
- Adding 10 and 11 to **data1** at indices 5 and 6: **daftar\$data1[c(5,6)] <- c(10,11)**.
- Removing elements from **data1** at indices 5 and 6: **daftar\$data1 <- daftar\$data1[-c(5,6)]**.
- Removing elements from **data1** at indices 1 and 3: **daftar\$data1 <- daftar\$data1[-c(1,3)]**.
- Removing elements from **data2** at indices 2 and 3: **daftar\$data2 <- daftar\$data2[-c(2,3)]**.

Gambar 6.14

6.12 Fungsi Regresi Linear *lm()* Mengembalikan *List*

Gambar 6.15 diberikan ilustrasi penggunaan fungsi *lm()* untuk melakukan metode statistika regresi linear, dengan Y sebagai variabel tak bebas dan X sebagai variabel bebas. Fungsi *lm()* mengembalikan *list*. Pada perintah R **regresi <- lm(Y ~ X)**, hasil dari **lm(Y ~ X)** ditugaskan atau disimpan ke **regresi**. Hasil dari perintah R **mode(regresi)** memperlihatkan bahwa **regresi** berjenis *list*. Perintah R **names(regresi)** memperlihatkan seluruh anggota dari *list regresi*, yakni “*coefficients*”, “*residuals*”, “*effects*”, dan sebagainya. Perintah R **regresi\$coefficients** menampilkan elemen dari *coefficients*, yakni berupa nilai *intercept* (-0,333333) dan *slope* (1,1428571).

```
Console ~/
> X <- c(1,2,3,4,5,6)
> Y <- c(2,1,3,4,4,8)
> regresi <- lm(Y~X)
> summary(regresi)

Call:
lm(formula = Y ~ X)

Residuals:
    1      2      3      4      5      6 
1.19048 -0.95238 -0.09524 -0.23810 -1.38095  1.47619 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.33333    1.1846   -0.281   0.7924
X              1.1429     0.3042    3.757   0.0198 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.272 on 4 degrees of freedom
Multiple R-squared:  0.7792,    Adjusted R-squared:  0.724 
F-statistic: 14.12 on 1 and 4 DF,  p-value: 0.01982

> mode(regresi)
[1] "list"
> names(regresi)
[1] "coefficients" "residuals"    "effects"      "rank"         "fitted.values" "assign"
[7] "qr"           "df.residual"  "xlevels"      "call"         "terms"        "model"
> regresi$coefficients
(Intercept)      X 
-0.3333333    1.1428571 
> regresi$residuals
    1      2      3      4      5      6 
1.1904762 -0.9523810 -0.0952381 -0.2380952 -1.3809524  1.4761905 
> regresi$call
lm(formula = Y ~ X)
```

Gambar 6.15

BAB 7

DATA FRAME

7.1 Antara Data Frame, List, dan Matriks

Gambar 7.1 diberikan contoh *data frame*, *list*, dan matriks. Perhatikan bahwa **contoh_dataframe** adalah *data frame*, **contoh_list** adalah *list*, dan **contoh_matriks** adalah matriks.

⇒ Matriks dan *data frame* sama-sama memiliki struktur baris dan kolom, sementara *list* tidak. Perhatikan bahwa perintah R **contoh_dataframe[1,2]** akan menampilkan data pada *data frame* **contoh_dataframe** pada baris ke-1 dan kolom ke-2, yakni 27. Perintah R **contoh_matriks[3,1]** akan menampilkan data pada matriks **contoh_matriks** pada baris ke-3 dan kolom ke-1, yakni 5. Perhatikan bahwa perintah R **contoh_list[1,2]** memberikan pesan kesalahan.

```
Console ~/
> contoh_dataframe <- data.frame>Nama=c("Ugi","Egi"), Usia=c(27,23))
> contoh_list <- list>Nama=c("Ugi","Egi","Sinta"), Usia=c(27,25,24,29,28))
> contoh_matriks <- matrix(c(1,2,3,4,5,6), ncol = 2, byrow = T)
> contoh_dataframe
  Nama Usia
1  Ugi   27
2  Egi   23
> contoh_dataframe[1,2]
[1] 27
> contoh_dataframe[2,1]
[1] Egi
Levels: Egi Ugi
> contoh_dataframe[2,2]
[1] 23
> contoh_list
$Nama
[1] "Ugi" "Egi" "Sinta"

$Usia
[1] 27 25 24 29 28

> contoh_list[2]
$Usia
[1] 27 25 24 29 28

> contoh_list[1,2]
Error in contoh_list[1, 2] : incorrect number of dimensions
> contoh_matriks
      [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
> contoh_matriks[3,1]
[1] 5
```

Gambar 7.1

⇒ Pada *data frame* **contoh_dataframe**, terdapat dua vektor atau kolom, yakni **Nama** dan **Usia**. **Nama** berjenis *character*, sementara **Usia** berjenis numerik. Pada *list* **contoh_list** juga terdapat dua vektor atau anggota (*list member*), yakni **Nama** dan **Usia**, yang masing-masing

berjenis *character* dan *numeric*. Pada *data frame* dan *list* sama-sama dapat memiliki beberapa vektor yang berbeda jenis, seperti yang telah diperlihatkan. Namun pada *data frame*, panjang dari tiap-tiap vektor atau kolom harus sama, sementara pada *list* dapat berbeda.

⇒ Pada *data frame*, panjang dari masing-masing vektor atau kolom harus sama, begitu juga pada matriks.

Norman Matloff dalam bukunya yang berjudul “*The Art of R Programming*” menyatakan sebagai berikut (2009).

“On an intuitive level, a data frame is like a matrix, with a rows-and-column structure. However, it differs from a matrix in that each column may have a different mode. For instance, one column may be numbers and another column might be character strings.”

On a technical level, a data frame is a list of equal-length vectors. Each column is one element of the list.”

7.2 Membuat Data Frame dengan Fungsi `data.frame()`

Dalam R, fungsi `data.frame()` digunakan untuk membuat suatu *data frame*. Gambar 7.1 diberikan contoh membuat *data frame* bernama **tabel**. Perhatikan bahwa pada *data frame* yang bernama **tabel** terdiri dari 3 vektor atau kolom, yakni **Nama**, **Jenis_Kelamin**, dan **Usia**. Masing-masing vektor memiliki panjang (*length*) yang sama, yakni 5. Pada *data frame*, panjang dari masing-masing kolom atau vektor harus sama.

```
Console ~/
> Nama <- c("Ugi", "Rina", "Andi", "Siti", "Intan")
> Jenis_Kelamin <- c("Laki-Laki", "Perempuan", "Laki-Laki", "Perempuan", "Perempuan")
> Usia <- c(27, 25, 24, 27, 22)
> tabel <- data.frame(Nama, Jenis_Kelamin, Usia)
> tabel
  Nama Jenis_Kelamin Usia
1  Ugi      Laki-Laki  27
2 Rina    Perempuan  25
3  Andi      Laki-Laki  24
4  Siti    Perempuan  27
5 Intan    Perempuan  22
> |
```

Gambar 7.2

Pada Gambar 7.3, bermaksud juga untuk membuat *data frame* bernama **tabel2**, namun memberikan pesan kesalahan. Hal ini karena panjang dari vektor **Nama2**, **Jenis_Kelamin2**, dan **Usia** berbeda-beda. Panjang dari vektor **Nama2** adalah 3, panjang dari vektor **Jenis_Kelamin2** adalah 2, dan panjang dari vektor **Usia** adalah 4.

```

> Nama2 <- c("Fitri", "Iman", "Zeri")
> Jenis_Kelamin2 <- c("Perempuan", "Laki-Laki")
> Usia <- c(24, 21, 32, 25)
> tabel2 <- data.frame(Nama2, Jenis_Kelamin2, Usia)
Error in data.frame(Nama2, Jenis_Kelamin2, Usia) :
  arguments imply differing number of rows: 3, 2, 4
> |

```

Gambar 7.3

Berdasarkan Gambar 7.2, *data frame* **tabel** memiliki tiga kolom, kolom pertama dengan nama **Nama**, kolom kedua dengan nama **Jenis_Kelamin**, dan kolom ketiga dengan nama **Usia**.

7.3Memperoleh Nama-Nama Kolom dari *Data Frame* dengan Fungsi *names()* dan *colnames()*

Diketahui sebelumnya, *data frame* **tabel** memiliki tiga kolom, dengan nama **Nama**, **Jenis_Kelamin**, dan **Usia**. Fungsi *names()* dapat digunakan untuk memperoleh nama-nama kolom dari *data frame*. Pada Gambar 7.4, perintah R **names(tabel)** dan **colnames(tabel)** menampilkan nama-nama kolom atau vektor dari *data frame* **tabel**, yakni **Nama**, **Jenis_Kelamin**, dan **Usia**.

```

Console ~/
> Nama <- c("Ugi", "Rina", "Andi", "Siti", "Intan")
> Jenis_Kelamin <- c("Laki-Laki", "Perempuan", "Laki-Laki", "Perempuan", "Perempuan")
> Usia <- c(27, 25, 24, 27, 22)
>
> tabel <- data.frame(Nama, Jenis_Kelamin, Usia)
> tabel
  Nama Jenis_Kelamin Usia
1  Ugi    Laki-Laki   27
2  Rina   Perempuan   25
3  Andi    Laki-Laki   24
4  Siti   Perempuan   27
5 Intan   Perempuan   22
> names(tabel)
[1] "Nama"          "Jenis_Kelamin" "Usia"
> colnames(tabel)
[1] "Nama"          "Jenis_Kelamin" "Usia"
> |

```

Gambar 7.4

7.4Mengetahui Jumlah Kolom dari Suatu *Data Frame* Fungsi *length()*

Berdasarkan Gambar 7.4, diketahui jumlah kolom dari *data frame* **tabel** sebanyak 3. Untuk mengetahui jumlah kolom dari suatu *data frame*, dapat digunakan fungsi *length()*. Berdasarkan Gambar 7.5, perintah R **length(tabel)** bertujuan untuk mengetahui banyaknya kolom atau vektor dari *data frame* **tabel**, yakni sebanyak 3.

```

Console ~/
> tabel
  Nama Jenis_Kelamin Usia
1  Ugi    Laki-Laki   27
2  Rina   Perempuan   25
3  Andi    Laki-Laki   24
4  Siti   Perempuan   27
5 Intan   Perempuan   22
> length(tabel)
[1] 3
> |

```

Gambar 7.5

7.5 Mengakses Kolom pada Data Frame

Gambar 7.6 diberikan ilustrasi untuk mengakses atau menampilkan kolom pada *data frame* **tabel**. Berdasarkan Gambar 7.6, perintah R **tabel[3]** bertujuan mengakses atau menampilkan kolom atau vektor ketiga dari *data frame* **tabel**, yakni **Usia**, serta terlihat bahwa elemen dari **Usia**, yakni 27, 25, 24, 27, dan 22. Perintah R **mode(tabel[3])** memberikan informasi bahwa **tabel[3]** berjenis *list*. Begitu juga perintah R **tabel["Usia"]** memberikan hasil yang sama dengan perintah R **tabel[3]**.

```
Console ~/
> tabel
  Nama Jenis_Kelamin Usia
1  Ugi      Laki-Laki  27
2  Rina    Perempuan  25
3  Andi      Laki-Laki  24
4  Siti    Perempuan  27
5  Intan    Perempuan  22
> mode(tabel)
[1] "list"
> tabel[3]
  Usia
1  27
2  25
3  24
4  27
5  22
> mode(tabel[3])
[1] "list"
> tabel["Usia"]
  Usia
1  27
2  25
3  24
4  27
5  22
> mode(tabel["Usia"])
[1] "list"
```

Gambar 7.6

Pada Gambar 7.7, perintah R **tabel[2]** bertujuan mengakses atau menampilkan kolom kedua dari *data frame* **tabel**, yakni **Jenis_Kelamin**, serta terlihat bahwa elemen dari **Jenis_Kelamin**, yakni “Laki-Laki”, “Perempuan”, “Laki-Laki”, “Perempuan”, dan “Perempuan”. Perintah R **mode(tabel[2])** memberikan informasi bahwa **tabel[2]** berjenis *list*. Perintah R **tabel["Jenis_Kelamin"]** memberikan hasil yang sama dengan perintah R **tabel[2]**. Sementara pada Gambar 7.8, perintah R **tabel[c(1,3)]** bertujuan mengakses atau menampilkan kolom pertama dan ketiga dari *data frame* **tabel**, yakni **Nama** dan **Usia**.

```

Console ~/
> tabel[2]
  Jenis_Kelamin
1     Laki-Laki
2     Perempuan
3     Laki-Laki
4     Perempuan
5     Perempuan
> mode(tabel[2])
[1] "list"
> tabel["Jenis_kelamin"]
  Jenis_Kelamin
1     Laki-Laki
2     Perempuan
3     Laki-Laki
4     Perempuan
5     Perempuan
> mode(tabel["Jenis_kelamin"])
[1] "list"
> |

```

Gambar 7.7

```

Console ~/
> tabel[c(1,3)]
  Nama  Usia
1   Ugi   27
2   Rina  25
3   Andi  24
4   Siti  27
5  Intan  22
> |

```

Gambar 7.8

7.6 Mengakses Baris pada Data Frame

Baris dari suatu *data frame* juga dapat diakses. Perhatikan Gambar 7.9. Berdasarkan Gambar 7.9, perintah R **tabel[1,]** bertujuan untuk mengakses atau menampilkan data pada baris pertama, dari *data frame* **tabel**. Sementara perintah R **tabel[5,]** bertujuan untuk mengakses atau menampilkan data pada baris kelima, dari *data frame* **tabel**. Perintah R **tabel[c(2,4),]** bertujuan untuk mengakses atau menampilkan data pada baris kedua dan keempat, dari *data frame* **tabel**.

```

Console ~/
> tabel
  Nama Jenis_Kelamin Usia
1   Ugi     Laki-Laki  27
2   Rina   Perempuan  25
3   Andi     Laki-Laki  24
4   Siti   Perempuan  27
5  Intan   Perempuan  22
> tabel[1,]
  Nama Jenis_Kelamin Usia
1   Ugi     Laki-Laki  27
> tabel[5,]
  Nama Jenis_Kelamin Usia
5  Intan   Perempuan  22
> tabel[c(2,4),]
  Nama Jenis_Kelamin Usia
2   Rina   Perempuan  25
4   Siti   Perempuan  27
>

```

Gambar 7.9

7.7 Mengakses Elemen dari Kolom pada Data Frame

Pada Gambar 7.10, perhatikan hasil dari perintah R `tabel[,3]` dan `tabel[3]`. Hasil dari perintah R `tabel[,3]` adalah

```
[1] 27 25 24 27 22
```

Sementara hasil dari perintah R `tabel[3]` adalah

	Usia
1	27
2	25
3	24
4	27
5	22

Jenis dari `tabel[,3]` adalah *numeric*, sementara jenis `tabel[3]` adalah *list*. Perintah R `tabel[,3]` bertujuan untuk mengakses elemen (data) dari kolom ketiga dari *data frame* `tabel`. Perintah R `tabel[3]` memberikan hasil yang sama dengan perintah R `tabel$Usia` atau `tabel[[3]]`.

```
Console ~/ / 
> tabel[,3]
[1] 27 25 24 27 22
> tabel$usia
[1] 27 25 24 27 22
> tabel[[3]]
[1] 27 25 24 27 22
> |
```

Gambar 7.10

7.8 Mengganti Nama Kolom dari Data Frame dengan Fungsi `names()` dan `colnames()`

Nama kolom atau vektor dari *data frame* dapat diganti. Perhatikan Gambar 7.11. Pada Gambar 7.11, nama kolom **Usia** diganti menjadi **Usia Responden**. Perintah R untuk mengganti nama kolom **Usia** menjadi **Usia Responden** adalah `names(tabel)[3] <- "Usia Responden"`.

```
Console ~/ / 
> tabel
  Nama Jenis_kelamin Usia
1  Ugi      Laki-Laki  27
2  Rina    Perempuan  25
3  Andi      Laki-Laki  24
4  Siti    Perempuan  27
5  Intan    Perempuan  22
> names(tabel)[3] <- "usia Responden"
> tabel
  Nama Jenis_kelamin usia Responden
1  Ugi      Laki-Laki      27
2  Rina    Perempuan      25
3  Andi      Laki-Laki      24
4  Siti    Perempuan      27
5  Intan    Perempuan      22
> |
```

Gambar 7.11

Pada Gambar 7.12, perintah R `names(tabel)[c(1,2)] <- c("Nama Responden", "Jenis Kelamin Responden")` bertujuan mengganti nama **Nama** menjadi **Nama Responden** dan mengganti nama **Jenis_Kelamin** menjadi **Jenis Kelamin Responden**.

```

Console ~/
> tabel
  Nama Jenis_Kelamin Usia
1  Ugi      Laki-Laki  27
2  Rina    Perempuan  25
3  Andi      Laki-Laki  24
4  Siti    Perempuan  27
5  Intan    Perempuan  22
> names(tabel)[3] <- "Usia Responden"
> tabel
  Nama Jenis_Kelamin Usia Responden
1  Ugi      Laki-Laki      27
2  Rina    Perempuan      25
3  Andi      Laki-Laki      24
4  Siti    Perempuan      27
5  Intan    Perempuan      22
> names(tabel)[c(1,2)] <- c("Nama Responden", "Jenis Kelamin Responden")
> tabel
  Nama Responden Jenis Kelamin Responden Usia Responden
1      Ugi      Laki-Laki      27
2      Rina    Perempuan      25
3      Andi      Laki-Laki      24
4      Siti    Perempuan      27
5      Intan    Perempuan      22

```

Gambar 7.12

Pada Gambar 7.13, digunakan fungsi `colnames()` untuk mengganti nama kolom.

```

Console ~/
> tabel
  Nama Responden Jenis Kelamin Responden Usia Responden
1      Ugi      Laki-Laki      27
2      Rina    Perempuan      25
3      Andi      Laki-Laki      24
4      Siti    Perempuan      27
5      Intan    Perempuan      22
> colnames(tabel)[1] <- "Nama"
> tabel
  Nama Jenis Kelamin Responden Usia Responden
1  Ugi      Laki-Laki      27
2  Rina      Perempuan      25
3  Andi      Laki-Laki      24
4  Siti      Perempuan      27
5  Intan      Perempuan      22
> colnames(tabel)[c(2:3)] <- c("Jenis", "Usia")
> tabel
  Nama      Jenis Usia
1  Ugi Laki-Laki  27
2  Rina Perempuan  25
3  Andi Laki-Laki  24
4  Siti Perempuan  27
5  Intan Perempuan  22
>

```

Gambar 7.13

7.9 Mengganti Data dari Data Frame

Data atau elemen pada *data frame* dapat diganti. Pada Gambar 7.14, diketahui usia Siti adalah 27. Nilai 27 berada pada baris ke-4 dan kolom ke-3. Nilai 27 kemudian diganti menjadi 30. Perintah R untuk mengganti usia 27 menjadi 30 adalah `tabel[4,3] <- 30`.

```

Console ~/
> tabel
  Nama      Jenis Usia
1  Ugi  Laki-Laki  27
2  Rina Perempuan  25
3  Andi  Laki-Laki  24
4  Siti  Perempuan  27
5  Intan Perempuan  22
> tabel[4,3]
[1] 27
> tabel[4,3] <- 30
> tabel[4,3]
[1] 30
> tabel
  Nama      Jenis Usia
1  Ugi  Laki-Laki  27
2  Rina Perempuan  25
3  Andi  Laki-Laki  24
4  Siti  Perempuan  30
5  Intan Perempuan  22
> |

```

Gambar 7.14

Pada Gambar 7.15, nama Intan diganti menjadi Andi, begitu juga dengan jenis kelaminnya menjadi laki-laki.

```

Console ~/
> tabel
  Nama      Jenis Usia
1  Ugi  Laki-Laki  27
2  Rina Perempuan  25
3  Andi  Laki-Laki  24
4  Siti  Perempuan  30
5  Intan Perempuan  22
> tabel[5,1] <- "Andi"
> tabel[5,2] <- "Laki-Laki"
> tabel
  Nama      Jenis Usia
1  Ugi  Laki-Laki  27
2  Rina Perempuan  25
3  Andi  Laki-Laki  24
4  Siti  Perempuan  30
5  Andi  Laki-Laki  22
> |

```

Gambar 7.15

7.10 Menghapus Kolom pada Data Frame

Kolom pada *data frame* dapat dihapus. Pada Gambar 7.16, kolom **Jenis** dihapus. Kolom **Jenis** berada pada indeks ke-2. Perintah R untuk menghapus kolom **Jenis** adalah **tabel <- tabel[-c(2)]**.

```

Console ~/
> tabel
  Nama      Jenis Usia
1  Ugi  Laki-Laki  27
2  Rina Perempuan  25
3  Andi  Laki-Laki  24
4  Siti  Perempuan  30
5  Andi  Laki-Laki  22
> tabel <- tabel[-c(2)]
> tabel
  Nama Usia
1  Ugi  27
2  Rina  25
3  Andi  24
4  Siti  30
5  Andi  22
> |

```

Gambar 7.16

7.11 Menghapus Data pada Data Frame

Data pada *data frame* dapat dihapus. Pada Gambar 7.17, data pada baris ke-4, yakni Siti dihapus. Perintah R untuk menghapus data pada baris ke-4 adalah `tabel <- tabel[-4,]`. Selanjutnya data Andi dengan usia 24 tahun dan Andi dengan usia 22 tahun dihapus dengan perintah R `tabel <- tabel[-c(3,4),]`.

```
Console ~/
> tabel
  Nama Usia
1  Ugi    27
2  Rina   25
3  Andi   24
4  Siti   30
5  Andi   22
> tabel <- tabel[-4,]
> tabel
  Nama Usia
1  Ugi    27
2  Rina   25
3  Andi   24
5  Andi   22
> tabel <- tabel[-c(3,4),]
> tabel
  Nama Usia
1  Ugi    27
2  Rina   25
>
```

Gambar 7.17

7.12 Menambah Kolom pada Data Frame

Selain dihapus, kolom pada *data frame* dapat ditambah. Pada Gambar 7.18, perintah R

```
tabel[3] <-c("S1","S2")
names(tabel)[3] <- "Pendidikan"
```

bertujuan untuk menambah kolom dengan nama **Pendidikan**.

```
Console ~/
> tabel
  Nama Usia
1  Ugi    27
2  Rina   25
> tabel[3] <-c("S1","S2")
> tabel
  Nama Usia V3
1  Ugi    27 S1
2  Rina   25 S2
> names(tabel)[3] <- "Pendidikan"
> tabel
  Nama Usia Pendidikan
1  Ugi    27         S1
2  Rina   25         S2
> tabel["Pendidikan"]
  Pendidikan
1         S1
2         S2
> |
```

Gambar 7.18

7.13 Menambah Data pada Data Frame

Gambar 7.19 memperlihatkan penambahan data pada *data frame* **tabel**. Pada Gambar 7.19, perintah R **tabel <- rbind(tabel, data.frame("Nama"="Intan", "Usia"=28))** bertujuan untuk menambah data dengan nama Intan, berusia 28 tahun.

```
Console ~/ /
> tabel
  Nama Usia
1  Ugi   23
2  Egi   29
3  Sari  32
> tabel <- rbind(tabel, data.frame("Nama"="Intan", "Usia"=28))
> tabel
  Nama Usia
1  Ugi   23
2  Egi   29
3  Sari  32
4 Intan  28
> |
```

Gambar 7.19

Pada Gambar 7.20, perintah R **tabel2 <- rbind(tabel2, data.frame("Nama"="Prana Ugi", "Jenis_Kelamin"="Laki-Laki", "Usia"=27, "Pendidikan"="S3", "Berat"=56))** bertujuan untuk menambah data pada *data frame* **tabel2** dengan nama Prana Ugi, berjenis kelamin laki-laki, usia 27 tahun, pendidikan S3, dan berat 56.

```
Console ~/ /
> Nama <- c("Andi", "Fitri", "Ali", "Sukri")
> Jenis_Kelamin <- c("Laki-Laki", "Perempuan", "Laki-Laki", "Laki-Laki")
> Usia <- c(27, 25, 32, 30)
> Pendidikan <- c("S1", "S2", "S3", "S1")
> Berat <- c(57.32, 59.24, 65.34, 49.53)
> tabel2 <- data.frame(Nama, Jenis_Kelamin, Usia, Pendidikan, Berat)
> tabel2
  Nama Jenis_Kelamin Usia Pendidikan Berat
1  Andi   Laki-Laki   27         S1  57.32
2 Fitri  Perempuan   25         S2  59.24
3  Ali   Laki-Laki   32         S3  65.34
4 Sukri  Laki-Laki   30         S1  49.53
> tabel2 <- rbind(tabel2, data.frame("Nama"="Prana Ugi", "Jenis_Kelamin"="Laki-Laki", "Usia"=27, "Pendidikan"="S3",
"Berat"=56))
> tabel2
  Nama Jenis_Kelamin Usia Pendidikan Berat
1  Andi   Laki-Laki   27         S1  57.32
2 Fitri  Perempuan   25         S2  59.24
3  Ali   Laki-Laki   32         S3  65.34
4 Sukri  Laki-Laki   30         S1  49.53
5 Prana Ugi Laki-Laki   27         S3  56.00
> |
```

Gambar 7.20

7.14 Menggabungkan Data Frame dengan Fungsi *rbind()* berdasarkan Baris

Pada Gambar 7.21, diperlihatkan dua *data frame*, yakni **tabel** dan **tabel2**. Jumlah kolom atau vektor pada *data frame* **tabel** dan **tabel2** adalah 2, yakni **Nama** dan **Usia**. Selanjutnya data pada *data frame* **tabel** dan **tabel2** digabung berdasarkan baris, dengan perintah R **rbind(tabel, tabel2)**. Hasilnya diperlihatkan pada Gambar 7.21.

```

Console ~/
> tabel
  Nama Usia
1  Ugi   23
2  Egi   29
3  Sari  32
4  Intan 28
> tabel2
  Nama Usia
1  Budi  27
2  Sinta 27
3  Rani  25
4  Andi  31
> tabel_gabungan <- rbind(tabel, tabel2)
> tabel_gabungan
  Nama Usia
1  Ugi   23
2  Egi   29
3  Sari  32
4  Intan 28
5  Budi  27
6  Sinta 27
7  Rani  25
8  Andi  31
~ |

```

Gambar 7.21

7.15 Menggabungkan *Data Frame* dengan Fungsi `dataframe()` berdasarkan Kolom

Pada Gambar 7.22, diperlihatkan dua *data frame*, yakni **tabel_gabungan** dan **tabel3**. Kolom-kolom atau vektor-vektor pada *data frame* **tabel_gabungan** adalah **Nama** dan **Usia**, sementara kolom-kolom atau vektor-vektor pada *data frame* **tabel3** adalah **Berat** dan **Jenis_Kelamin**. Selanjutnya kolom-kolom pada *data frame* **tabel_gabungan** dan **tabel3** digabung, dengan perintah R `data.frame(tabel_gabungan, tabel3)`. Hasilnya diperlihatkan pada Gambar 7.22.

```

> tabel_gabungan
  Nama Usia
1  Ugi    23
2  Egi    29
3  Sari   32
4  Intan  28
5  Budi   27
6  Sinta  27
7  Rani   25
8  Andi   31
> Berat <- c(54.43, 67.53, 66.65, 72.32, 49.54, 56.54, 52.45, 56.43)
> Jenis_Kelamin <- c("Laki-Laki", "Laki-Laki", "Perempuan", "Perempuan", "Laki-Laki", "Perempuan", "Perempuan", "Laki-Laki")
> tabel3 <- data.frame(Berat, Jenis_Kelamin)
> tabel3
  Berat Jenis_Kelamin
1 54.43      Laki-Laki
2 67.53      Laki-Laki
3 66.65      Perempuan
4 72.32      Perempuan
5 49.54      Laki-Laki
6 56.54      Perempuan
7 52.45      Perempuan
8 56.43      Laki-Laki
> tabel_gabungan <- data.frame(tabel_gabungan, tabel3)
> tabel_gabungan
  Nama Usia Berat Jenis_Kelamin
1  Ugi    23 54.43      Laki-Laki
2  Egi    29 67.53      Laki-Laki
3  Sari   32 66.65      Perempuan
4  Intan  28 72.32      Perempuan
5  Budi   27 49.54      Laki-Laki
6  Sinta  27 56.54      Perempuan
7  Rani   25 52.45      Perempuan
8  Andi   31 56.43      Laki-Laki

```

Gambar 7.22

7.16 Membuat *Data Frame* berdasarkan Matriks

Perhatikan Gambar 7.23. Pada Gambar 7.23, dibuat suatu matriks bernama **X** dengan jumlah baris sebanyak 3 dan jumlah kolom sebanyak 2. Selanjutnya perintah R **Y <- data.frame(X)** bertujuan untuk membuat *data frame* bernama **Y** berdasarkan data-data pada matriks **X**.

```

Console ~/
> X <- matrix(c(1,2,3,4,5,6), nrow = 3) #Membuat matriks dengan nama X
> X
      [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> Y <- data.frame(X) #Membuat data frame bernama Y berdasarkan matriks X
> Y
   x1 x2
1   1  4
2   2  5
3   3  6
> |

```

Gambar 7.23

7.17 Fungsi *is.data.frame()*, *is.list()*, *is.matrix()*, dan *is.vector()*

Pada Gambar 7.24 diperlihatkan penggunaan fungsi *is.data.frame()*, *is.list()*, *is.matrix()*, dan *is.vector()*. Penggunaan fungsi *is.data.frame()* bertujuan untuk mengetahui apakah suatu objek merupakan *data frame* atau tidak. Penggunaan fungsi *is.data.frame()* akan menghasilkan nilai logika TRUE jika suatu objek merupakan *data frame*, FALSE bila suatu objek bukan *data frame*.

- ⇒ Pada Gambar 7.24, dibentuk *data frame* bernama **contoh_dataframe**, *list* bernama **contoh_list**, matriks bernama **contoh_matriks**, dan vektor bernama **contoh_vektor**.
- ⇒ Perintah R **is.data.frame(contoh_dataframe)** menghasilkan nilai TRUE, yang berarti **contoh_dataframe** merupakan *data frame*. Perintah R **is.data.frame(contoh_list)** menghasilkan nilai FALSE, yang berarti **contoh_list** bukan merupakan *data frame*.
- ⇒ Perhatikan bahwa berdasarkan Gambar 7.24, terdapat informasi menarik bahwa suatu *data frame* juga merupakan *list*, yakni pada perintah R **is.list(contoh_dataframe)** yang menghasilkan TRUE.

```

Console ~/
> contoh_dataframe <- data.frame>Nama=c("Ugi","Egi"), Usia=c(27,23))
> contoh_list <- list>Nama=c("Ugi","Egi","Sinta"), Usia=c(27,25,24,29,28))
> contoh_matriks <- matrix(c(1,2,3,4,5,6), ncol=2, byrow=T)
> contoh_vektor <- c(4,3,4)
> contoh_dataframe
  Nama Usia
1  Ugi   27
2  Egi   23
> contoh_list
$Nama
[1] "Ugi" "Egi" "Sinta"

$Usia
[1] 27 25 24 29 28
> contoh_matriks
  [,1] [,2]
[1,]  1   2
[2,]  3   4
[3,]  5   6
> contoh_vektor
[1] 4 3 4
> is.data.frame(contoh_dataframe)
[1] TRUE
> is.data.frame(contoh_list)
[1] FALSE
> is.data.frame(contoh_matriks)
[1] FALSE
> is.data.frame(contoh_vektor)
[1] FALSE
> is.list(contoh_dataframe)
[1] TRUE
> is.list(contoh_list)
[1] TRUE
> is.list(contoh_matriks)
[1] FALSE
> is.list(contoh_vektor)
[1] FALSE
> is.matrix(contoh_dataframe)

```

Gambar 7.24

```

Console ~/ / ↵
$usia
[1] 27 25 24 29 28

> contoh_matriks
      [,1] [,2]
[1,]     1     2
[2,]     3     4
[3,]     5     6
> contoh_vektor
[1] 4 3 4
> is.data.frame(contoh_dataframe)
[1] TRUE
> is.data.frame(contoh_list)
[1] FALSE
> is.data.frame(contoh_matriks)
[1] FALSE
> is.data.frame(contoh_vektor)
[1] FALSE
> is.list(contoh_dataframe)
[1] TRUE
> is.list(contoh_list)
[1] TRUE
> is.list(contoh_matriks)
[1] FALSE
> is.list(contoh_vektor)
[1] FALSE
> is.matrix(contoh_dataframe)
[1] FALSE
> is.matrix(contoh_list)
[1] FALSE
> is.matrix(contoh_matriks)
[1] TRUE
> is.vector(contoh_vektor)
[1] TRUE
> is.vector(contoh_dataframe)
[1] FALSE

```

Gambar 7.25

7.18 Contoh Kode Program R: Simulasi Distribusi Sampling Rata-Rata Sampel

Berikut diberikan kode program R, untuk menghasilkan distribusi sampling dari rata-rata sampel.

```

library(prob)
seluruh_sampel <- urnsamples(c(1,2,3), size=2, replace=TRUE, ordered=TRUE)
seluruh_sampel
mode(seluruh_sampel)
class(seluruh_sampel)
jumlah_seluruh_sampel <- length(seluruh_sampel$X1)
jumlah_seluruh_sampel
jumlah_kolom <- length(seluruh_sampel)
jumlah_kolom

X1 <- seluruh_sampel[1]
mode(X1)
X1 <- unlist(X1)
mode(X1)
X2 <- seluruh_sampel[2]
mode(X2)
X2 <- unlist(X2)
mode(X2)

X1
X2

names(X1) <- NULL

```

```

names(X2) <- NULL

X1
X2

mode(X1)
mode(X2)

fungsi_rata_rata <- function(X1,X2, jumlah_seluruh_sampel)
{
  rata_rata_sampel <- vector(length=jumlah_seluruh_sampel)
  i=1
  for(n in X1)
  {
    rata_rata_sampel[i] = (X1[i] + X2[i])/2
    i = i+1
  }
  return(rata_rata_sampel)
}

rata_rata_sampel <- fungsi_rata_rata(X1, X2, jumlah_seluruh_sampel)
rata_rata_sampel

barplot(table(rata_rata_sampel))

data_frame <- data.frame(X1, X2, rata_rata_sampel)
data_frame

```

Gambar 7.26 sampai dengan Gambar 7.29 merupakan hasil eksekusi dari kode program R tersebut. Andaikan suatu populasi terdiri dari 3 bilangan, yakni 1, 2, dan 3. Selanjutnya dari 3 bilangan tersebut, dipilih 2, dengan memperhatikan urutan. Maka seluruh kemungkinan sampelnya adalah

(1,1), (1,2), (1,3)
 (2,1), (2,2), (2,3)
 (3,1), (3,2), (3,3)

Perhatikan bahwa terdapat 9 sampel yang mungkin terbentuk. Selanjutnya, dihitung nilai rata-rata dari masing-masing sampel tersebut. Sebagai contoh nilai rata-rata dari (1,1) adalah 2, nilai rata-rata dari (1,2) adalah 1,5, nilai rata-rata dari (1,3) adalah 4, dan seterusnya.

```

Console ~/
> library(prob)
> seluruh_sampel <- urnsamples(c(1,2,3), size=2, replace=TRUE, ordered=TRUE)
> seluruh_sampel
  x1 x2
1  1  1
2  2  1
3  3  1
4  1  2
5  2  2
6  3  2
7  1  3
8  2  3
9  3  3
> mode(se seluruh_sampel)
[1] "list"
> class(se seluruh_sampel)
[1] "data.frame"
> jumlah_seluruh_sampel <- length(se seluruh_sampel$x1)
> jumlah_seluruh_sampel
[1] 9
> jumlah_kolom <- length(se seluruh_sampel)
> jumlah_kolom
[1] 2
>
> x1 <- seluruh_sampel[1]
> mode(x1)
[1] "list"
> x1 <- unlist(x1)
> mode(x1)
[1] "numeric"
> x2 <- seluruh_sampel[2]
> mode(x2)
[1] "list"
> x2 <- unlist(x2)
> mode(x2)
[1] "numeric"

```

Gambar 7.26

```

Console ~/
> x1 <- unlist(x1)
> mode(x1)
[1] "numeric"
> x2 <- seluruh_sampel[2]
> mode(x2)
[1] "list"
> x2 <- unlist(x2)
> mode(x2)
[1] "numeric"
>
> x1
x11 x12 x13 x14 x15 x16 x17 x18 x19
  1  2  3  1  2  3  1  2  3
> x2
x21 x22 x23 x24 x25 x26 x27 x28 x29
  1  1  1  2  2  2  3  3  3
>
> names(x1) <- NULL
> names(x2) <- NULL
>
> x1
[1] 1 2 3 1 2 3 1 2 3
> x2
[1] 1 1 1 2 2 2 3 3 3
>
> mode(x1)
[1] "numeric"
> mode(x2)
[1] "numeric"
>
> fungsi_rata_rata <- function(x1,x2, jumlah_seluruh_sampel)
+ {
+   rata_rata_sampel <- vector(length=jumlah_seluruh_sampel)
+   i=1
+   for(n in x1)
+   {
+     rata_rata_sampel[i] = (x1[i] + x2[i])/2
+     i = i+1
+   }
+ }

```

Gambar 7.27

```

>
> mode(X1)
[1] "numeric"
> mode(X2)
[1] "numeric"
>
> fungsi_rata_rata <- function(X1,X2, jumlah_seluruh_sampel)
+ {
+   rata_rata_sampel <- vector(length=jumlah_seluruh_sampel)
+   i=1
+   for(n in X1)
+   {
+     rata_rata_sampel[i] = (X1[i] + X2[i])/2
+     i = i+1
+   }
+   return(rata_rata_sampel)
+ }
>
> rata_rata_sampel <- fungsi_rata_rata(X1, X2, jumlah_seluruh_sampel)
> rata_rata_sampel
[1] 1.0 1.5 2.0 1.5 2.0 2.5 2.0 2.5 3.0
>
> barplot(table(rata_rata_sampel))
>
> data_frame <- data.frame(X1, X2, rata_rata_sampel)
> data_frame
  X1 X2 rata_rata_sampel
1  1  1             1.0
2  2  1             1.5
3  3  1             2.0
4  1  2             1.5
5  2  2             2.0
6  3  2             2.5
7  1  3             2.0
8  2  3             2.5
9  3  3             3.0
>

```

Gambar 7.28

Gambar 7.29

7.19Berlatih

Gambar 7.30 diberikan *data frame* bernama **simpan_data** yang terdiri dari 4 vektor atau kolom, yakni **Nama**, **Jenis_Kelamin**, **Pendidikan**, dan **Jumlah_Bersaudara**.


```

Console ~/
> Nama <- c("Ugi", "Egi", "Andi", "Udin", "Boy", "Sinta", "Rini", "Rani", "Fitri", "Dini", "Rudi", "Firman",
"Agung", "Deni", "Salman", "Fina", "Suci", "wati", "Gina", "Silvia")
> Jenis_kelamin <- c("Laki-Laki", "Laki-Laki", "Laki-Laki", "Laki-Laki", "Laki-Laki", "Perempuan", "Per
empuan", "Perempuan", "Perempuan", "Laki-Laki", "Laki-Laki", "Laki-Laki", "Laki-Laki", "Laki-Laki", "Perempuan", "Perempuan", "Perempuan", "Perempuan", "Perempuan", "Perempuan")
> Pendidikan <- c("S1", "S2", "S3", "S1", "S2", "S2", "S2", "S1", "S3", "S2", "S1", "S2", "S3", "S1", "S1", "S1",
"S2", "S3", "S1", "S1")
> Jumlah_Bersaudara <- c(2,3,1,2,5,2,4,2,3,5,2,3,1,2,5,2,4,2,3,1)
> simpan_data <- data.frame(Nama, Jenis_kelamin, Pendidikan, Jumlah_Bersaudara)
> simpan_data
  Nama Jenis_kelamin Pendidikan Jumlah_Bersaudara
1  Ugi      Laki-Laki         S1                2
2  Egi      Laki-Laki         S2                3
3  Andi     Laki-Laki         S3                1
4  Udin     Laki-Laki         S1                2
5  Boy      Laki-Laki         S2                5
6  Sinta    Perempuan         S2                2
7  Rini     Perempuan         S2                4
8  Rani     Perempuan         S1                2
9  Fitri    Perempuan         S3                3
10 Dini     Perempuan         S2                5
11 Rudi     Laki-Laki         S1                2
12 Firman   Laki-Laki         S2                3
13 Agung    Laki-Laki         S3                1
14 Deni     Laki-Laki         S1                2
15 Salman   Laki-Laki         S1                5
16 Fina     Perempuan         S1                2
17 Suci     Perempuan         S2                4
18 wati     Perempuan         S3                2
19 Gina     Perempuan         S1                3
20 Silvia   Perempuan         S1                1
>

```

Gambar 7.30

Perintah R pada Gambar 7.31 `dim(simpan_data)` bertujuan untuk mengetahui dimensi dari *data frame* **simpan_data**. Diketahui *data frame* **simpan_data** terdiri dari 20 baris (responden) dan 4 kolom (vektor).

```

Console ~/
> dim(simpan_data)
[1] 20 4
>

```

Gambar 7.31

Perintah R pada Gambar 7.32 `names(simpan_data)` dan `colnames(simpan_data)` bertujuan untuk mengetahui nama-nama kolom atau vektor.

```

Console ~/
> names(simpan_data)
[1] "Nama"      "Jenis_kelamin"  "Pendidikan"    "Jumlah_Bersaudara"
> colnames(simpan_data)
[1] "Nama"      "Jenis_kelamin"  "Pendidikan"    "Jumlah_Bersaudara"
> |

```

Gambar 7.32

Perintah R pada Gambar 7.33 `levels(factor(simpan_data$Pendidikan))` bertujuan untuk mengetahui *level* dari suatu variabel. Pada variabel **Pendidikan** terdiri dari 3 *level*, yakni "S1", "S2", dan "S3". Sementara pada variabel **Jenis_Kelamin** terdiri dari 2 *level*, yakni "Laki-Laki" dan "Perempuan".

```

Console ~/
> levels(factor(simpan_data$Pendidikan))
[1] "S1" "S2" "S3"
> levels(factor(simpan_data$Jenis_kelamin))
[1] "Laki-Laki" "Perempuan"
> |

```

Gambar 7.33

Berdasarkan Gambar 7.34, diketahui jumlah responden laki-laki sebanyak 10 dan jumlah responden perempuan sebanyak 10. Perintah R untuk menampilkan distribusi frekuensi untuk data pada variabel **Jenis_Kelamin** adalah `table(factor(simpan_data$Jenis_Kelamin))`. Diketahui jumlah responden yang berpendidikan S1 sebanyak 9, berpendidikan S2 sebanyak 7, dan berpendidikan S3 sebanyak 4. Perintah R untuk menampilkan distribusi frekuensi untuk data pada variabel **Pendidikan** adalah `table(factor(simpan_data$Pendidikan))`.

```

Console ~/
> table(factor(simpan_data$Jenis_kelamin))

Laki-Laki Perempuan
      10       10
> table(factor(simpan_data$Pendidikan))

S1 S2 S3
 9  7  4
> |

```

Gambar 7.34

Perintah R pada Gambar 7.35 `barplot(table(factor(simpan_data$Jenis_Kelamin)))` bertujuan untuk menyajikan grafik batang frekuensi berdasarkan data pada kolom **Jenis_Kelamin**, sementara perintah R pada Gambar 7.36 `barplot(table(factor(simpan_data$Pendidikan)))` bertujuan untuk menyajikan grafik batang frekuensi berdasarkan data pada kolom **Pendidikan**. Perintah R pada Gambar 7.37 `simpan_data[1]` bertujuan untuk menampilkan data pada kolom ke-1 (**Nama**) dari *data frame* **simpan_data**, atau bisa juga dengan perintah R `simpan_data["Nama"]`. Perintah R pada Gambar 7.38 `simpan_data[c(1,3)]` dan `simpan_data[c("Nama","Pendidikan")]` bertujuan untuk menampilkan data pada kolom ke-1 dan kolom ke-3.

Gambar 7.35

Gambar 7.36

```
> simpan_data[1]
```

	Nama
1	Ugi
2	Egi
3	Andi
4	Udin
5	Boy
6	Sinta
7	Rini
8	Rani
9	Fitri
10	Dini
11	Rudi
12	Firman
13	Agung
14	Deni
15	Salman
16	Fina
17	Suci
18	Wati
19	Gina
20	Silvia

```
> simpan_data["Nama"]
```

```
Console ~/
```

```
> simpan_data["Nama"]
```

```
Nama
```

```
1 Ugi
```

```
2 Egi
```

```
3 Andi
```

```
4 Udin
```

```
5 Boy
```

```
6 Sinta
```

```
7 Rini
```

```
8 Rani
```

```
9 Fitri
```

```
10 Dini
```

```
11 Rudi
```

```
12 Firman
```

```
13 Agung
```

```
14 Deni
```

```
15 Salman
```

```
16 Fina
```

```
17 Suci
```

```
18 Wati
```

```
19 Gina
```

```
20 Silvia
```

Gambar 7.37

```
> simpan_data[c(1,3)]
```

```
Console ~/
```

```
> simpan_data[c(1,3)]
```

```
Nama Pendidikan
```

```
1 Ugi S1
```

```
2 Egi S2
```

```
3 Andi S3
```

```
4 Udin S1
```

```
5 Boy S2
```

```
6 Sinta S2
```

```
7 Rini S2
```

```
8 Rani S1
```

```
9 Fitri S3
```

```
10 Dini S2
```

```
11 Rudi S1
```

```
12 Firman S2
```

```
13 Agung S3
```

```
14 Deni S1
```

```
15 Salman S1
```

```
16 Fina S1
```

```
17 Suci S2
```

```
18 Wati S3
```

```
19 Gina S1
```

```
20 Silvia S1
```

```
> simpan_data[c("Nama", "Pendidikan")]
```

```
Console ~/
```

```
> simpan_data[c("Nama", "Pendidikan")]
```

```
Nama Pendidikan
```

```
1 Ugi S1
```

```
2 Egi S2
```

```
3 Andi S3
```

```
4 Udin S1
```

```
5 Boy S2
```

```
6 Sinta S2
```

```
7 Rini S2
```

```
8 Rani S1
```

```
9 Fitri S3
```

```
10 Dini S2
```

```
11 Rudi S1
```

```
12 Firman S2
```

```
13 Agung S3
```

```
14 Deni S1
```

```
15 Salman S1
```

```
16 Fina S1
```

```
17 Suci S2
```

```
18 Wati S3
```

```
19 Gina S1
```

```
20 Silvia S1
```

Gambar 7.38

Perintah R pada Gambar 7.39 `simpan_data[5,]` bertujuan untuk menampilkan data pada responden ke-5. Perintah R pada Gambar 7.40 `simpan_data[c(7,12,20),]` bertujuan untuk menampilkan data pada responden ke-7, 12, dan 20.

```
> simpan_data[5,]
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
5   Boy      Laki-Laki         S2                5
> |
```

Gambar 7.39

```
> simpan_data[c(7,12,20),]
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
7   Rini      Perempuan         S2                4
12 Firman     Laki-Laki         S2                3
20 Silvia    Perempuan         S1                1
> |
```

Gambar 7.40

```
> simpan_data[c(5:10),]
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
5   Boy      Laki-Laki         S2                5
6   Sinta    Perempuan         S2                2
7   Rini      Perempuan         S2                4
8   Rani      Perempuan         S1                2
9   Fitri     Perempuan         S3                3
10  Dini      Perempuan         S2                5
> |
```

Gambar 7.41

Perintah R pada Gambar 7.41 `simpan_data[c(5:10),]` bertujuan untuk menampilkan data pada responden ke-5 sampai dengan responden ke-10. Perintah R pada Gambar 7.42 `simpan_data[c(5:10),][c(1,3)]` bertujuan untuk menampilkan data pada responden ke-5 sampai dengan responden ke-10 dan hanya menampilkan kolom **Nama** dan **Pendidikan**.

```
Console ~/
> simpan_data[c(5:10),][c(1,3)]
  Nama Pendidikan
5   Boy         S2
6   Sinta       S2
7   Rini        S2
8   Rani        S1
9   Fitri       S3
10  Dini        S2
>
```

Gambar 7.42

```

Console ~/
> simpan_indeks <- which(simpan_data["Pendidikan"]=="S2")
> simpan_data[simpan_indeks,][c("Nama","Pendidikan")]
  Nama Pendidikan
2   Egi         S2
5   Boy         S2
6  Sinta        S2
7   Rini        S2
10  Dini        S2
12 Firman       S2
17  Suci        S2
> |

Console ~/
> simpan_data["Pendidikan"]=="S2"
Pendidikan
[1,] FALSE
[2,]  TRUE
[3,] FALSE
[4,] FALSE
[5,]  TRUE
[6,]  TRUE
[7,]  TRUE
[8,] FALSE
[9,] FALSE
[10,] TRUE
[11,] FALSE
[12,]  TRUE
[13,] FALSE
[14,] FALSE
[15,] FALSE
[16,] FALSE
[17,]  TRUE
[18,] FALSE
[19,] FALSE
[20,] FALSE
> which(simpan_data["Pendidikan"]=="S2")
[1] 2 5 6 7 10 12 17
> |

```

Gambar 7.43

Gambar 7.43 bertujuan untuk menampilkan responden yang berpendidikan S2. Perhatikan bahwa terdapat 7 responden yang berpendidikan S2, yakni Egi, Boy, Sinta, Rini, Dini, Firman, dan Suci.

- ⇒ Perintah R **simpan_data["Pendidikan"]=="S2"** akan menampilkan nilai logika, yakni TRUE atau FALSE dari masing-masing responden.
- ⇒ Jika responden berpendidikan S2, maka akan menampilkan nilai TRUE, sementara jika responden tidak berpendidikan S2, maka akan menampilkan nilai FALSE.
- ⇒ Berdasarkan Gambar 7.42, responden yang berpendidikan S2 berada pada posisi atau indeks ke-2, 5, 6, 7, 10, 12, dan 17.
- ⇒ Perintah R **which(simpan_data["Pendidikan"]=="S2")** akan menampilkan indeks responden yang berpendidikan S2. Perintah R **simpan_indeks <- which(simpan_data["Pendidikan"]=="S2")** bertujuan menyimpan indeks responden yang berpendidikan S2.
- ⇒ Perintah R **simpan_data[simpan_indeks,][c("Nama","Pendidikan")]** bertujuan untuk menampilkan responden yang berpendidikan S2, dan hanya menampilkan kolom **Nama** dan **Pendidikan**.

```

Console ~/
> simpan_data["Pendidikan"]=="S1" & simpan_data["Jumlah_Bersaudara"]==2
Pendidikan
[1,] TRUE
[2,] FALSE
[3,] FALSE
[4,] TRUE
[5,] FALSE
[6,] FALSE
[7,] FALSE
[8,] TRUE
[9,] FALSE
[10,] FALSE
[11,] TRUE
[12,] FALSE
[13,] FALSE
[14,] TRUE
[15,] FALSE
[16,] TRUE
[17,] FALSE
[18,] FALSE
[19,] FALSE
[20,] FALSE
> which(simpan_data["Pendidikan"]=="S1" & simpan_data["Jumlah_Bersaudara"]==2)
[1] 1 4 8 11 14 16
> simpan_indeks <- which(simpan_data["Pendidikan"]=="S1" & simpan_data["Jumlah_Bersaudara"]==2)
> simpan_data[simpan_indeks,][c("Nama","Pendidikan","Jumlah_Bersaudara")]
  Nama Pendidikan Jumlah_Bersaudara
1  Ugi          S1                2
4  Udin          S1                2
8  Rani          S1                2
11 Rudi          S1                2
14 Deni          S1                2
16 Fina          S1                2
> |

```

Gambar 7.44

Pada Gambar 7.44, bertujuan untuk menampilkan responden yang berpendidikan S1 dengan jumlah bersaudara sebanyak 2. Diketahui terdapat 6 responden yang berpendidikan S1 dengan jumlah bersaudara sebanyak 2, yakni Ugi, Udin, Rani, Rudi, Deni, dan Fina.

```

Console ~/
> simpan_indeks <- which(simpan_data["Pendidikan"]=="S3" & (simpan_data["Jumlah_Bersaudara"]==2 | simpan_data["Jumlah_Bersaudara"]==3) )
> simpan_data[simpan_indeks,][c("Nama","Pendidikan","Jumlah_Bersaudara")]
  Nama Pendidikan Jumlah_Bersaudara
9  Fitri          S3                3
18 wati           S3                2
> |

```

Gambar 7.45

Pada Gambar 7.45 bertujuan untuk menampilkan responden yang berpendidikan S3, dengan jumlah bersaudara 2 atau 3.

```

Console ~/
> simpan_data
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1   Ugi      Laki-Laki      S1           2
2   Egi      Laki-Laki      S2           3
3   Andi     Laki-Laki      S3           1
4   Udin     Laki-Laki      S1           2
5   Boy      Laki-Laki      S2           5
6   Sinta    Perempuan      S2           2
7   Rini     Perempuan      S2           4
8   Rani     Perempuan      S1           2
9   Fitri    Perempuan      S3           3
10  Dini     Perempuan      S2           5
11  Rudi     Laki-Laki      S1           2
12  Firman   Laki-Laki      S2           3
13  Agung   Laki-Laki      S3           1
14  Deni     Laki-Laki      S1           2
15  Salman   Laki-Laki      S1           5
16  Fina     Perempuan      S1           2
17  Suci     Perempuan      S2           4
18  Wati     Perempuan      S3           2
19  Gina     Perempuan      S1           3
20 Silvia   Perempuan      S1           1
> library(dplyr)
> grup <- group_by(simpan_data, Jenis_Kelamin, Pendidikan)
> summarise(grup, Jumlah_Responden = length(Jumlah_Bersaudara) )
source: local data frame [6 x 3]
Groups: Jenis_Kelamin [?]

  Jenis_Kelamin Pendidikan Jumlah_Responden
  <fctr>      <fctr>      <int>
1   Laki-Laki      S1           5
2   Laki-Laki      S2           3
3   Laki-Laki      S3           2
4   Perempuan      S1           4
5   Perempuan      S2           4
6   Perempuan      S3           2
>

```

Gambar 7.46

Berdasarkan Gambar 7.46, diketahui:

- ⇒ Jumlah responden yang berjenis kelamin laki-laki dan berpendidikan S1 sebanyak 5 responden.
- ⇒ Jumlah responden yang berjenis kelamin laki-laki dan berpendidikan S2 sebanyak 3 responden.
- ⇒ Jumlah responden yang berjenis kelamin laki-laki dan berpendidikan S3 sebanyak 2 responden.
- ⇒ Jumlah responden yang berjenis kelamin perempuan dan berpendidikan S1 sebanyak 4 responden.
- ⇒ Jumlah responden yang berjenis kelamin perempuan dan berpendidikan S2 sebanyak 5 responden.
- ⇒ Jumlah responden yang berjenis kelamin perempuan dan berpendidikan S3 sebanyak 2 responden.

```

> simpan_data
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1   Ugi      Laki-Laki      S1           2
2   Egi      Laki-Laki      S2           3
3   Andi     Laki-Laki      S3           1
4   Udin     Laki-Laki      S1           2
5   Boy      Laki-Laki      S2           5
6   Sinta    Perempuan      S2           2
7   Rini     Perempuan      S2           4
8   Rani     Perempuan      S1           2
9   Fitri    Perempuan      S3           3
10  Dini     Perempuan      S2           5
11  Rudi     Laki-Laki      S1           2
12  Firman   Laki-Laki      S2           3
13  Agung   Laki-Laki      S3           1
14  Deni     Laki-Laki      S1           2
15  Salman   Laki-Laki      S1           5
16  Fina     Perempuan      S1           2
17  Suci     Perempuan      S2           4
18  wati     Perempuan      S3           2
19  Gina     Perempuan      S1           3
20  Silvia   Perempuan      S1           1
> library(dplyr)
> grup <- group_by(simpan_data, Jenis_Kelamin, Pendidikan)
> summarise(filter(grup, Jumlah_Bersaudara==3), Jumlah_Responden = length(Jumlah_Bersaudara))
Source: local data frame [3 x 3]
Groups: Jenis_Kelamin [?]

  Jenis_Kelamin Pendidikan Jumlah_Responden
  <fctr>      <fctr>      <int>
1   Laki-Laki      S2           2
2   Perempuan      S1           1
3   Perempuan      S3           1
>

```

Gambar 7.47

Berdasarkan Gambar 7.47, diketahui:

- ⇒ Terdapat 2 responden berjenis kelamin laki-laki, berpendidikan S2, **dengan jumlah bersaudara sebanyak 3.**
- ⇒ Terdapat 1 responden berjenis kelamin perempuan, berpendidikan S1, **dengan jumlah bersaudara sebanyak 3.**
- ⇒ Terdapat 1 responden berjenis kelamin perempuan, berpendidikan S3, **dengan jumlah bersaudara sebanyak 3.**

BAB 8

FACTOR

8.1 Factor

Gambar 8.1 diberikan ilustrasi dalam R penggunaan fungsi *factor()* .

```
Console ~/
> bilangan <- c(1,1,2,2,2,3,3,3,4,5,5)
> jawaban <- c("ya","tidak","ya","ya","tidak")
> bilangan
[1] 1 1 2 2 2 3 3 3 4 5 5
> jawaban
[1] "ya"      "tidak"   "ya"      "ya"      "tidak"
> mode(bilangan)
[1] "numeric"
> mode(jawaban)
[1] "character"
> class(bilangan)
[1] "numeric"
> class(jawaban)
[1] "character"
> is.vector(bilangan)
[1] TRUE
> is.vector(jawaban)
[1] TRUE
> is.factor(bilangan)
[1] FALSE
> is.factor(jawaban)
[1] FALSE
>

Console ~/
> bilangan2 <- factor(c(1,1,2,2,2,3,3,3,4,5,5))
> jawaban2 <- factor(c("ya","tidak","ya","ya","tidak"))
> bilangan2
[1] 1 1 2 2 2 3 3 3 4 5 5
Levels: 1 2 3 4 5
> jawaban2
[1] ya      tidak ya      ya      tidak
Levels: tidak ya
> mode(bilangan2)
[1] "numeric"
> mode(jawaban2)
[1] "numeric"
> class(bilangan2)
[1] "factor"
> class(jawaban2)
[1] "factor"
> levels(bilangan2)
[1] "1" "2" "3" "4" "5"
> levels(jawaban2)
[1] "tidak" "ya"
> mode(levels(bilangan2))
[1] "character"
> mode(levels(jawaban2))
[1] "character"
> is.vector(bilangan2)
[1] FALSE
> is.vector(jawaban2)
[1] FALSE
> is.factor(bilangan2)
[1] TRUE
> is.factor(jawaban2)
[1] TRUE
> |
```

Gambar 8.1

Berdasarkan Gambar 8.1, dapat ditarik informasi sebagai berikut.

- ⇒ Dibentuk vektor bernama **bilangan** dan **jawaban**. Diketahui *mode* dari vektor **bilangan** adalah *numeric*, sementara *mode* dari vektor **jawaban** adalah *character*. Diketahui juga *class* dari vektor **bilangan** adalah *numeric*, sementara *class* dari vektor **jawaban** adalah *character*.
- ⇒ Selanjutnya digunakan fungsi **factor()** untuk membentuk objek *factor* bernama **bilangan2** dan **jawaban2**.
- ⇒ Perhatikan bahwa *mode* dari *factor* **bilangan2** adalah *numeric*, begitu juga *mode* dari *factor* **jawaban2** adalah *numeric* (meskipun data pada *factor* **jawaban2** adalah huruf/karakter (“ya”, “tidak”)). Diketahui juga *class* dari *factor* **bilangan2** adalah *factor*, dan *class* dari *factor* **jawaban2** juga *factor*.

```

Console ~/
> is.vector(bilangan)
[1] TRUE
> is.vector(jawaban)
[1] TRUE
> is.factor(bilangan)
[1] FALSE
> is.factor(jawaban)
[1] FALSE
> is.vector(bilangan2)
[1] FALSE
> is.vector(jawaban2)
[1] FALSE
> is.factor(bilangan2)
[1] TRUE
> is.factor(jawaban2)
[1] TRUE

```

Gambar 8.2

- ⇒ Berdasarkan Gambar 8.2, perintah R **is.vector(bilangan)** dan **is.vector(jawaban)** menampilkan hasil TRUE, yang berarti **bilangan** dan **jawaban** adalah vektor. Perintah R **is.factor(bilangan)** dan **is.factor(jawaban)** menampilkan hasil FALSE, yang berarti **bilangan** dan **jawaban** bukan *factor*.
- ⇒ Perintah R **is.vector(bilangan2)** dan **is.vector(jawaban2)** menampilkan hasil FALSE, yang berarti **bilangan2** dan **jawaban2** bukan merupakan vektor. Perintah R **is.factor(bilangan2)** dan **is.factor(jawaban2)** menampilkan hasil TRUE, yang berarti **bilangan2** dan **jawaban2** adalah *factor*.

```

Console ~/
> bilangan2
[1] 1 1 2 2 2 3 3 3 4 5 5
Levels: 1 2 3 4 5
> mode(bilangan2)
[1] "numeric"
> mode(levels(bilangan2))
[1] "character"

```

Gambar 8.3

Berdasarkan Gambar 8.3, *mode* dari nilai-nilai pada *factorbilangan2* adalah *numeric*, sementara *mode* dari *level-level* atau kategori-kategori pada *factorbilangan2* adalah *character*. Sebagaimana yang diutarakan oleh Phil Spector (2008:67) sebagai berikut.

“Factors in R are stored as a vector of integer values with a corresponding set of character values to use when the factor is displayed. The factor function is used to create a factor. The only required argument to factor is a vector of values which will be returned as a vector of factor values. Both numeric and character variables can be made into factors, but factor’s levels will always be character values. You can see the possible levels for a factor by calling the levels function; the nlevels function will return the number of levels of a factor.”

```
> bilangan2
[1] 1 1 2 2 2 3 3 3 4 5 5
Levels: 1 2 3 4 5
> mode(bilangan2)
[1] "numeric"
> mode(levels(bilangan2))
[1] "character"
> jawaban2
[1] ya      tidak ya      ya      tidak
Levels: tidak ya
> mode(jawaban2)
[1] "numeric"
> mode(levels(jawaban2))
[1] "character"
> levels(jawaban2)
[1] "tidak" "ya"
> |
```

Gambar 8.4

Berdasarkan Gambar 8.4, nilai-nilai dari *factor jawaban2* adalah ya, tidak, ya, ya, dan tidak, yang mana memiliki *mode numeric*. Sementara *level-level* dari *factor jawaban2* adalah ya dan tidak, yang mana berjenis *character*.

8.2 Mengganti Level dari Suatu Factor dengan Fungsi *levels()*

Perhatikan ilustrasi pada Gambar 8.5. Berdasarkan Gambar 8.5, digunakan fungsi *factor()* untuk membentuk *factor* bernama *fdata*. Diketahui *level* dari *factor fdata* pada awalnya adalah 1, 2, dan 3. Kemudian perintah R `levels(fdata) <- c("I","II","III")` mengganti *level* dari *factor fdata*, yang sebelumnya 1, 2, dan 3, menjadi I, II, dan III.

```

Console ~/
> data <- c(1,2,3,3,3,2,1,2,2,3,3,3,2,1,2)
> fdata <- factor(c(1,2,3,3,3,2,1,2,2,3,3,3,2,1,2))
> fdata
[1] 1 2 3 3 3 2 1 2 2 3 3 3 2 1 2
Levels: 1 2 3
> levels(fdata) <- c("I","II","III")
> fdata
[1] I   II  III III III II  I   II  II  III III III II  I   II
Levels: I II III
> |

```

Gambar 8.5

Pada Gambar 8.6, melibatkan argumen *levels* pada pembentukan *factor* yang bernama **fbulan**.

```

Console ~/
> bulan <- c("Februari","Februari","Maret","Juni","Januari","Januari","S
eptember","Oktober","Oktober","Maret","Desember","Desember","September",
"Desember","Juni","Juli","Mei","Juli")
> fbulan <- factor(bulan)
> table(fbulan)
fbulan
Desember  Februari  Januari      Juli      Juni      Maret
          3          2          2          2          2          2
      Mei  Oktober September
          1          2          2
> fbulan
[1] Februari Februari Maret      Juni      Januari Januari
[7] September Oktober Oktober      Maret      Desember Desember
[13] September Desember Juni      Juli      Mei      Juli
9 Levels: Desember Februari Januari Januari Juli Juni Maret ... September
>
> fbulan <- factor(bulan, levels = c("Januari","Februari","Maret","April
","Mei","Juni","Juli","Agustus","September","Oktober","November","Desemb
er"))
> table(fbulan)
fbulan
Januari  Februari  Maret      April      Mei      Juni
          2          2          2          0          1          2
      Juli  Agustus September      Oktober      November      Desember
          2          0          2          2          0          3
>

```

Gambar 8.6

BAB 9

FUNGSI TABLE DAN FTABLE

9.1 Membuat Tabel Distribusi Frekuensi dengan Fungsi *table()*

Fungsi *table()* dapat digunakan untuk membuat tabel distribusi frekuensi. Perhatikan Gambar 9.1.

```
Console ~/
> bilangan <- c(1,2,3,2,3,3,4,5,3,2,3,4,5,5,5,5,3,2,1,3)
> length(bilangan)
[1] 20
> table(bilangan)
bilangan
 1  2  3  4  5
 2  4  7  2  5
> |
```

Gambar 9.1

Berdasarkan Gambar 9.1 dibentuk vektor bernama **bilangan** yang terdiri dari 20 bilangan. Kemudian fungsi *table()* digunakan untuk membuat distribusi frekuensi berdasarkan bilangan-bilangan pada vektor **bilangan**. Diketahui nilai 1 terdapat sebanyak 2, nilai 2 terdapat sebanyak 4, nilai 3 terdapat sebanyak 7, dan seterusnya.

```
Console ~/
> jawaban <- c("ya","tidak","tidak","tidak","ya")
> length(jawaban)
[1] 5
> table(jawaban)
jawaban
tidak   ya
      3    2
.      1
```

Gambar 9.2

Berdasarkan Gambar 9.2 dibentuk vektor bernama **jawaban** yang terdiri dari 5 elemen. Kemudian fungsi *table()* digunakan untuk membuat distribusi frekuensi berdasarkan elemen-elemen pada vektor **jawaban**. Diketahui jawaban “ya” sebanyak 2, sementara jawaban “tidak” sebanyak 3.

Gambar 9.3 dan 9.4 digunakan fungsi *barplot()* untuk membuat grafik batang frekuensi. Perbedaan grafik batang antara Gambar 9.3 dan Gambar 9.4 adalah pada Gambar 9.3, tidak menyajikan nilai frekuensi pada tiap-tiap batang, sementara pada Gambar 9.4 menyajikan nilai frekuensi pada tiap-tiap batang. Berdasarkan Gambar 9.4, nilai 1 sebanyak 2, nilai 2 sebanyak 4, nilai 3 sebanyak 7, dan seterusnya.

```

Console ~/
> bilangan <- c(1,2,3,2,3,3,4,5,3,2,3,4,5,5,5,5,3,2,1,3)
> length(bilangan)
[1] 20
> table(bilangan)
bilangan
1 2 3 4 5
2 4 7 2 5
> tabel <- table(bilangan)
> barplot(tabel, col = c(heat.colors(5)), main="Grafik Batang Frekuensi", xlab="Nilai", ylab = "Jumlah")
>

```

Gambar 9.3

Pada Gambar 9.3, argumen *main* bertujuan untuk memberi judul dari grafik batang, argumen *xlab* untuk memberi nama pada sumbu *x*, dan argumen *ylab* untuk memberi nama pada sumbu *y*. Argumen *col* bertujuan untuk memberi warna pada grafik batang.

```

Console ~/
> A <- barplot(tabel, col = c(heat.colors(5)), main="Grafik Batang Frekuensi", xlab="Nilai", ylab = "Jumlah")
> text(A, 0.5, tabel)
>

```

Gambar 9.4

9.2 Membuat Tabel Distribusi Frekuensi untuk Data Berkelompok

Gambar 9.5 diberikan ilustrasi dalam R membuat tabel distribusi frekuensi untuk data berkelompok. Pada ilustrasi tersebut digunakan fungsi `cut()` untuk membuat suatu interval. Argumen *break* digunakan untuk menentukan batas-batas interval. Diketahui terdapat 7 nilai yang termasuk ke dalam interval $1 \leq \text{bilangan} < 5$, sementara terdapat 9 nilai yang termasuk ke dalam interval $5 \leq \text{bilangan} \leq 10$.

```
Console ~/
> bilangan <- c(1,2,3,4,5,6,7,8,9,10,10,9,8,4,3,2)
> distribusi_frekuensi <- cut(bilangan, breaks = c(1,5,10), include.lowest = TRUE, right = FALSE)
> tabel <- table(distribusi_frekuensi)
> tabel
distribusi_frekuensi
[1,5] [5,10]
      7      9
> names(tabel)
[1] "[1,5]" "[5,10]"
> names(tabel) <- c("1<=x<5", "5<=x<=10")
> tabel
1<=x<5 5<=x<=10
      7      9
.
```

Gambar 9.5

9.3 Membuat Tabel Contingency Dua Arah dengan Fungsi `table()`

Misalkan dibentuk *data frame* dengan nama `data_frame`, seperti pada Gambar 9.6.

```
Console ~/
> pendidikan <- c("s1","s1","s1","s1","s1","s2","s2","s2","s2","s2")
> jenis_kelamin <- c("laki-laki","laki-laki","laki-laki","perempuan","perempuan","perempuan","perempuan","perempuan","laki-laki")
> data_frame <- data.frame(pendidikan,jenis_kelamin)
> data_frame
  pendidikan jenis_kelamin
1          s1      laki-laki
2          s1      laki-laki
3          s1      laki-laki
4          s1      perempuan
5          s1      perempuan
6          s2      perempuan
7          s2      perempuan
8          s2      perempuan
9          s2      perempuan
10         s2      laki-laki
> table(data_frame)
      pendidikan jenis_kelamin
      s1      laki-laki      perempuan
      3          3          2
      s2          1          4
> mode(table(data_frame))
[1] "numeric"
> class(table(data_frame))
[1] "table"
> simpan <- table(data_frame)
> simpan
      pendidikan jenis_kelamin
      s1      laki-laki      perempuan
      3          3          2
      s2          1          4
> simpan[,1]
s1 s2
3  1
> simpan[1,]
laki-laki perempuan
      3          2
.
```

Digunakan fungsi `table()` pada *data frame* `data_frame`, sehingga menampilkan tabel *contingency*.

Gambar 9.6

Berdasarkan Gambar 9.6, digunakan fungsi **table()** pada *data frame* **data_frame**, sehingga menampilkan tabel *contingency*. Berdasarkan tabel *contingency* tersebut, diketahui:

- ⇒ Terdapat sebanyak 3 responden yang berpendidikan S1, dengan jenis kelamin laki-laki.
- ⇒ Terdapat sebanyak 2 responden yang berpendidikan S2, dengan jenis kelamin perempuan.
- ⇒ Terdapat sebanyak 1 responden yang berpendidikan S2, dengan jenis kelamin laki-laki.
- ⇒ Terdapat sebanyak 4 responden yang berpendidikan S2, dengan jenis kelamin perempuan.

Berdasarkan Gambar 9.6, perintah R **class(table(data_frame))** memperlihatkan hasil *table*.

Gambar 9.7 hingga Gambar 9.9 diperlihatkan langkah demi langkah untuk menyajikan grafik batang frekuensi. Fungsi **barplot()** digunakan untuk membuat grafik batang frekuensi.

```

Console ~/
> pendidikan <- c("s1","s1","s1","s1","s1","s2","s2","s2","s2","s2")
> jenis_kelamin <- c("laki-laki","laki-laki","laki-laki","perempuan",
+ "perempuan","perempuan","perempuan","perempuan","laki-laki")
> data_frame <- data.frame(pendidikan,jenis_kelamin)
> data_frame
  pendidikan jenis_kelamin
1          s1      laki-laki
2          s1      laki-laki
3          s1      laki-laki
4          s1      perempuan
5          s1      perempuan
6          s2      perempuan
7          s2      perempuan
8          s2      perempuan
9          s2      perempuan
10         s2      laki-laki
> tabel_kontingensi <- table(data_frame)
> tabel_kontingensi
      jenis_kelamin
pendidikan laki-laki perempuan
      s1           3           2
      s2           1           4
> |

Console ~/
> tabel_kontingensi
      jenis_kelamin
pendidikan laki-laki perempuan
      s1           3           2
      s2           1           4
> as.data.frame(tabel_kontingensi)
  pendidikan jenis_kelamin Freq
1          s1      laki-laki   3
2          s2      laki-laki   1
3          s1      perempuan   2
4          s2      perempuan   4
> frekuensi <- as.data.frame(tabel_kontingensi)$Freq
> frekuensi
[1] 3 1 2 4
> |

Console ~/
> A <- barplot(tabel_kontingensi, main="Grafik Batang Frekuensi berdasarkan P
endidikan dan Jenis Kelamin", xlab="Jenis Kelamin", ylab="Jumlah", col = c("g
reen","yellow"), legend=rownames(tabel_kontingensi), args.legend = list(cex=.
9, x="topright",title="Pendidikan"), ylim=c(0,8), beside = TRUE )
> text(A,0.5, frekuensi, cex=1, pos=2)
> |

```


Grafik Batang Frekuensi berdasarkan Pendidikan dan Jenis Kelamin

Gambar 9.7

```

Console ~/
> barplot(tabel_kontingensi, main="Grafik Batang Frekuensi berdasarkan Pendidikan dan Jenis Kelamin", xlab="Jenis Kelamin", ylab="Jumlah", col = c("green", "yellow"), legend=rownames(tabel_kontingensi), args.legend = list(cex=.9, x="topright", title="Pendidikan"), ylim=c(0,8) )
>

```

Gambar 9.8

```

Console ~/
> A <- barplot(tabel_kontingensi, main="Grafik Batang Frekuensi berdasarkan Pendidikan dan Jenis Kelamin", xlab="Jumlah", ylab="Jenis kelamin", col=c("green","yellow"), legend=rownames(tabel_kontingensi), args.legend=list(cex=.9, x="topright", title="Pendidikan"), xlim=c(0,5), beside = TRUE, horiz = TRUE)
> text(x=tabel_kontingensi+0.1, y=A, labels = as.character(tabel_kontingensi))
>

```

Gambar 9.9

9.4 Membuat Tabel *Contingency* Tiga Arah dengan Fungsi *table()*

Perhatikan Gambar 9.10. Pada Gambar 9.10 digunakan fungsi *ftable()* untuk membentuk tabel *contingency* tiga arah dari *data frame* **data_frame**. Berdasarkan tabel *contingency* tiga arah tersebut, diketahui:

- ⇒ Terdapat sebanyak 1 responden dengan pendidikan S1, jenis kelamin laki-laki, dan belum menikah.
- ⇒ Terdapat sebanyak 2 responden dengan pendidikan S1, jenis kelamin laki-laki, dan sudah menikah, dan seterusnya.

```

Console ~/
> data_frame
  pendidikan jenis_kelamin
1          S1      laki-laki
2          S1      laki-laki
3          S1      laki-laki
4          S1      perempuan
5          S1      perempuan
6          S2      perempuan
7          S2      perempuan
8          S2      perempuan
9          S2      perempuan
10         S2      laki-laki
> status <- c("sudah menikah","sudah menikah","belum menikah","sudah meni
kah","sudah menikah","belum menikah","sudah menikah","belum menikah","bel
um menikah","sudah menikah")
> data_frame <- data.frame(data_frame,status)
> data_frame
  pendidikan jenis_kelamin      status
1          S1      laki-laki sudah menikah
2          S1      laki-laki sudah menikah
3          S1      laki-laki belum menikah
4          S1      perempuan sudah menikah
5          S1      perempuan sudah menikah
6          S2      perempuan belum menikah
7          S2      perempuan sudah menikah
8          S2      perempuan belum menikah
9          S2      perempuan belum menikah
10         S2      laki-laki sudah menikah
> ftable(data_frame)
      status belum menikah sudah menikah
pendidikan jenis_kelamin
S1      laki-laki          1            2
      perempuan          0            2
S2      laki-laki          0            1
      perempuan          3            1
> |

```

Gambar 9.10

9.5 Membuat Tabel *Contingency* Empat Arah dengan Fungsi *table()*

Perhatikan Gambar 9.11. Pada Gambar 9.11 digunakan fungsi *ftable()* untuk membentuk tabel *contingency* empat arah dari *data frame* **data_frame**. Berdasarkan tabel *contingency* tiga empat tersebut, diketahui:

- ⇒ Tidak terdapat responden dengan jenis kelamin laki-laki, pendidikan S1, status belum menikah, dan hobi memasak.
- ⇒ Terdapat satu responden dengan jenis kelamin laki-laki, pendidikan S1, status belum menikah, dan hobi membaca, dan seterusnya.

```
Console ~/
> pendidikan <- c("S1","S1","S1","S1","S1","S2","S2","S2","S2","S2")
> hobi <- c("membaca","membaca","membaca","membaca","memasak","membaca","membaca","memasak","membaca","memasak")
> jenis_kelamin <- c("laki-laki","laki-laki","laki-laki","perempuan","perempuan","perempuan","perempuan","perempuan","laki-laki")
> status <- c("sudah menikah","sudah menikah","belum menikah","sudah menikah","sudah menikah","belum menikah","sudah menikah","belum menikah","belum menikah","sudah menikah")
>
> data_frame <- data.frame(jenis_kelamin,pendidikan,status,hobi)
> data_frame
  jenis_kelamin pendidikan      status      hobi
1      laki-laki         S1 sudah menikah membaca
2      laki-laki         S1 sudah menikah membaca
3      laki-laki         S1 belum menikah membaca
4      perempuan         S1 sudah menikah membaca
5      perempuan         S1 sudah menikah memasak
6      perempuan         S2 belum menikah membaca
7      perempuan         S2 sudah menikah membaca
8      perempuan         S2 belum menikah memasak
9      perempuan         S2 belum menikah membaca
10     laki-laki         S2 sudah menikah memasak
> ftable(data_frame)
      jenis_kelamin pendidikan      status      hobi memasak membaca
laki-laki      S1      belum menikah           0           1
              S1      sudah menikah           0           2
              S2      belum menikah           0           0
              S2      sudah menikah           1           0
perempuan      S1      belum menikah           0           0
              S1      sudah menikah           1           1
              S2      belum menikah           1           2
              S2      sudah menikah           0           1
> |
```

Gambar 9.11

9.6 Menghitung beberapa Ukuran Statistik berdasarkan Kelompok

Gambar 9.12 diberikan ilustrasi dalam R untuk menghitung nilai rata-rata, minimum, dan maksimum dari data gaji, berdasarkan tingkat pendidikan. Berdasarkan Gambar 9.12 digunakan fungsi *subset()* untuk melakukan perhitungan berdasarkan kelompok. Diketahui rata-rata gaji pada kelompok responden dengan pendidikan S1 adalah 4,74 juta, sementara rata-rata gaji pada kelompok responden dengan pendidikan S2 adalah 7,5 juta. Diketahui juga gaji minimum pada responden dengan pendidikan S1 adalah 3,2 juta, sementara gaji maksimum pada responden dengan pendidikan S2 adalah 9.9 juta.

```

> data_frame
pendidikan gaji
1          S1  4.6
2          S1  5.5
3          S1  3.2
4          S1  5.5
5          S1  4.9
6          S2  5.9
7          S2  8.7
8          S2  6.5
9          S2  6.5
10         S2  9.9
> mean(subset(data_frame, pendidikan=="S1")$gaji )
[1] 4.74
> mean(subset(data_frame, pendidikan=="S2")$gaji )
[1] 7.5
> min(subset(data_frame, pendidikan=="S1")$gaji )
[1] 3.2
> max(subset(data_frame, pendidikan=="S2")$gaji )
[1] 9.9

```

Gambar 9.12

9.7 Menghitung beberapa Ukuran Statistik berdasarkan Kelompok dengan *Package dplyr*

Gambar 9.13 diberikan ilustrasi dalam R untuk menghitung nilai rata-rata, minimum, dan maksimum dari variabel gaji dan pengeluaran berdasarkan variabel kategori jenis kelamin dan pendidikan. **Perhatikan bahwa** perhitungan tersebut menggunakan fungsi *summarise()* yang mana terlebih dahulu memasang (*attach*) *package dplyr*.

```

> data_frame
pendidikan jenis_kelamin gaji pengeluaran
1          S1      laki-laki  4.6         3.3
2          S1      laki-laki  5.5         3.5
3          S1      laki-laki  3.2         2.8
4          S1      perempuan  5.5         3.1
5          S1      perempuan  4.9         3.3
6          S2      perempuan  5.9         4.0
7          S2      perempuan  8.7         4.1
8          S2      perempuan  6.5         3.9
9          S2      laki-laki  6.5         4.4
10         S2      laki-laki  9.9         3.5
> library(dplyr)
> library(dplyr)
> grup <- group_by(data_frame, pendidikan, jenis_kelamin)
> grup %>% summarise(
+   minimum_gaji = min(gaji),
+   minimum_pengeluaran = min(pengeluaran),
+   maksimum_gaji = max(gaji),
+   maksimum_pengeluaran = max(pengeluaran),
+   rata_rata_gaji = mean(gaji),
+   rata_rata_pengeluaran = mean(pengeluaran))
Source: local data frame [5 x 8]
Groups: pendidikan [?]
  pendidikan jenis_kelamin minimum_gaji minimum_pengeluaran maksimum_gaji maksimum_pengeluaran rata_rata_gaji rata_rata_pengeluaran
1          S1      laki-laki          3.2              2.8          5.5              3.5          4.433333          3.233333
2          S1      perempuan          4.9              3.1          5.5              3.3          5.200000          3.400000
3          S2      laki-laki          9.9              3.5          9.9              3.5          9.900000          3.900000
4          S2      laki-laki          6.5              4.4          6.5              4.4          6.500000          4.400000
5          S2      perempuan          5.9              3.9          8.7              4.1          7.033333          4.033333

```

Perhatikan bahwa perhitungan tersebut menggunakan fungsi *summarise()* yang mana terlebih dahulu memasang (*attach*) *package dplyr*

Gambar 9.13

BAB 10

STRUKTUR PEMROGRAMAN R

Norman Matloff (2009:67) dalam bukunya “*The Art of R Programming*” menyatakan sebagai berikut.

“R is a full programming language, similar to scripting languages such as Perl and Python. One can define functions, use constructs such as loops and conditionals, etc.

R is also block-structured, in a manner similar to those of the above languages, as well as C. Blocks are delineated by braces, though they are optional if the block consists of just a single statement.”

Jadi, pernyataan di atas memberikan informasi bahwa R merupakan bahasa pemrograman, sama seperti bahasa pemrograman *Perl* dan *Python*. Dalam R, dapat mendefinisikan suatu fungsi, menggunakan konstruk-konstruk seperti pengulangan (*loops*) dan kondisi (*conditionals*), dan sebagainya. R juga merupakan *block-structured*, mirip dengan bahasa pemrograman *Perl* dan *Python*, dan juga C. Suatu blok diawali dengan buka kurawal (*braces*) dan diakhiri tutup kurawal, meskipun hal tersebut bersifat opsional jika dalam blok tersebut hanya terdiri dari satu pernyataan.

10.1 Pengulangan (*Loop*) dengan *for* dan Kondisi (*Condition*) dengan *if*

Gambar 10.1 diberikan ilustrasi mengenai pengulangan dalam R.

```
Console ~/
> x <- c(1,2,3,4,5)
> for(n in x)
+ {
+   print(n^2)
+   print("hai")
+ }
[1] 1
[1] "hai"
[1] 4
[1] "hai"
[1] 9
[1] "hai"
[1] 16
[1] "hai"
[1] 25
[1] "hai"
> |
```

Gambar 10.1

Berdasarkan Gambar 10.1, kode R

```
for (n in x)
{
print(n^2)
print("hai")
}
```

merupakan contoh dari **pengulangan** (*loop*). Diketahui vektor **x** menyimpan bilangan 1, 2, 3, 4, dan 5. Bilangan-bilangan tersebut kemudian dikuadratkan, sehingga menghasilkan 1, 4, 9, 16, dan 25. Kode R

```
print(n^2)
```

bertujuan untuk mengkuadratkan bilangan-bilangan dalam *n*.

- ⇒ Pada iterasi pertama, $n = 1$, maka $n^2 = 1$.
- ⇒ Pada iterasi kedua, $n = 2$, maka $n^2 = 4$.
- ⇒ Pada iterasi ketiga, $n = 3$, maka $n^2 = 9$.
- ⇒ Pada iterasi keempat, $n = 4$, maka $n^2 = 16$.
- ⇒ Pada iterasi kelima, $n = 5$, maka $n^2 = 25$.

Perhatikan juga bahwa tulisan “hai” tercetak sebanyak 5 kali (hal ini menandakan terdapat 5 kali pengulangan).

Informasi:

Fungsi **print()** digunakan untuk mencetak.

10.2 Contoh Ke-2: Pengulangan (*Loop*) dengan *for* dan Kondisi (*Condition*) dengan *if*

Gambar 10.2 diberikan contoh lagi mengenai pengulangan dalam R. Berdasarkan Gambar 10.2, kode R

```
x <- c(3,4,5,100,2)
for(n in x)
{
  if(n%%2==0)
  {
    cat(sprintf("%d adalah bilangan genap \n",n))
  }
  else
  {
    cat(sprintf("%d adalah bilangan ganjil \n",n))
  }
}
```

merupakan contoh dari pengulangan yang juga melibatkan suatu kondisi.

- ⇒ Kondisi pertama, jika suatu bilangan dibagi 2, memiliki nilai sisa sama dengan 0, maka bilangan tersebut adalah bilangan genap, yakni dinyatakan dengan

`if(n%%2==0)`

- ⇒ Kondisi kedua, jika suatu bilangan dibagi 2, memiliki nilai sisa tidak sama 0, maka bilangan tersebut adalah bilangan ganjil.

```
Console ~/ /
> x <- c(3,4,5,100,2)
> for(n in x)
+ {
+   if(n%%2==0)
+   {
+     cat(sprintf("%d adalah bilangan genap \n",n))
+   }
+   else
+   {
+     cat(sprintf("%d adalah bilangan ganjil \n",n))
+   }
+ }
3 adalah bilangan ganjil
4 adalah bilangan genap
5 adalah bilangan ganjil
100 adalah bilangan genap
2 adalah bilangan genap
>
```

Gambar 10.2

Diketahui vektor **x** menyimpan bilangan 3, 4, 5, 100, dan 2.

- ⇒ Pada iterasi pertama, $n = 3$. Nilai 3 dibagi 2 memiliki nilai sisa 1 (tidak sama dengan nol), maka 3 adalah bilangan ganjil.
- ⇒ Pada iterasi kedua, $n = 4$. Nilai 4 dibagi 2 memiliki nilai sisa 0, maka 4 adalah bilangan genap.
- ⇒ Pada iterasi ketiga, $n = 5$. Nilai 5 dibagi 2 memiliki nilai sisa 1 (tidak sama dengan nol), maka 5 adalah bilangan ganjil.
- ⇒ Pada iterasi keempat, $n = 100$. Nilai 100 dibagi 2 memiliki nilai sisa 0, maka 100 adalah bilangan genap.
- ⇒ Pada iterasi kelima, $n = 2$. Nilai 2 dibagi 2 memiliki nilai sisa 0, maka 2 adalah bilangan genap.

Informasi:

- ⇒ Fungsi **`cat(sprintf())`** digunakan untuk mencetak.
- ⇒ Perhatikan bahwa **`%d`** digunakan untuk menyatakan bilangan bulat.

BAB 11

FUNGSI

11.1 Membuat Fungsi Sederhana dalam R dan Memanggil Fungsi

Berikut diberikan contoh kode R terkait pembuatan fungsi.

Gambar 11.1

Pada kode R Gambar 11.1, dibentuk suatu fungsi bernama **panggil**.

Suatu fungsi memiliki nama. Fungsi yang dibentuk pada Gambar 11.1 bernama **panggil**.

Berdasarkan kode R Gambar 11.1, fungsi bernama **panggil** dipanggil sebanyak 4 kali (Perhatikan Gambar 11.2).

Gambar 11.2

Fungsi bernama **panggil**, dipanggil sebanyak 4 kali. Setiap pemanggilan fungsi tersebut, akan mencetak tulisan **Hai**. Karena fungsi bernama **panggil** dipanggil sebanyak 4 kali, maka fungsi bernama **panggil** mencetak tulisan **Hai** sebanyak 4 kali. Dimulai dari buka kurawal "{", sampai dengan tutup kurawal "}" disebut **tubuh fungsi** (*body of function*) (Gambar 11.3).

```
+ {  
+ print("Hai")  
+ }
```

Gambar 11.3 Tubuh Fungsi dari Fungsi Panggil

Tubuh fungsi dimulai dari buka kurawal "{", sampai dengan tutup kurawal "}".

Setelah suatu fungsi dibentuk/dibuat, maka suatu fungsi dapat dipanggil. Berikut sintaks untuk memanggil suatu fungsi yang telah dibentuk.

nama_fungsi()

Jadi untuk memanggil suatu fungsi yang telah dibentuk/dibuat, maka perlu disebutkan nama fungsi, kemudian ditambah buka kurung "(" dan tutup kurung ")".

Jadi untuk memanggil suatu fungsi yang telah dibentuk/dibuat, **maka perlu disebutkan nama fungsi**, kemudian ditambah buka kurung "(" dan tutup kurung ")".

11.2 Contoh Fungsi yang Melibatkan Dua Argumen

Berikut diberikan contoh kode R yang melibatkan dua argumen dalam suatu fungsi (Gambar 11.4).

```

Console ~/
> hitung <- function(x,y)
+ {
+   cat(sprintf("Penjumlahan %d + %d = %d\n",x,y,x+y))
+   cat(sprintf("Pengurangan %d - %d = %d\n",x,y,x-y))
+   cat(sprintf("Pembagian %d / %d = %f\n",x,y,x/y))
+   cat(sprintf("Perkalian %d * %d = %d\n",x,y,x*y))
+ }
>
> hitung(5,2)
Penjumlahan 5 + 2 = 7
Pengurangan 5 - 2 = 3
Pembagian 5 / 2 = 2.500000
Perkalian 5 * 2 = 10
> hitung(10,5)
Penjumlahan 10 + 5 = 15
Pengurangan 10 - 5 = 5
Pembagian 10 / 5 = 2.000000
Perkalian 10 * 5 = 50
> |

```

Fungsi bernama **hitung** dipanggil sebanyak 2 kali, yakni **hitung(5,2)** untuk pemanggilan pertama, dan **hitung(10,5)** untuk pemanggilan yang kedua.

Gambar 11.4

Berdasarkan Gambar 11.4, dibentuk suatu fungsi bernama **hitung** yang memiliki argumen sebanyak 2, yakni **x** dan **y**.

Suatu fungsi **memiliki nama**. Fungsi yang dibentuk pada Gambar 11.4 bernama **hitung**, memiliki argumen sebanyak 2, yakni **x** dan **y**.

Berdasarkan kode R pada Gambar 11.4, fungsi bernama **hitung** dipanggil sebanyak 2 kali (Perhatikan Gambar 11.4).

Pada fungsi bernama **hitung**, kode R

cat(sprintf("Penjumlahan %d + %d = %d \n", x, y, x+y))

bertujuan untuk menjumlahkan dua buah bilangan. Perhatikan bahwa dalam sintaks tersebut terdapat “%d” yang berarti untuk menyatakan bilangan bulat.

cat(sprintf("Penjumlahan %d + %d = %d \n", x, y, x+y))

Pada fungsi bernama **hitung**, kode R

cat(sprintf("Pengurangan %d - %d = %d \n", x, y, x-y))

bertujuan untuk mengurangi dua buah bilangan. Perhatikan bahwa dalam sintaks tersebut terdapat “%d” yang berarti untuk menyatakan bilangan bulat.

```
cat(sprintf("Penjumlahan %d - %d = %d \n", x, y, x-y))
```

Pada fungsi bernama hitung, kode R

```
cat(sprintf("Pembagian %d / %d = %f \n", x, y, x/y))
```

bertujuan untuk pembagian dua buah bilangan. Perhatikan bahwa dalam sintaks tersebut terdapat “%d” yang berarti untuk menyatakan bilangan bulat, sementara “%f” untuk menyatakan bilangan pecahan.

```
cat(sprintf("Penjumlahan %d / %d = %f \n", x, y, x/y))
```

Perhatikan bahwa %f digunakan karena hasil pembagian dari dua buah bilangan, yakni x/y dapat berupa **bilangan pecahan**.

Kode R

```
hitung(5,2)
```

berarti memanggil fungsi bernama **hitung**, kemudian nilai **x** diisi dengan 5, sementara nilai **y** diisi dengan 2.

$$5 + 2 = 7$$

```
cat(sprintf("Penjumlahan %d + %d = %d \n", x, y, x+y))
```

$$5 - 2 = 3$$

```
cat(sprintf("Pengurangan %d - %d = %d \n", x, y, x-y))
```

$$5 / 2 = 2.5$$

```
cat(sprintf("Pembagian %d / %d = %f \n", x, y, x/y))
```

$$5 * 2 = 10$$

```
cat(sprintf("Perkalian %d * %d = %d \n", x, y, x*y))
```

11.3 Contoh Fungsi yang Melibatkan Argumen Vektor

Berikut diberikan contoh kode R yang melibatkan argumen vektor dalam suatu fungsi (Gambar 11.5).

```
Console ~/
> kuadrat <- function(x)
+ {
+   print(x^2)
+ }
>
> a <- c(1,4,2,6)
> kuadrat(a)
[1] 1 16 4 36
> |
```

Fungsi bernama **kuadrat** dipanggil sebanyak 1 kali, yakni **hitung(a)**.

Gambar 11.5

Pada kode R Gambar 11.5, dibentuk suatu fungsi bernama **kuadrat** yang memiliki argumen sebanyak 1, yakni **x**.

Suatu fungsi **memiliki nama**. Fungsi yang dibentuk pada Gambar 11.5 bernama **kuadrat**, memiliki argumen sebanyak 1, yakni **x**.

Berdasarkan kode R Gambar 11.5, fungsi bernama **kuadrat** dipanggil sebanyak 1 kali.

11.4 Contoh Fungsi yang Melibatkan Pengembalian Nilai (*Return Value*)

Berikut diberikan contoh kode R yang melibatkan fungsi dengan pengembalian nilai (Gambar 11.6).

```
Console ~/
> kuadrat <- function(x)
+ {
+   return(x^2)
+ }
>
> a <- c(1,4,2,6)
> a
[1] 1 4 2 6
> a <- kuadrat(a)
> a
[1] 1 16 4 36
> |
```

Gambar 11.6

Pada kode R Gambar 11.6, dibentuk suatu fungsi bernama **kuadrat** yang memiliki argumen sebanyak 1, yakni **x**.

Suatu fungsi memiliki nama. Fungsi yang dibentuk pada Gambar 11.6 bernama **kuadrat**, memiliki argumen sebanyak 1, yakni **x**.

Berdasarkan kode R Gambar 11.6, fungsi bernama **kuadrat** dipanggil sebanyak 1 kali.

BAB 12

MEMANIPULASI DATA DENGAN FUNGSI DPLYR

Pada Bab 12 ini, disajikan langkah demi langkah mengenai memanipulasi data dengan fungsi *dplyr* (). Sebelum menggunakan fungsi *dplyr* (), terlebih dahulu telah diinstal *package dplyr* dan mengaktifkan *package dplyr* (perhatikan Gambar 12.1 dan Gambar 12.2).

Gambar 12.1

Gambar 12.2

Praktek Ke-1: Menggunakan Fungsi *filter()*

```
> Nama <- c("Ugi", "Egi", "Andi", "Udin", "Boy", "Sinta", "Rini", "Rani", "Fitri", "Dini")
> Jenis_Kelamin <- c("Laki-Laki", "Laki-Laki", "Laki-Laki", "Laki-Laki", "Laki-Laki", "Perempuan", "Perempuan", "Perempuan", "Perempuan", "Perempuan")
> Pendidikan <- c("S1", "S2", "S3", "S1", "S2", "S2", "S1", "S3", "S2")
> Jumlah_Bersaudara <- c(2, 3, 1, 2, 5, 2, 4, 2, 3, 5)
> Data1 <- data.frame(Nama, Jenis_Kelamin, Pendidikan, Jumlah_Bersaudara)
> Data1
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1  Ugi      Laki-Laki         S1                2
2  Egi      Laki-Laki         S2                3
3  Andi     Laki-Laki         S3                1
4  Udin     Laki-Laki         S1                2
5  Boy       Laki-Laki         S2                5
6  Sinta    Perempuan         S2                2
7  Rini     Perempuan         S2                4
8  Rani     Perempuan         S1                2
9  Fitri    Perempuan         S3                3
10 Dini     Perempuan         S2                5
> library(dplyr)
> filter(Data1, Jenis_Kelamin=="Laki-Laki")
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1  Ugi      Laki-Laki         S1                2
2  Egi      Laki-Laki         S2                3
3  Andi     Laki-Laki         S3                1
4  Udin     Laki-Laki         S1                2
5  Boy       Laki-Laki         S2                5
> Data1 %>%
+ filter(Jenis_Kelamin=="Laki-Laki")
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1  Ugi      Laki-Laki         S1                2
2  Egi      Laki-Laki         S2                3
3  Andi     Laki-Laki         S3                1
4  Udin     Laki-Laki         S1                2
5  Boy       Laki-Laki         S2                5
```

Aktifkan/attach package dplyr.

Memilih responden yang berjenis kelamin laki-laki.

Gambar 12.1

Berdasarkan data Gambar 12.1:

- ⇒ Terdapat 4 kolom atau variabel, yakni **Nama**, **Jenis_Kelamin**, **Pendidikan**, dan **Jumlah_Bersaudara**.
- ⇒ Jumlah responden yang diamati sebanyak 10 responden.
- ⇒ Sebagai contoh responden ke-1, bernama Ugi, dengan jenis kelamin laki-laki, pendidikan S1, jumlah bersaudara 2. Responden ke-10 bernama Dini, dengan jenis kelamin perempuan, pendidikan S2, jumlah bersaudara 5.

"Tugas:

- ⇒ **Gunakan fungsi *filter()* untuk memilih hanya responden berjenis kelamin laki-laki.**
- ⇒ **Kemudian tampilkan informasi dari variabel nama, jenis kelamin, pendidikan, dan jumlah bersaudara."**

Perlu diperhatikan bahwa sebelum menggunakan fungsi *filter()*, terlebih dahulu memasang (*attach*) package **dplyr**. Fungsi *filter()* tersedia dalam package **dplyr**.

Perintah R

```
library(dplyr)
filter(Data1, Jenis_Kelamin=="Laki-Laki")
```

Alternatif Perintah R

```
Data1 %>%
filter(Jenis_Kelamin=="Laki-Laki")
```

Praktek Ke-2: Menggunakan Fungsi *filter()*

```
Console ~/ /
> Data1
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1   Ugi      Laki-Laki      S1           2
2   Egi      Laki-Laki      S2           3
3   Andi     Laki-Laki      S3           1
4   Udin     Laki-Laki      S1           2
5   Boy      Laki-Laki      S2           5
6   Sinta    Perempuan      S2           2
7   Rini     Perempuan      S2           4
8   Rani     Perempuan      S1           2
9   Fitri    Perempuan      S3           3
10  Dini     Perempuan      S2           5
> filter(Data1, Pendidikan=="S2" | Pendidikan=="S3")
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1   Egi      Laki-Laki      S2           3
2   Andi     Laki-Laki      S3           1
3   Boy      Laki-Laki      S2           5
4   Sinta    Perempuan      S2           2
5   Rini     Perempuan      S2           4
6   Fitri    Perempuan      S3           3
7   Dini     Perempuan      S2           5
> Data1 %>%
+ filter(Pendidikan=="S2" | Pendidikan=="S3")
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1   Egi      Laki-Laki      S2           3
2   Andi     Laki-Laki      S3           1
3   Boy      Laki-Laki      S2           5
4   Sinta    Perempuan      S2           2
5   Rini     Perempuan      S2           4
6   Fitri    Perempuan      S3           3
7   Dini     Perempuan      S2           5
> |
```

Memilih responden dengan pendidikan S2 atau S3.

Gambar 12.2

"Tugas:

- ⇒ Gunakan fungsi *filter()* untuk memilih responden dengan pendidikan S2 atau S3.
- ⇒ Kemudian tampilkan informasi dari variabel nama, jenis kelamin, pendidikan, dan jumlah bersaudara."

Perhatikan bahwa simbol "|" berarti "atau".

Perintah R

```
filter(Data1, Pendidikan=="S2" | Pendidikan=="S3")
```

Alternatif Perintah R

```
Data1 %>%
filter(Pendidikan=="S2" | Pendidikan=="S3")
```


Praktek Ke-3: Menggunakan Fungsi *names()* dan *colnames()*

```
Source
Console ~/
> Data1
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1   Ugi      Laki-Laki        S1                2
2   Egi      Laki-Laki        S2                3
3  Andi      Laki-Laki        S3                1
4  Udin      Laki-Laki        S1                2
5   Boy      Laki-Laki        S2                5
6  Sinta    Perempuan        S2                2
7   Rini    Perempuan        S2                4
8   Rani    Perempuan        S1                2
9  Fitri    Perempuan        S3                3
10 Dini     Perempuan        S2                5
> colnames(Data1)
[1] "Nama"          "Jenis_Kelamin" "Pendidikan"    "Jumlah_Bersaudara"
> names(Data1)
[1] "Nama"          "Jenis_Kelamin" "Pendidikan"    "Jumlah_Bersaudara"
>
```

Gambar 12.3

"Tugas:

Gunakan fungsi *colnames()* atau *names()* untuk menampilkan nama-nama kolom atau variabel pada *data frame* *Data1*."

Berdasarkan Gambar 12.3, diketahui terdapat 4 kolom atau variabel dengan nama, **Nama**, **Jenis_Kelamin**, **Pendidikan**, dan **Jumlah_Bersaudara**. Fungsi *names()* dan *colnames()* dapat digunakan untuk menampilkan nama-nama dari variabel atau kolom pada *data frame*.

Perintah R

```
colnames(Data1)
```

Alternatif Perintah R

```
names(Data1)
```

Praktek Ke-4: Menggunakan Fungsi *length()*

```
Console ~/
> Data1
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1  Ugi      Laki-Laki      S1             2
2  Egi      Laki-Laki      S2             3
3  Andi     Laki-Laki      S3             1
4  Udin     Laki-Laki      S1             2
5  Boy      Laki-Laki      S2             5
6  Sinta    Perempuan     S2             2
7  Rini     Perempuan     S2             4
8  Rani     Perempuan     S1             2
9  Fitri    Perempuan     S3             3
10 Dini     Perempuan     S2             5
> length(Data1)
[1] 4
> |
```

Gambar 12.4

"Tugas:

Gunakan fungsi *length()* untuk mengetahui jumlah variabel atau kolom pada *data frame* *Data1*."

Berdasarkan Gambar 12.4, diketahui jumlah kolom atau variabel sebanyak 4, yakni **Nama**, **Jenis_Kelamin**, **Pendidikan**, dan **Jumlah_Bersaudara**. Fungsi *length()* dapat digunakan untuk mengetahui jumlah kolom atau variabel pada *data frame*.

Perintah R

length(Data1)

Praktek Ke-5: Menggunakan Fungsi *length()*

```
Console ~/
> Data1
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1  Ugi      Laki-Laki         S1              2
2  Egi      Laki-Laki         S2              3
3  Andi     Laki-Laki         S3              1
4  Udin     Laki-Laki         S1              2
5  Boy      Laki-Laki         S2              5
6  Sinta    Perempuan         S2              2
7  Rini     Perempuan         S2              4
8  Rani     Perempuan         S1              2
9  Fitri    Perempuan         S3              3
10 Dini     Perempuan         S2              5
> length(Data1[,1]) #Mengetahui jumlah baris pada kolom pertama
[1] 10
> length(Data1[,2]) #Mengetahui jumlah baris pada kolom kedua
[1] 10
> |
```

Gambar 12.5

"Tugas:

Gunakan fungsi *length()* untuk mengetahui jumlah baris dari masing-masing kolom atau variabel pada data frame *Data1*."

Berdasarkan Gambar 12.5, digunakan fungsi *length()* untuk mengetahui jumlah baris dari masing-masing kolom.

Perintah R

```
length(Data1[,1]) #Mengetahui jumlah baris pada kolom pertama
```

```
length(Data1[,2]) #Mengetahui jumlah baris pada kolom kedua
```

Praktek Ke-6: Menggunakan Fungsi *length()* dan *na.omit()*

```
> Nama <- c("Ugi", "Egi", "Andi", "Udin", "Boy", "Sinta", "Rini", "Rani", "Fitri", "Dini")
> Jenis_kelamin <- c("Laki-Laki", "Laki-Laki", "Laki-Laki", "Laki-Laki", "Laki-Laki", "Perempuan", "Perempuan", "Perempuan", "Perempuan")
> Nilai_Matematika <- c(75, NA, 45, 40, 50, 75, 70, 80, 90, 80)
> Nilai_Kimia <- c(50, 70, 75, NA, 50, 85, NA, 90, NA, 70)
> Nilai_Biologi <- c(NA, 50, NA, NA, 75, 80, 40, 80, NA, 90)
> Data1 <- data.frame(Nama, Jenis_kelamin, Nilai_Matematika, Nilai_Kimia, Nilai_Biologi)
> Data1
  Nama Jenis_kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi
1  Ugi      Laki-Laki             75           50           NA
2  Egi      Laki-Laki             NA           70           50
3  Andi     Laki-Laki             45           75           NA
4  Udin     Laki-Laki             40           NA           NA
5  Boy      Laki-Laki             50           50           75
6  Sinta    Perempuan             75           85           80
7  Rini     Perempuan             70           NA           40
8  Rani     Perempuan             80           90           80
9  Fitri    Perempuan             90           NA           NA
10 Dini     Perempuan             80           70           90
> length(Data1[,3])
[1] 10
> length(na.omit(Data1[,3]))
[1] 9
> length(Data1[,4])
[1] 10
> length(na.omit(Data1[,4]))
[1] 7
> length(Data1[,5])
[1] 10
> length(na.omit(Data1[,5]))
[1] 6
> |
```

Gambar 12.6

"Tugas:

Gunakan fungsi *length()* dan *na.omit()* untuk mengetahui jumlah baris yang tidak kosong (NA) dari masing-masing variabel atau kolom pada *data frame* Data1."

Berdasarkan Gambar 12.6:

- ⇒ Diketahui jumlah responden yang ada nilai matematika-nya sebanyak 9. NA berarti *not available*, yang berarti tidak ada.
- ⇒ Diketahui jumlah responden yang ada nilai kimia-nya sebanyak 7.
- ⇒ Diketahui jumlah responden yang ada nilai biologi-nya sebanyak 6.

Penggunaan fungsi *na.omit()* akan mengeliminasi NA.

Perintah R

```
length(na.omit(Data1[,3])) #Mengetahui jumlah responden yang ada nilai matematikanya
```

```
length(na.omit(Data1[,4])) #Mengetahui jumlah responden yang ada nilai kimianya
```

```
length(na.omit(Data1[,5])) #Mengetahui jumlah responden yang ada nilai biologinya
```

Praktek Ke-7: Menggunakan Fungsi `dim()`

```
Source
Console ~/
> Nama <- c("Ugi", "Egi", "Andi", "Udin", "Boy", "Sinta", "Rini", "Rani", "Fitri", "Dini")
> Jenis_kelamin <- c("Laki-Laki", "Laki-Laki", "Laki-Laki", "Laki-Laki", "Laki-Laki", "Perempuan", "Perempuan", "Perempuan", "Perempuan", "Perempuan")
> Nilai_Matematika <- c(75, 60, 45, 40, 50, 75, 70, 80, 90, 80)
> Nilai_kimia <- c(50, 70, 75, 50, 50, 85, 60, 90, 50, 70)
> Nilai_Biologi <- c(70, 50, 80, 80, 75, 80, 40, 80, 75, 90)
> Data1 <- data.frame(Nama, Jenis_kelamin, Nilai_Matematika, Nilai_Kimia, Nilai_Biologi)
> Data1
  Nama Jenis_kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi
1  Ugi      Laki-Laki             75           50             70
2  Egi      Laki-Laki             60           70             50
3  Andi     Laki-Laki             45           75             80
4  Udin     Laki-Laki             40           50             80
5  Boy      Laki-Laki             50           50             75
6  Sinta    Perempuan             75           85             80
7  Rini     Perempuan             70           60             40
8  Rani     Perempuan             80           90             80
9  Fitri    Perempuan             90           50             75
10 Dini     Perempuan             80           70             90
> dim(Data1)
[1] 10  5
> |
```

Gambar 12.7

"Tugas:

Gunakan fungsi `dim()` untuk mengetahui jumlah baris dan kolom pada *data frame* `Data1`."

Berdasarkan Gambar 12.7, diketahui jumlah baris (responden) sebanyak 10 dan jumlah kolom sebanyak 5. Fungsi `dim()` dapat digunakan untuk mengetahui jumlah baris dan kolom pada *data frame*.

Perintah R

```
dim(Data1)
```

Praktek Ke-8: Menggunakan Fungsi *factor()* dan Fungsi *levels()*

```
Console ~/
> Data1
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1   Ugi      Laki-Laki         S1                2
2   Egi      Laki-Laki         S2                3
3   Andi     Laki-Laki         S3                1
4   Udin     Laki-Laki         S1                2
5   Boy      Laki-Laki         S2                5
6   Sinta    Perempuan        S2                2
7   Rini     Perempuan        S2                4
8   Rani     Perempuan        S1                2
9   Fitri    Perempuan        S3                3
10  Dini     Perempuan        S2                5
> levels(factor(Data1[,2]))
[1] "Laki-Laki" "Perempuan"
> levels(factor(Data1[,3]))
[1] "S1" "S2" "S3"
> levels(factor(Data1[,4]))
[1] "1" "2" "3" "4" "5"
> |
```

Gambar 12.8

"Tugas:

Gunakan fungsi *levels()* dan *factor()* untuk mengetahui *level* dari suatu variabel."

Berdasarkan Gambar 12.8, diketahui variabel **Jenis_Kelamin** terdiri dari 2 *level*, yakni "Laki-Laki" dan "Perempuan". Diketahui variabel **Pendidikan** terdiri dari 3 *level*, yakni "S1", "S2", dan "S3". Pada variabel **Jumlah_Bersaudara** terdiri dari 5 *level*, yakni 1, 2, 3, 4, dan 5. Kombinasi dari fungsi *factor()* dan *levels()* dapat digunakan untuk menampilkan *level* dari suatu vektor.

Perintah R

```
levels(factor(Data1[,2]))
```

```
levels(factor(Data1[,3]))
```

```
levels(factor(Data1[,4]))
```

Praktek Ke-9: Menggunakan Fungsi *filter()*

```
Console ~/
> Data1
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1  Ugi      Laki-Laki      S1             2
2  Egi      Laki-Laki      S2             3
3  Andi     Laki-Laki      S3             1
4  Udin     Laki-Laki      S1             2
5  Boy      Laki-Laki      S2             5
6  Sinta    Perempuan     S2             2
7  Rini     Perempuan     S2             4
8  Rani     Perempuan     S1             2
9  Fitri    Perempuan     S3             3
10 Dini     Perempuan     S2             5
> filter(Data1, Jenis_Kelamin=="Perempuan" & Pendidikan=="S2")
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1 Sinta    Perempuan     S2             2
2 Rini     Perempuan     S2             4
3 Dini     Perempuan     S2             5
> Data1 %>%
+ filter(Jenis_Kelamin=="Perempuan",Pendidikan=="S2")
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1 Sinta    Perempuan     S2             2
2 Rini     Perempuan     S2             4
3 Dini     Perempuan     S2             5
> |
```

Gambar 12.9

"Tugas:

- ⇒ Gunakan fungsi *filter()* untuk memilih responden berjenis kelamin perempuan dan berpendidikan S2.
- ⇒ Kemudian tampilkan informasi dari variabel nama, jenis kelamin, pendidikan, dan jumlah bersaudara."

Simbol "&" berarti "dan".

Perintah R

```
filter(Data1, Jenis_Kelamin=="Perempuan" & Pendidikan=="S2")
```

Alternatif Perintah R

```
Data1 %>%
filter(Jenis_Kelamin=="Perempuan",Pendidikan=="S2")
```

Praktek Ke-10: Menggunakan Fungsi *filter()*

```
Console ~/
> Data1
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1  Ugi      Laki-Laki      S1           2
2  Egi      Laki-Laki      S2           3
3  Andi     Laki-Laki      S3           1
4  Udin     Laki-Laki      S1           2
5  Boy      Laki-Laki      S2           5
6  Sinta    Perempuan     S2           2
7  Rini     Perempuan     S2           4
8  Rani     Perempuan     S1           2
9  Fitri    Perempuan     S3           3
10 Dini     Perempuan     S2           5
> filter(Data1, Jenis_Kelamin=="Perempuan" & (Pendidikan=="S2" | Pendidikan=="S3") & Jumlah_Bersaudara > 2 )
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1  Rini     Perempuan     S2           4
2  Fitri    Perempuan     S3           3
3  Dini     Perempuan     S2           5
> Data1 %>%
+ filter(Jenis_Kelamin=="Perempuan" & (Pendidikan=="S2" | Pendidikan=="S3") & Jumlah_Bersaudara > 2 )
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1  Rini     Perempuan     S2           4
2  Fitri    Perempuan     S3           3
3  Dini     Perempuan     S2           5
> |
```

Gambar 12.10

"Tugas:

- ⇒ Gunakan fungsi *filter()* untuk memilih responden berjenis kelamin perempuan, berpendidikan S2 atau S3, dengan jumlah bersaudara di atas 2.
- ⇒ Kemudian tampilkan informasi dari variabel nama, jenis kelamin, pendidikan, dan jumlah bersaudara."

Perintah R

```
filter(Data1, Jenis_Kelamin=="Perempuan" & (Pendidikan=="S2" | Pendidikan=="S3") & Jumlah_Bersaudara > 2 )
```

Alternatif Perintah R

```
Data1 %>%
filter(Jenis_Kelamin=="Perempuan" & (Pendidikan=="S2" | Pendidikan=="S3") & Jumlah_Bersaudara > 2 )
```


Praktek Ke-11: Menggunakan Fungsi `select()`

The screenshot shows an R console window with the following content:

```
> Data1
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1  Ugi      Laki-Laki         S1              2
2  Egi      Laki-Laki         S2              3
3  Andi     Laki-Laki         S3              1
4  Udin     Laki-Laki         S1              2
5  Boy      Laki-Laki         S2              5
6  Sinta    Perempuan        S2              2
7  Rini     Perempuan        S2              4
8  Rani     Perempuan        S1              2
9  Fitri    Perempuan        S3              3
10 Dini     Perempuan        S2              5
> select(Data1, Nama)
  Nama
1  Ugi
2  Egi
3  Andi
4  Udin
5  Boy
6  Sinta
7  Rini
8  Rani
9  Fitri
10 Dini
> Data1 %>%
+ select(Nama)
  Nama
1  Ugi
2  Egi
3  Andi
4  Udin
5  Boy
6  Sinta
7  Rini
8  Rani
9  Fitri
10 Dini
> |
```

Gambar 12.11

"Tugas:

- ⇒ Gunakan fungsi `select()` untuk menampilkan informasi hanya pada variabel nama."

Pada Gambar 12.11, digunakan fungsi `select()` untuk hanya menampilkan variabel nama.

Perintah R

```
select(Data1, Nama)
```

Alternatif Perintah R

```
Data1 %>%
select(Nama)
```

Praktek Ke-12: Menggunakan Fungsi *select()*

```
Console ~/ /
> Data1
  Nama Jenis_kelamin Pendidikan Jumlah_Bersaudara
1  Ugi      Laki-Laki         S1              2
2  Egi      Laki-Laki         S2              3
3  Andi     Laki-Laki         S3              1
4  Udin     Laki-Laki         S1              2
5  Boy      Laki-Laki         S2              5
6  Sinta    Perempuan        S2              2
7  Rini     Perempuan        S2              4
8  Rani     Perempuan        S1              2
9  Fitri    Perempuan        S3              3
10 Dini     Perempuan        S2              5
> select(Data1, Nama, Pendidikan)
  Nama Pendidikan
1  Ugi         S1
2  Egi         S2
3  Andi        S3
4  Udin        S1
5  Boy         S2
6  Sinta       S2
7  Rini        S2
8  Rani        S1
9  Fitri       S3
10 Dini        S2
> Data1 %>% select(Nama, Pendidikan)
  Nama Pendidikan
1  Ugi         S1
2  Egi         S2
3  Andi        S3
4  Udin        S1
5  Boy         S2
6  Sinta       S2
7  Rini        S2
8  Rani        S1
9  Fitri       S3
10 Dini        S2
>
```

Gambar 12.12

"Tugas:

- ⇒ Gunakan fungsi *select()* untuk menampilkan informasi hanya pada variabel **Nama** dan **pendidikan**."

Berdasarkan Gambar 12.12, digunakan fungsi *select()*, untuk hanya menampilkan variabel **Nama** dan **Pendidikan**.

Perintah R

```
select(Data1, Nama, Pendidikan)
```

Alternatif Perintah R

```
Data1 %>% select(Nama, Pendidikan)
```

Praktek Ke-13: Kombinasi Penggunaan Fungsi *filter()* dan *select()*

```
Console ~/
> Data1
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1   Ugi      Laki-Laki        S1              2
2   Egi      Laki-Laki        S2              3
3   Andi     Laki-Laki        S3              1
4   Udin     Laki-Laki        S1              2
5   Boy      Laki-Laki        S2              5
6   Sinta    Perempuan        S2              2
7   Rini     Perempuan        S2              4
8   Rani     Perempuan        S1              2
9   Fitri    Perempuan        S3              3
10  Dini     Perempuan        S2              5
> filter(select(Data1, Nama, Pendidikan), Pendidikan=="S1")
  Nama Pendidikan
1   Ugi         S1
2  Udin         S1
3  Rani         S1
> Data1 %>%
+ filter(Pendidikan=="S1") %>%
+ select(Nama, Pendidikan)
  Nama Pendidikan
1   Ugi         S1
2  Udin         S1
3  Rani         S1
> |
```

Gambar 12.13

"Tugas:

- ⇒ Gunakan fungsi *filter()* untuk memilih responden berpendidikan S1.
- ⇒ Gunakan fungsi *select()* untuk menampilkan informasi hanya pada variabel nama dan pendidikan."

Perintah R

```
filter(select(Data1, Nama, Pendidikan), Pendidikan=="S1")
```

Alternatif Perintah R

```
Data1 %>%
filter(Pendidikan=="S1") %>%
select(Nama, Pendidikan)
```

Praktek Ke-14: Penggunaan Fungsi *filter()*, *select()*, dan *arrange()*

```
Console ~/
> Data1
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1  Ugi      Laki-Laki         S1              2
2  Egi      Laki-Laki         S2              3
3  Andi     Laki-Laki         S3              1
4  Udin     Laki-Laki         S1              2
5  Boy      Laki-Laki         S2              5
6  Sinta    Perempuan        S2              2
7  Rini     Perempuan        S2              4
8  Rani     Perempuan        S1              2
9  Fitri    Perempuan        S3              3
10 Dini     Perempuan        S2              5
> arrange(filter(select(Data1, Nama, Pendidikan, Jumlah_Bersaudara), Pendidikan=="S2" | Pendidikan=="S3"), desc(Jumlah_Bersaudara))
  Nama Pendidikan Jumlah_Bersaudara
1  Boy      S2              5
2  Dini     S2              5
3  Rini     S2              4
4  Egi      S2              3
5  Fitri    S3              3
6  Sinta    S2              2
7  Andi     S3              1
> Data1 %>%
+ filter(Pendidikan=="S2" | Pendidikan=="S3") %>%
+ select(Nama, Pendidikan, Jumlah_Bersaudara) %>%
+ arrange(desc(Jumlah_Bersaudara))
  Nama Pendidikan Jumlah_Bersaudara
1  Boy      S2              5
2  Dini     S2              5
3  Rini     S2              4
4  Egi      S2              3
5  Fitri    S3              3
6  Sinta    S2              2
7  Andi     S3              1
> |
```

Gambar 12.14

"Tugas:

- ⇒ Gunakan fungsi *filter()* untuk memilih responden berpendidikan S2 atau S3.
- ⇒ Gunakan fungsi *select()* untuk menampilkan informasi hanya pada variabel nama, pendidikan, dan jumlah bersaudara.
- ⇒ Gunakan fungsi *arrange()* untuk mengurutkan data berdasarkan variabel jumlah bersaudara.
- ⇒ Gunakan fungsi *desc()* untuk mengurutkan data dari yang terbesar, sampai terkecil."

Perintah R

```
arrange(filter(select(Data1, Nama, Pendidikan, Jumlah_Bersaudara), Pendidikan=="S2" | Pendidikan=="S3"), desc(Jumlah_Bersaudara))
```

Alternatif Perintah R

```
Data1 %>%
filter(Pendidikan=="S2" | Pendidikan=="S3") %>%
select(Nama, Pendidikan, Jumlah_Bersaudara) %>%
arrange(desc(Jumlah_Bersaudara))
```

Praktek Ke-15: Menggunakan Fungsi *mutate()* ()

```

Console -/
> Nama <- c("Ugi", "Egi", "Andi", "Udin", "Boy", "Sinta", "Rini", "Rani", "Fitri", "Dini")
> Jenis_Kelamin <- c("Laki-Laki", "Laki-Laki", "Laki-Laki", "Laki-Laki", "Laki-Laki", "Perempuan", "Perempuan", "Perempuan", "Perempuan", "Perempuan")
> Nilai_Matematika <- c(75, 60, 45, 40, 50, 75, 70, 80, 90, 80)
> Nilai_Kimia <- c(50, 70, 75, 50, 50, 85, 60, 90, 50, 70)
> Nilai_Biologi <- c(70, 50, 80, 80, 75, 80, 40, 80, 75, 90)
> Data1 <- data.frame(Nama, Jenis_Kelamin, Nilai_Matematika, Nilai_Kimia, Nilai_Biologi)
> Data1
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi
1  Ugi      Laki-Laki           75           50           70
2  Egi      Laki-Laki           60           70           50
3  Andi     Laki-Laki           45           75           80
4  Udin     Laki-Laki           40           50           80
5  Boy      Laki-Laki           50           50           75
6  Sinta    Perempuan          75           85           80
7  Rini     Perempuan          70           60           40
8  Rani     Perempuan          80           90           80
9  Fitri    Perempuan          90           50           75
10 Dini     Perempuan          80           70           90

> mutate(Data1, Rata_Rata = (Nilai_Matematika + Nilai_Kimia + Nilai_Biologi)/3)
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi Rata_Rata
1  Ugi      Laki-Laki           75           50           70  65.00000
2  Egi      Laki-Laki           60           70           50  60.00000
3  Andi     Laki-Laki           45           75           80  66.66667
4  Udin     Laki-Laki           40           50           80  56.66667
5  Boy      Laki-Laki           50           50           75  58.33333
6  Sinta    Perempuan          75           85           80  80.00000
7  Rini     Perempuan          70           60           40  56.66667
8  Rani     Perempuan          80           90           80  83.33333
9  Fitri    Perempuan          90           50           75  71.66667
10 Dini     Perempuan          80           70           90  80.00000

> |

>
> mutate(Data1, Rata_Rata = (Nilai_Matematika + Nilai_Kimia + Nilai_Biologi)/3)
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi Rata_Rata
1  Ugi      Laki-Laki           75           50           70  65.00000
2  Egi      Laki-Laki           60           70           50  60.00000
3  Andi     Laki-Laki           45           75           80  66.66667
4  Udin     Laki-Laki           40           50           80  56.66667
5  Boy      Laki-Laki           50           50           75  58.33333
6  Sinta    Perempuan          75           85           80  80.00000
7  Rini     Perempuan          70           60           40  56.66667
8  Rani     Perempuan          80           90           80  83.33333
9  Fitri    Perempuan          90           50           75  71.66667
10 Dini     Perempuan          80           70           90  80.00000

> Data1 %>%
+ mutate(Rata_Rata = (Nilai_Matematika + Nilai_Kimia + Nilai_Biologi)/3)
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi Rata_Rata
1  Ugi      Laki-Laki           75           50           70  65.00000
2  Egi      Laki-Laki           60           70           50  60.00000
3  Andi     Laki-Laki           45           75           80  66.66667
4  Udin     Laki-Laki           40           50           80  56.66667
5  Boy      Laki-Laki           50           50           75  58.33333
6  Sinta    Perempuan          75           85           80  80.00000
7  Rini     Perempuan          70           60           40  56.66667
8  Rani     Perempuan          80           90           80  83.33333
9  Fitri    Perempuan          90           50           75  71.66667
10 Dini     Perempuan          80           70           90  80.00000

> |

```

Gambar 12.15

"Tugas:

- ⇒ Gunakan fungsi *mutate()* untuk membentuk variabel baru bernama rata-rata, berdasarkan nilai ujian matematika, kimia, dan biologi, dari masing-masing responden.
- ⇒ Nilai rata-rata dihitung dengan rumus:

$$rata - rata = \frac{nilai\ matematika + nilai\ kimia + nilai\ biologi}{3}$$

⇒ Kemudian tampilkan informasi dari variabel nama, jenis kelamin, nilai matematika, nilai kimia, nilai biologi, dan nilai rata-rata.”

Perintah R

```
mutate(Data1, Rata_Rata = (Nilai_Matematika + Nilai_Kimia +  
Nilai_Biologi)/3)
```

Alternatif Perintah R

```
Data1 %>%  
mutate(Rata_Rata = (Nilai_Matematika + Nilai_Kimia + Nilai_Biologi)/3)
```

Praktek Ke-16: Menggunakan Fungsi *mutate()* dan *ifelse()*

```
Console ~/
> Data1
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi
1 Ugi Laki-Laki 75 50 70
2 Egi Laki-Laki 60 70 50
3 Andi Laki-Laki 45 75 80
4 Udin Laki-Laki 40 50 80
5 Boy Laki-Laki 50 50 75
6 Sinta Perempuan 75 85 80
7 Rini Perempuan 70 60 40
8 Rani Perempuan 80 90 80
9 Fitri Perempuan 90 50 75
10 Dini Perempuan 80 70 90
> mutate(Data1, Kelulusan_Matematika = ifelse(Nilai_Matematika<60, "TL", "L"))
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi Kelulusan_Matematika
1 Ugi Laki-Laki 75 50 70 L
2 Egi Laki-Laki 60 70 50 L
3 Andi Laki-Laki 45 75 80 TL
4 Udin Laki-Laki 40 50 80 TL
5 Boy Laki-Laki 50 50 75 TL
6 Sinta Perempuan 75 85 80 L
7 Rini Perempuan 70 60 40 L
8 Rani Perempuan 80 90 80 L
9 Fitri Perempuan 90 50 75 L
10 Dini Perempuan 80 70 90 L
> Data1 %>%
+ mutate(Kelulusan_Matematika = ifelse(Nilai_Matematika<60, "TL", "L"))
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi Kelulusan_Matematika
1 Ugi Laki-Laki 75 50 70 L
2 Egi Laki-Laki 60 70 50 L
3 Andi Laki-Laki 45 75 80 TL
4 Udin Laki-Laki 40 50 80 TL
5 Boy Laki-Laki 50 50 75 TL
6 Sinta Perempuan 75 85 80 L
7 Rini Perempuan 70 60 40 L
8 Rani Perempuan 80 90 80 L
9 Fitri Perempuan 90 50 75 L
10 Dini Perempuan 80 70 90 L
> |
```

Gambar 12.16

"Tugas:

- ⇒ Gunakan fungsi *mutate()* untuk membentuk variabel baru bernama kelulusan matematika.
- ⇒ Gunakan fungsi *ifelse()* untuk membuat kriteria dari variabel kelulusan matematika sebagai berikut.
 - Jika nilai matematika < 60, maka tidak lulus "TL".
 - Jika nilai matematika > 60, maka lulus "L".

Perintah R

```
mutate(Data1, Kelulusan_Matematika = ifelse(Nilai_Matematika<60, "TL", "L"))
```

Alternatif Perintah R

```
Data1 %>%
mutate(Kelulusan_Matematika = ifelse(Nilai_Matematika<60, "TL", "L"))
```

Praktek Ke-17: Menggunakan Fungsi *mutate()* dan *ifelse()*

```

Console ~/
> Data1
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi
1  Ugi      Laki-Laki           75           50           70
2  Egi      Laki-Laki           60           70           50
3  Andi     Laki-Laki           45           75           80
4  Udin     Laki-Laki           40           50           80
5  Boy      Laki-Laki           50           50           75
6  Sinta    Perempuan          75           85           80
7  Rini     Perempuan          70           60           40
8  Rani     Perempuan          80           90           80
9  Fitri    Perempuan          90           50           75
10 Dini     Perempuan          80           70           90
> ekstrak <- select(
+   mutate(Data1, Rata_Rata = (Nilai_Matematika+Nilai_Kimia+Nilai_Biologi)/3), Nama, Rata_Rata)
> ekstrak
  Nama Rata_Rata
1  Ugi  65.00000
2  Egi  60.00000
3  Andi 66.66667
4  Udin 56.66667
5  Boy  58.33333
6  Sinta 80.00000
7  Rini 56.66667
8  Rani 83.33333
9  Fitri 71.66667
10 Dini 80.00000
>
> Nilai <- mutate(ekstrak, Nilai =
+   ifelse(Rata_Rata < 60, "D",
+   +   ifelse(Rata_Rata < 70, "C",
+   +   +   ifelse(Rata_Rata < 80, "B", "A"))
+   )
+ )
> Nilai
  Nama Rata_Rata Nilai
1  Ugi  65.00000    C
2  Egi  60.00000    C
3  Andi 66.66667    C
4  Udin 56.66667    D
5  Boy  58.33333    D
6  Sinta 80.00000    A
7  Rini 56.66667    D
8  Rani 83.33333    A
9  Fitri 71.66667    B
10 Dini 80.00000    A
>
>
> Nilai <- mutate(ekstrak, Nilai =
+   ifelse(Rata_Rata < 60, "D",
+   +   ifelse(Rata_Rata < 70, "C",
+   +   +   ifelse(Rata_Rata < 80, "B", "A"))
+   )
+ )
> Nilai
  Nama Rata_Rata Nilai
1  Ugi  65.00000    C
2  Egi  60.00000    C
3  Andi 66.66667    C
4  Udin 56.66667    D
5  Boy  58.33333    D
6  Sinta 80.00000    A
7  Rini 56.66667    D
8  Rani 83.33333    A
9  Fitri 71.66667    B
10 Dini 80.00000    A
>
> |

```



```

Console ~/
> ekstrak <- Data1 %>%
+   mutate(Rata_Rata = (Nilai_Matematika+Nilai_Kimia+Nilai_Biologi)/3) %>%
+   select>Nama, Rata_Rata)
> ekstrak
  Nama Rata_Rata
1   Ugi 65.00000
2   Egi 60.00000
3  Andi 66.66667
4  Udin 56.66667
5   Boy 58.33333
6  Sinta 80.00000
7   Rini 56.66667
8   Rani 83.33333
9  Fitri 71.66667
10 Dini 80.00000
>
> nilai <- ekstrak %>%
+   mutate(Nilai = ifelse(Rata_Rata < 60, "D",
+                         ifelse(Rata_Rata < 70, "C",
+                         ifelse(Rata_Rata < 80, "B", "A"))))
> nilai
  Nama Rata_Rata Nilai
1   Ugi 65.00000    C
2   Egi 60.00000    C
3  Andi 66.66667    C
4  Udin 56.66667    D
5   Boy 58.33333    D
6  Sinta 80.00000    A
7   Rini 56.66667    D
8   Rani 83.33333    A
9  Fitri 71.66667    B
10 Dini 80.00000    A
>

```

Gambar 12.17

"Tugas:

Tahap 1

- ⇒ Gunakan fungsi *mutate()* pada data *frameData1* untuk membentuk variabel baru bernama rata-rata.
- ⇒ Nilai rata-rata dihitung dengan rumus:

$$\text{rata - rata} = \frac{\text{nilai matematika} + \text{nilai kimia} + \text{nilai biologi}}{3}$$

- ⇒ Kemudian gunakan fungsi *select()* untuk memilih hanya variabel nama dan rata-rata, dan kemudian disimpan dalam data frame ekstrak.

Tahap2

- ⇒ Gunakan fungsi *mutate()* pada data *frameekstrak* untuk membentuk variabel baru bernama nilai.
- ⇒ Gunakan fungsi *ifelse()* untuk membuat kriteria dari variabel nilai sebagai berikut.
 - Jika nilai rata-rata < 60, maka beri nilai D.
 - Jika nilai rata-rata < 70, maka beri nilai C.
 - Jika nilai rata-rata < 80, maka beri nilai B.
 - Jika nilai rata-rata ≥ 80, maka beri nilai A.

- ⇒ Kemudian gunakan fungsi *select()* untuk memilih hanya variabel nama, rata-rata dan nilai, yang disimpan dalam *data frame* nilai.

Perintah R

```
ekstrak <- select(
  mutate(Data1, Rata_Rata =
    (Nilai_Matematika+Nilai_Kimia+Nilai_Biologi)/3), Nama, Rata_Rata)
ekstrak

Nilai <- mutate(ekstrak, Nilai =
  ifelse(Rata_Rata < 60, "D",
        ifelse(Rata_Rata < 70, "C",
              ifelse(Rata_Rata < 80, "B", "A"))
        )
  )
Nilai
```

Alternatif Perintah R

```
ekstrak <- Data1 %>%
  mutate(Rata_Rata = (Nilai_Matematika+Nilai_Kimia+Nilai_Biologi)/3) %>%
  select(Nama, Rata_Rata)
ekstrak

nilai <- ekstrak %>%
  mutate(Nilai = ifelse(Rata_Rata < 60, "D",
                      ifelse(Rata_Rata < 70, "C",
                            ifelse(Rata_Rata < 80, "B", "A"))))
nilai
```

Praktek Ke-18: Menggunakan Fungsi *mutate()* dan *ifelse()*

```
Console ~/ |
> Data1
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi
1  Ugi      Laki-Laki           75           50           70
2  Egi      Laki-Laki           60           70           50
3  Andi     Laki-Laki           45           75           80
4  Udin     Laki-Laki           40           50           80
5  Boy      Laki-Laki           50           50           75
6  Sinta    Perempuan          75           85           80
7  Rini     Perempuan          70           60           40
8  Rani     Perempuan          80           90           80
9  Fitri    Perempuan          90           50           75
10 Dini     Perempuan          80           70           90
> Data1 %>%
+   mutate(
+     kelulusan_Matematika = ifelse(Nilai_Matematika < 60, "TL", "L"),
+     kelulusan_Kimia = ifelse(Nilai_Kimia < 60, "TL", "L"),
+     kelulusan_Biologi = ifelse(Nilai_Biologi < 60, "TL", "L"))
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi kelulusan_Matematika kelulusan_Kimia kelulusan_Biologi
1  Ugi      Laki-Laki           75           50           70             L             TL             L
2  Egi      Laki-Laki           60           70           50             L             L             TL
3  Andi     Laki-Laki           45           75           80             TL             L             L
4  Udin     Laki-Laki           40           50           80             TL             TL             L
5  Boy      Laki-Laki           50           50           75             TL             TL             L
6  Sinta    Perempuan          75           85           80             L             L             L
7  Rini     Perempuan          70           60           40             L             L             TL
8  Rani     Perempuan          80           90           80             L             L             L
9  Fitri    Perempuan          90           50           75             L             TL             L
10 Dini     Perempuan          80           70           90             L             L             L
> |
```

Gambar 12.18

"Tugas:

- ⇒ Gunakan fungsi *mutate()* untuk membentuk variabel kelulusan matematika.
- ⇒ Gunakan fungsi *ifelse()* untuk membuat syarat dari variabel kelulusan matematika. Syarat dari variabel kelulusan matematika adalah sebagai berikut.
 - Jika nilai matematika < 60, maka tidak lulus "TL".
 - Jika nilai matematika > 60, maka lulus "L".
- ⇒ Langkah yang sama untuk membuat variabel kelulusan kimia dan biologi."

Perintah R

```
Data1 %>%
  mutate(
    kelulusan_Matematika = ifelse(Nilai_Matematika < 60, "TL", "L"),
    kelulusan_Kimia = ifelse(Nilai_Kimia < 60, "TL", "L"),
    kelulusan_Biologi = ifelse(Nilai_Biologi < 60, "TL", "L"))
```

Praktek Ke-19: Menggunakan Fungsi *mutate()*, *ifelse()*, dan *select()*

```
Console ~/
> Data1
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi
1  Ugi      Laki-Laki           75           50           70
2  Egi      Laki-Laki           60           70           50
3  Andi     Laki-Laki           45           75           80
4  Udin     Laki-Laki           40           50           80
5  Boy      Laki-Laki           50           50           75
6  Sinta    Perempuan          75           85           80
7  Rini     Perempuan          70           60           40
8  Rani     Perempuan          80           90           80
9  Fitri    Perempuan          90           50           75
10 Dini     Perempuan          80           70           90
> |

Console ~/
> simpan <- Data1 %>%
+   mutate(
+     kelulusan_Matematika = ifelse(Nilai_Matematika < 60, 0, 1),
+     kelulusan_Kimia = ifelse(Nilai_Kimia < 60, 0, 1),
+     kelulusan_Biologi = ifelse(Nilai_Biologi < 60, 0, 1))
>
> simpan
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi kelulusan_Matematika kelulusan_Kimia kelulusan_Biologi
1  Ugi      Laki-Laki           75           50           70             1             0             1
2  Egi      Laki-Laki           60           70           50             1             1             0
3  Andi     Laki-Laki           45           75           80             0             1             1
4  Udin     Laki-Laki           40           50           80             0             0             1
5  Boy      Laki-Laki           50           50           75             0             0             1
6  Sinta    Perempuan          75           85           80             1             1             1
7  Rini     Perempuan          70           60           40             1             1             0
8  Rani     Perempuan          80           90           80             1             1             1
9  Fitri    Perempuan          90           50           75             1             0             1
10 Dini     Perempuan          80           70           90             1             1             1
>
> simpan %>%
+   mutate(Jumlah_Pelajaran_yang_Lulus = kelulusan_Matematika + kelulusan_Kimia + kelulusan_Biologi) %>%
+   select(Nama, Jumlah_Pelajaran_yang_Lulus)
  Nama Jumlah_Pelajaran_yang_Lulus
1  Ugi                2
2  Egi                2
3  Andi               2
4  Udin               1
5  Boy               1
6  Sinta              3
7  Rini               2
8  Rani               3
9  Fitri              2
10 Dini               3
>
```

Gambar 12.19

"Tugas:

Tahap 1

- ⇒ Gunakan fungsi *mutate()* pada data frame *Data1* untuk membentuk variabel kelulusan matematika.
- ⇒ Gunakan fungsi *ifelse()* untuk membuat syarat dari variabel kelulusan matematika. Syarat dari variabel kelulusan matematika adalah sebagai berikut.
 - Jika nilai matematika < 60, maka tidak lulus, diberi angka 0.
 - Jika nilai matematika > 60, maka lulus, diberi angka 1.
- ⇒ Langkah yang sama untuk membuat variabel kelulusan kimia dan biologi.
- ⇒ Selanjutnya simpan ke dalam data frame *simpan*.

Tahap 2

- ⇒ Gunakan fungsi *mutate()* pada *data frame* *simpan* untuk membentuk variabel jumlah pelajaran yang lulus.
- ⇒ Jumlah pelajaran yang lulus dari tiap-tiap responden dihitung dengan menjumlahkan

kelulusan matematika + kelulusan kimia + kelulusan biologi

- ⇒ Gunakan fungsi *select()* untuk hanya menampilkan informasi dari variabel nama dan jumlah pelajaran yang lulus.

Perintah R

```
simpan <- Data1 %>%
```

```
  mutate(  
    kelulusan_Matematika = ifelse(Nilai_Matematika < 60, 0, 1),  
    kelulusan_Kimia = ifelse(Nilai_Kimia < 60, 0, 1),  
    kelulusan_Biologi = ifelse(Nilai_Biologi < 60, 0, 1))
```

```
simpan
```

```
simpan %>%
```

```
mutate(Jumlah_Pelajaran_yang_Lulus = kelulusan_Matematika + kelulusan_Kimia  
+ kelulusan_Biologi ) %>%
```

```
select>Nama, Jumlah_Pelajaran_yang_Lulus)
```

Praktek Ke-20: Menggunakan Fungsi *mutate()*, *select()*, dan *apply()*

```
Console ~/
> Data1
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi
1  Ugi      Laki-Laki           75           50           70
2  Egi      Laki-Laki           60           70           50
3  Andi     Laki-Laki           45           75           80
4  Udin     Laki-Laki           40           50           80
5  Boy      Laki-Laki           50           50           75
6  Sinta    Perempuan          75           85           80
7  Rini     Perempuan          70           60           40
8  Rani     Perempuan          80           90           80
9  Fitri    Perempuan          90           50           75
10 Dini     Perempuan          80           70           90
>
> pelajaran <- select(Data1, Nilai_Matematika, Nilai_Kimia, Nilai_Biologi)
> pelajaran
  Nilai_Matematika Nilai_Kimia Nilai_Biologi
1              75           50           70
2              60           70           50
3              45           75           80
4              40           50           80
5              50           50           75
6              75           85           80
7              70           60           40
8              80           90           80
9              90           50           75
10             80           70           90
>
<
> Data1 %>%
+   mutate(
+     Minimum = apply(pelajaran, 1, FUN=min),
+     Maksimum = apply(pelajaran, 1, FUN=max),
+     Rata_Rata = apply(pelajaran, 1, FUN=mean),
+     Median = apply(pelajaran, 1, FUN=median),
+     standardeviasi = apply(pelajaran, 1, FUN=sd))
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi Minimum Maksimum Rata_Rata Median Standardeviasi
1  Ugi      Laki-Laki           75           50           70           50           75 65.00000 70 13.228757
2  Egi      Laki-Laki           60           70           50           50           70 60.00000 60 10.000000
3  Andi     Laki-Laki           45           75           80           45           80 66.66667 75 18.929694
4  Udin     Laki-Laki           40           50           80           40           80 56.66667 50 20.816660
5  Boy      Laki-Laki           50           50           75           50           75 58.33333 50 14.433757
6  Sinta    Perempuan          75           85           80           75           85 80.00000 80 5.000000
7  Rini     Perempuan          70           60           40           40           70 56.66667 60 15.275252
8  Rani     Perempuan          80           90           80           80           90 83.33333 80 5.773503
9  Fitri    Perempuan          90           50           75           50           90 71.66667 75 20.207259
10 Dini     Perempuan          80           70           90           70           90 80.00000 80 10.000000
>
```

Gambar 12.20

"Tugas:

- ⇒ Gunakan fungsi *select()* pada data frame *Data1* untuk memilih variabel nilai matematika, nilai kimia, dan nilai biologi dan simpan ke dalam data frame *pelajaran*.
- ⇒ Gunakan *mutate()* pada data frame *Data1* untuk membentuk variabel minimum, maksimum, rata-rata, median, dan standar deviasi.
- ⇒ Pada pembentukan variabel minimum, maksimum, rata-rata, median, dan standar deviasi, digunakan fungsi *apply()*.

Returns a vector or array or list of values obtained by applying a function to margins of an array or matrix.

Usage

```
apply(X, MARGIN, FUN, ...)
```

Arguments

X	an array, including a matrix.
MARGIN	a vector giving the subscripts which the function will be applied over. E.g., for a matrix 1 indicates rows, 2 indicates columns, c(1, 2) indicates rows and columns. Where X has named dimnames, it can be a character vector selecting dimension names.
FUN	the function to be applied: see 'Details'. In the case of functions like +, %*%, etc., the function name must be backquoted or quoted.
...	optional arguments to FUN.

Gambar 12.21

- ⇒ Pada penggunaan fungsi *apply()*, terdapat argumen *MARGIN*, yang mana nilai 1 menunjukkan baris, dan nilai 2 menunjukkan kolom (Gambar 21).

Perintah R

Data1

```
pelajaran <- select(Data1, Nilai_Matematika, Nilai_Kimia, Nilai_Biologi)
pelajaran
```

```
Data1 %>%
mutate(
  Minimum = apply(pelajaran, 1, FUN=min),
  Maksimum = apply(pelajaran, 1, FUN=max),
  Rata_Rata = apply(pelajaran, 1, FUN=mean),
  Median = apply(pelajaran, 1, FUN=median),
  Standardeviasi = apply(pelajaran, 1, FUN=sd))
```

Praktek Ke-21: Menggunakan Fungsi *mutate()*, *select()*, dan *apply()*

```

Console ~/
> Data1
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi
1  Ugi      Laki-Laki           75           50           NA
2  Egi      Laki-Laki           NA           70           50
3  Andi     Laki-Laki           45           75           NA
4  Udin     Laki-Laki           40           NA           NA
5  Boy      Laki-Laki           50           50           75
6  Sinta    Perempuan          75           85           80
7  Rini     Perempuan          70           NA           40
8  Rani     Perempuan          80           90           80
9  Fitri    Perempuan          90           NA           NA
10 Dini     Perempuan          80           70           90
> pelajaran <- select(Data1, Nilai_Matematika, Nilai_Kimia, Nilai_Biologi)
> pelajaran
  Nilai_Matematika Nilai_Kimia Nilai_Biologi
1             75             50             NA
2             NA             70             50
3             45             75             NA
4             40             NA             NA
5             50             50             75
6             75             85             80
7             70             NA             40
8             80             90             80
9             90             NA             NA
10            80             70             90
> Data1 %>%
+   mutate(
+     rata_rata = apply(pelajaran, 1, FUN = mean, na.rm=TRUE),
+     rata_rata2 = apply(pelajaran, 1, FUN = mean))
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi rata_rata rata_rata2
1  Ugi      Laki-Laki           75           50           NA 62.50000      NA
2  Egi      Laki-Laki           NA           70           50 60.00000      NA
3  Andi     Laki-Laki           45           75           NA 60.00000      NA
4  Udin     Laki-Laki           40           NA           NA 40.00000      NA
5  Boy      Laki-Laki           50           50           75 58.33333 58.33333
6  Sinta    Perempuan          75           85           80 80.00000 80.00000
7  Rini     Perempuan          70           NA           40 55.00000      NA
8  Rani     Perempuan          80           90           80 83.33333 83.33333
9  Fitri    Perempuan          90           NA           NA 90.00000      NA
10 Dini     Perempuan          80           70           90 80.00000 80.00000
> pelajaran <- select(Data1, Nilai_Matematika, Nilai_Kimia, Nilai_Biologi)
> pelajaran
  Nilai_Matematika Nilai_Kimia Nilai_Biologi
1             75             50             NA
2             NA             70             50
3             45             75             NA
4             40             NA             NA
5             50             50             75
6             75             85             80
7             70             NA             40
8             80             90             80
9             90             NA             NA
10            80             70             90
> Data1 %>%
+   mutate(
+     rata_rata = apply(pelajaran, 1, FUN = mean, na.rm=TRUE),
+     rata_rata2 = apply(pelajaran, 1, FUN = mean))
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi rata_rata rata_rata2
1  Ugi      Laki-Laki           75           50           NA 62.50000      NA
2  Egi      Laki-Laki           NA           70           50 60.00000      NA
3  Andi     Laki-Laki           45           75           NA 60.00000      NA
4  Udin     Laki-Laki           40           NA           NA 40.00000      NA
5  Boy      Laki-Laki           50           50           75 58.33333 58.33333
6  Sinta    Perempuan          75           85           80 80.00000 80.00000
7  Rini     Perempuan          70           NA           40 55.00000      NA
8  Rani     Perempuan          80           90           80 83.33333 83.33333
9  Fitri    Perempuan          90           NA           NA 90.00000      NA
10 Dini     Perempuan          80           70           90 80.00000 80.00000

```

Gambar 12.22

"Tugas:

- ⇒ Gunakan fungsi *select()* pada *data frame* *Data1* untuk memilih variabel nilai matematika, nilai kimia, dan nilai biologi dan simpan ke dalam *data frame* *pelajaran*.

- ⇒ Gunakan `mutate( )` pada *data frame* `Data1` untuk membentuk variabel rata-rata dan rata-rata2.
- ⇒ Pada pembentukan variabel rata-rata dan rata-rata2 digunakan fungsi `apply()`.
- ⇒ Pada pembentukan variabel rata-rata, melibatkan argumen `na.rm=TRUE`, sementara pada pembentukan variabel rata-rata2, tidak melibatkan argumen `na.rm=TRUE`.
- ⇒ Perhatikan perbedaan hasilnya.

Perintah R

`Data1`

```
pelajaran <- select(Data1, Nilai_Matematika, Nilai_Kimia, Nilai_Biologi)
pelajaran
```

```
Data1 %>%
  mutate(
    rata_rata = apply(pelajaran, 1, FUN = mean, na.rm=TRUE),
    rata_rata2 = apply(pelajaran, 1, FUN = mean))
```

Praktek Ke-22: Menggunakan Fungsi *mutate()*, *select()*, dan *apply()*

```

Console ~/
> Data1
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi
1  Ugi      Laki-Laki           75           50           NA
2  Egi      Laki-Laki           NA           70           50
3  Andi     Laki-Laki           45           75           NA
4  Udin     Laki-Laki           40           NA           NA
5  Boy      Laki-Laki           50           50           75
6  Sinta    Perempuan          75           85           80
7  Rini     Perempuan          70           NA           40
8  Rani     Perempuan          80           90           80
9  Fitri    Perempuan          90           NA           NA
10 Dini     Perempuan          80           70           90
> pelajaran <- select(Data1, Nilai_Matematika, Nilai_Kimia, Nilai_Biologi)
> pelajaran
  Nilai_Matematika Nilai_Kimia Nilai_Biologi
1             75             50             NA
2             NA             70             50
3             45             75             NA
4             40             NA             NA
5             50             50             75
6             75             85             80
7             70             NA             40
8             80             90             80
9             90             NA             NA
10            80             70             90
> Data1 %>%
+   mutate(
+     jumlah_ujian_yang_diikuti = apply(pelajaran, 1, function(x){ length(v
+ data rata = apply(pelajaran, 1, FUN = mean, na.rm=TRUE)
+
> Data1 %>%
+   mutate(
+     jumlah_ujian_yang_diikuti = apply(pelajaran, 1, function(x){ length(which(!is.na(x))) } ),
+     rata_rata = apply(pelajaran, 1, FUN = mean, na.rm=TRUE),
+     standar_deviasi = apply(pelajaran, 1, FUN = sd, na.rm=TRUE),
+     median = apply(pelajaran, 1, FUN = median, na.rm=TRUE),
+     minimum = apply(pelajaran, 1, FUN = min, na.rm=TRUE),
+     maksimum = apply(pelajaran, 1, FUN = mean, na.rm=TRUE),
+     jumlah = apply(pelajaran, 1, FUN = sum, na.rm=TRUE)
+   )
+
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi jumlah_ujian_yang_diikuti rata_rata standar_deviasi median minimum maksimum jumlah
1  Ugi      Laki-Laki           75           50           NA                2  62.50000    17.677670    62.5     50  62.50000    125
2  Egi      Laki-Laki           NA           70           50                2  60.00000    14.142136    60.0     50  60.00000    120
3  Andi     Laki-Laki           45           75           NA                2  60.00000    21.213203    60.0     45  60.00000    120
4  Udin     Laki-Laki           40           NA           NA                1  40.00000         NA      40.0     40  40.00000     40
5  Boy      Laki-Laki           50           50           75                3  58.33333    14.433757    50.0     50  58.33333    175
6  Sinta    Perempuan          75           85           80                3  80.00000     5.000000    80.0     75  80.00000    240
7  Rini     Perempuan          70           NA           40                2  55.00000    21.213203    55.0     40  55.00000    110
8  Rani     Perempuan          80           90           80                3  83.33333     5.773503    80.0     80  83.33333    250
9  Fitri    Perempuan          90           NA           NA                1  90.00000         NA     90.0     90  90.00000     90
10 Dini     Perempuan          80           70           90                3  80.00000    10.000000    80.0     70  80.00000    240

```

Gambar 12.23

"Tugas:

- ⇒ Gunakan fungsi *select()* pada *data frame* *Data1* untuk memilih variabel nilai matematika, nilai kimia, dan nilai biologi dan simpan ke dalam *data frame* *pelajaran*.
- ⇒ Gunakan *mutate()* pada *data frame* *Data1* untuk membentuk variabel jumlah ujian yang diikuti, rata-rata, standar deviasi, median, minimum, maksimum, dan jumlah.
- ⇒ Pada pembentukan variabel jumlah ujian yang diikuti, rata-rata, standar deviasi, median, minimum, maksimum, dan jumlah, digunakan fungsi *apply()*.
- ⇒ Pada penggunaan fungsi *apply()*, argumen *na.rm=TRUE* bertujuan untuk mengabaikan data yang kosong.

Perintah R

Data1

```
pelajaran <- select(Data1, Nilai_Matematika, Nilai_Kimia, Nilai_Biologi)
pelajaran
```

```
Data1 %>%
  mutate(
    jumlah_ujian_yang_diikuti = apply(pelajaran, 1, function(x){
length(which(!is.na(x))) } ),
    rata_rata = apply(pelajaran, 1, FUN = mean, na.rm=TRUE),
    standar_deviasi = apply(pelajaran, 1, FUN = sd, na.rm=TRUE),
    median = apply(pelajaran, 1, FUN = median, na.rm=TRUE),
    minimum = apply(pelajaran, 1, FUN = min, na.rm=TRUE),
    maksimum = apply(pelajaran, 1, FUN = max, na.rm=TRUE),
    jumlah = apply(pelajaran, 1, FUN = sum, na.rm=TRUE)
  )
```

Praktek Ke-23: Menggunakan Fungsi *summarise()*

```
Console ~/ |
> Data1
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi
1  Ugi      Laki-Laki           75           50           70
2  Egi      Laki-Laki           60           70           50
3  Andi     Laki-Laki           45           75           80
4  Udin     Laki-Laki           40           50           80
5  Boy      Laki-Laki           50           50           75
6  Sinta    Perempuan          75           85           80
7  Rini     Perempuan          70           60           40
8  Rani     Perempuan          80           90           80
9  Fitri    Perempuan          90           50           75
10 Dini     Perempuan          80           70           90
> Data1 %>%
+ summarise(
+ rata_rata_matematika = mean(Nilai_Matematika),
+ rata_rata_kimia = mean(Nilai_Kimia),
+ rata_rata_biologi = mean(Nilai_Biologi),
+ standar_deviasi_matematika = sd(Nilai_Matematika),
+ standar_deviasi_kimia = sd(Nilai_Kimia),
+ standar_deviasi_biologi = sd(Nilai_Biologi),
+ minimum_matematika = min(Nilai_Matematika),
+ minimum_kimia = min(Nilai_Kimia),
+ minimum_biologi = min(Nilai_Biologi),
+ maksimum_matematika = max(Nilai_Matematika),
+ maksimum_kimia = max(Nilai_Kimia),
+ maksimum_biologi = max(Nilai_Biologi)
+ )
  rata_rata_matematika rata_rata_kimia rata_rata_biologi standar_deviasi_matematika standar_deviasi_kimia standar_deviasi_biologi minimum_matematika
1                66.5             65              72                16.84076             15.27525             15.31158             40
  minimum_kimia minimum_biologi maksimum_matematika maksimum_kimia maksimum_biologi
1              50              40              90              90              90
> |
```

Gambar 12.24

"Tugas:

- ⇒ **Gunakan fungsi *summarise()* untuk menghitung nilai rata-rata, standar deviasi, minimum, dan maksimum, dari variabel nilai matematika, kimia, dan biologi."**

Perintah R

```
Data1 %>%
summarise(
rata_rata_matematika = mean(Nilai_Matematika),
rata_rata_kimia = mean(Nilai_Kimia),
rata_rata_biologi = mean(Nilai_Biologi),
standar_deviasi_matematika = sd(Nilai_Matematika),
standar_deviasi_kimia = sd(Nilai_Kimia),
standar_deviasi_biologi = sd(Nilai_Biologi),
minimum_matematika = min(Nilai_Matematika),
minimum_kimia = min(Nilai_Kimia),
minimum_biologi = min(Nilai_Biologi),
maksimum_matematika = max(Nilai_Matematika),
maksimum_kimia = max(Nilai_Kimia),
maksimum_biologi = max(Nilai_Biologi)
)
```

Praktek Ke-24: Menggunakan Fungsi *summarise()*

The screenshot shows an R console window with the following content:

```
> Data1
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi
1  Ugi      Laki-Laki             75          50           NA
2  Egi      Laki-Laki             NA          70           50
3  Andi     Laki-Laki             45          75           NA
4  Udin     Laki-Laki             40          NA           NA
5  Boy      Laki-Laki             50          50           75
6  Sinta    Perempuan             75          85           80
7  Rini     Perempuan             70          NA           40
8  Rani     Perempuan             80          90           80
9  Fitri    Perempuan             90          NA           NA
10 Dini     Perempuan             80          70           90

> Data1 %>%
+ summarise(
+ rata_rata_matematika = mean(Nilai_Matematika),
+ rata_rata_matematika2 = mean(Nilai_Matematika, na.rm=TRUE))
  rata_rata_matematika rata_rata_matematika2
1                  NA              67.22222
```

Gambar 12.25

"Tugas:

- ⇒ Gunakan fungsi *summarise()* untuk menghitung nilai rata-rata matematika.
- ⇒ Karena data pada variabel nilai matematika memuat NA, maka beri tambahan argumen *na.rm=TRUE*."

Perintah R

```
Data1 %>%
summarise(
rata_rata_matematika = mean(Nilai_Matematika),
rata_rata_matematika2 = mean(Nilai_Matematika, na.rm=TRUE))
```

Praktek Ke-25: Menggunakan Fungsi *summarise()*

```
Console ~/
> Data1
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi
1  Ugi      Laki-Laki           75           50           NA
2  Egi      Laki-Laki           NA           70           50
3  Andi     Laki-Laki           45           75           NA
4  Udin     Laki-Laki           40           NA           NA
5  Boy      Laki-Laki           50           50           75
6  Sinta    Perempuan          75           85           80
7  Rini     Perempuan          70           NA           40
8  Rani     Perempuan          80           90           80
9  Fitri    Perempuan          90           NA           NA
10 Dini     Perempuan          80           70           90
> Data1 %>%
+   summarise(
+     rata_rata_matematika = mean(Nilai_Matematika, na.rm=TRUE),
+     median_matematika = median(Nilai_Matematika, na.rm=TRUE),
+     standar_deviasi_matematika = sd(Nilai_Matematika, na.rm=TRUE),
+     minimum_matematika = min(Nilai_Matematika, na.rm=TRUE),
+     maksimum_matematika = max(Nilai_Matematika, na.rm=TRUE),
+     jumlah_seluruh_nilai = sum(Nilai_Matematika, na.rm=TRUE),
+     banyaknya_nilai = length(na.omit(Nilai_Matematika)),
+     banyaknya_nilai_yang_berbeda = n_distinct(na.omit(Nilai_Matematika)))
# A tibble: 1 x 7
  rata_rata_matematika median_matematika standar_deviasi_matematika minimum_matematika maksimum_matematika jumlah_seluruh_nilai banyaknya_nilai
1          67.22222          75          17.6973          40          90          605          9
>
```

Gambar 12.26

"Tugas:

- ⇒ Gunakan fungsi *summarise()* untuk menghitung nilai rata-rata, median, modus, standar deviasi, minimum, maksimum, jumlah seluruh nilai, banyaknya nilai, dan banyaknya nilai yang berbeda, dari variabel nilai matematika.

Perintah R

```
Data1 %>%
  summarise(
    rata_rata_matematika = mean(Nilai_Matematika, na.rm=TRUE),
    median_matematika = median(Nilai_Matematika, na.rm=TRUE),
    standar_deviasi_matematika = sd(Nilai_Matematika, na.rm=TRUE),
    minimum_matematika = min(Nilai_Matematika, na.rm=TRUE),
    maksimum_matematika = max(Nilai_Matematika, na.rm=TRUE),
    jumlah_seluruh_nilai = sum(Nilai_Matematika, na.rm=TRUE),
    banyaknya_nilai = length(na.omit(Nilai_Matematika)),
    banyaknya_nilai_yang_berbeda =
n_distinct(na.omit(Nilai_Matematika)))
```

Praktek Ke-26: Menghitung Modus dari Suatu Data Menggunakan Fungsi *summarise()* dan *mfv()*

The screenshot shows an R console window with the following content:

```
> Data1
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi
1  Ugi      Laki-Laki           75           50           NA
2  Egi      Laki-Laki           NA           70           50
3  Andi     Laki-Laki           45           75           NA
4  Udin     Laki-Laki           40           NA           NA
5  Boy      Laki-Laki           50           50           75
6  Sinta    Perempuan          75           85           80
7  Rini     Perempuan          70           NA           40
8  Rani     Perempuan          80           90           80
9  Fitri    Perempuan          90           NA           NA
10 Dini     Perempuan          80           70           90

> library(modeest)
> mfv(na.omit(Data1$Nilai_Matematika)) #Modus dari data nilai matematika
[1] 75 80
> mfv(Data1$Nilai_Matematika, na.rm=TRUE) #Modus dari data nilai matematika
[1] 75 80
> mfv(na.omit(Data1$Nilai_Kimia)) #Modus dari data nilai kimia
[1] 50 70
> mfv(Data1$Nilai_Kimia, na.rm=TRUE) #Modus dari data nilai kimia
[1] 50 70
> mfv(na.omit(Data1$Nilai_Biologi)) #Modus dari data nilai biologi
[1] 80
> mfv(Data1$Nilai_Biologi, na.rm=TRUE) #Modus dari data nilai biologi
[1] 80
>
> Data1 %>%
+   summarise(
+     modus_matematika = mlv(Nilai_Matematika, na.rm=TRUE, method='mfv')[['M'
+ ]],
+     modus_kimia = mlv(Nilai_Kimia, na.rm=TRUE, method='mfv')[['M']],
+     modus_biologi = mlv(Nilai_Biologi, na.rm=TRUE, method='mfv')[['M']]
+   )
# A tibble: 1 x 4
#   modus_matematika modus_kimia modus_biologi
#   <dbl>         <dbl>         <dbl>
1  77.5          60          80
```

Gambar 12.27

"Tugas:

- ⇒ Gunakan fungsi *summarise()* untuk menghitung nilai rata-rata, median, modus, standar deviasi, minimum, maksimum, jumlah seluruh nilai, banyaknya nilai, dan banyaknya nilai yang berbeda, dari variabel nilai matematika.

Perintah R

```
Data1
library(modeest)
mfv(na.omit(Data1$Nilai_Matematika)) #Modus dari data nilai matematika
mfv(Data1$Nilai_Matematika, na.rm=TRUE) #Modus dari data nilai matematika
mfv(na.omit(Data1$Nilai_Kimia)) #Modus dari data nilai kimia
mfv(Data1$Nilai_Kimia, na.rm=TRUE) #Modus dari data nilai kimia
mfv(na.omit(Data1$Nilai_Biologi)) #Modus dari data nilai biologi
mfv(Data1$Nilai_Biologi, na.rm=TRUE) #Modus dari data nilai biologi
```

```
Data1 %>%
  summarise(
    modus_matematika = mlv(Nilai_Matematika, na.rm=TRUE,
method='mfv')[['M']],
    modus_kimia = mlv(Nilai_Kimia, na.rm=TRUE, method='mfv')[['M']],
    modus_biologi = mlv(Nilai_Biologi, na.rm=TRUE,
method='mfv')[['M']]
  )
```

Praktek Ke-27: Menggunakan Fungsi *distinct()*, *n_distinct()*, dan *summarise()*

```
Console ~/
> Data1
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1  Ugi      Laki-Laki      S1           2
2  Egi      Laki-Laki      S2           3
3  Andi     Laki-Laki      S3           1
4  Udin     Laki-Laki      S1           2
5  Boy      Laki-Laki      S2           5
6  Sinta    Perempuan      S2           2
7  Rini     Perempuan      S2           4
8  Rani     Perempuan      S1           2
9  Fitri    Perempuan      S3           3
10 Dini     Perempuan      S2           5
> Data1 %>%
+ distinct(Jenis_Kelamin)
  Jenis_Kelamin
1    Laki-Laki
2    Perempuan
> Data1 %>%
+ distinct(Pendidikan)
  Pendidikan
1      S1
2      S2
3      S3
> Data1 %>%
+ distinct(Jumlah_Bersaudara)
  Jumlah_Bersaudara
1      2
2      3
3      1
4      5
5      4
> Data1 %>%
+ summarise(n_distinct(Jumlah_Bersaudara))
  n_distinct(Jumlah_Bersaudara)
1                             5
> Data1 %>%
+ summarise(jumlah_kategori_pendidikan = n_distinct(Pendidikan))
  jumlah_kategori_pendidikan
1                           3
```

Gambar 12.28

"Tugas:

- ⇒ Gunakan fungsi *distinct()* untuk menentukan data-data yang berbeda (*unique values*) dari variabel jenis kelamin, pendidikan, dan jumlah bersaudara.
- ⇒ Gunakan fungsi *n_distinct()* untuk menentukan jumlah data yang berbeda.

Perintah R

```
Data1
```

```
Data1 %>%
distinct(Jenis_Kelamin)
```

```
Data1 %>%
distinct(Pendidikan)
```

```
Data1 %>%
distinct(Jumlah_Bersaudara)
```

```
Data1 %>%
summarise(n_distinct(Jumlah_Bersaudara))
```

```
Data1 %>%
summarise(jumlah_kategori_pendidikan = n_distinct(Pendidikan))
```


Praktek Ke-28: Menggunakan Fungsi *transmute()* dan *mutate()*

```
Console ~/
> Data1
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi
1  Ugi      Laki-Laki           75           50           70
2  Egi      Laki-Laki           60           70           50
3  Andi     Laki-Laki           45           75           80
4  Udin     Laki-Laki           40           50           80
5  Boy      Laki-Laki           50           50           75
6  Sinta    Perempuan          75           85           80
7  Rini     Perempuan          70           60           40
8  Rani     Perempuan          80           90           80
9  Fitri    Perempuan          90           50           75
10 Dini     Perempuan          80           70           90
> Data1 %>%
+ mutate(Rata_Rata = (Nilai_Matematika+Nilai_Kimia+Nilai_Biologi)/3)
  Nama Jenis_Kelamin Nilai_Matematika Nilai_Kimia Nilai_Biologi Rata_Rata
1  Ugi      Laki-Laki           75           50           70 65.00000
2  Egi      Laki-Laki           60           70           50 60.00000
3  Andi     Laki-Laki           45           75           80 66.66667
4  Udin     Laki-Laki           40           50           80 56.66667
5  Boy      Laki-Laki           50           50           75 58.33333
6  Sinta    Perempuan          75           85           80 80.00000
7  Rini     Perempuan          70           60           40 56.66667
8  Rani     Perempuan          80           90           80 83.33333
9  Fitri    Perempuan          90           50           75 71.66667
10 Dini     Perempuan          80           70           90 80.00000
> Data1 %>%
+ transmute(Rata_Rata = (Nilai_Matematika+Nilai_Kimia+Nilai_Biologi)/3)
  Rata_Rata
1 65.00000
2 60.00000
3 66.66667
4 56.66667
5 58.33333
6 80.00000
7 56.66667
8 83.33333
9 71.66667
10 80.00000
> |
```

Gambar 12.29

Perintah R

```
Data1
```

```
Data1 %>%
```

```
mutate(Rata_Rata = (Nilai_Matematika+Nilai_Kimia+Nilai_Biologi)/3)
```

```
Data1 %>%
```

```
transmute(Rata_Rata = (Nilai_Matematika+Nilai_Kimia+Nilai_Biologi)/3)
```

Praktek Ke-29: Menggunakan Fungsi `group_by()` dan `n()`

```
Console ~/
> Data1
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1  Ugi      <NA>      <NA>          2
2  Egi   Laki-Laki    S2           3
3  Andi   Laki-Laki    S3           1
4  Udin   Laki-Laki    S1           2
5  Boy    Laki-Laki    <NA>         5
6  Sinta   <NA>      <NA>          2
7  Rini    <NA>      S2           4
8  Rani   Perempuan    S1           2
9  Fitri   Perempuan    S3           3
10 Dini   Perempuan    S2           5
> grup <- group_by(Data1, Jenis_Kelamin)
> grup %>%
+ summarise(
+ distribusi_frekuensi_jenis_kelamin = n() )
# A tibble: 3 x 2
  Jenis_Kelamin distribusi_frekuensi_jenis_kelamin
  <fctr>          <int>
1  Laki-Laki         4
2  Perempuan         3
3      NA           3
> |
```

Gambar 12.30

Berdasarkan Gambar 12.30, diketahui jumlah laki-laki sebanyak 4, jumlah perempuan sebanyak 3, dan NA sebanyak 3.

Perintah R

```
grup <- group_by(Data1, Jenis_Kelamin)
grup %>%
summarise(
distribusi_frekuensi_jenis_kelamin = n() )
```

Praktek Ke-30: Menggunakan Fungsi `group_by()` dan `n()`

```
Console ~/
> Data1
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1   Ugi          <NA>         <NA>              2
2   Egi      Laki-Laki         S2              3
3  Andi      Laki-Laki         S3              1
4  Udin      Laki-Laki         S1              2
5   Boy      Laki-Laki         <NA>             5
6 Sinta          <NA>         <NA>              2
7  Rini          <NA>         S2              4
8  Rani      Perempuan         S1              2
9 Fitri      Perempuan         S3              3
10 Dini      Perempuan         S2              5
> grup <- group_by(Data1, Jenis_Kelamin) %>%
+ filter(Jenis_Kelamin != "NA" )
> grup
Source: local data frame [7 x 4]
Groups: Jenis_Kelamin [2]

  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
<fctr>      <fctr>      <fctr>      <dbl>
1   Egi      Laki-Laki         S2              3
2  Andi      Laki-Laki         S3              1
3  Udin      Laki-Laki         S1              2
4   Boy      Laki-Laki         NA              5
5  Rani      Perempuan         S1              2
6 Fitri      Perempuan         S3              3
7  Dini      Perempuan         S2              5
> grup %>%
+ summarise(
+ distribusi_frekuensi_jenis_kelamin = n())
# A tibble: 2 x 2
  Jenis_Kelamin distribusi_frekuensi_jenis_kelamin
  <fctr>      <int>
1   Laki-Laki         4
2  Perempuan         3
> |
```

Gambar 12.31

Berdasarkan Gambar 12.31, diketahui jumlah laki-laki sebanyak 4, dan jumlah perempuan sebanyak 3.

Perintah R

`Data1`

```
grup <- group_by(Data1, Jenis_Kelamin) %>%
filter(Jenis_Kelamin != "NA" )
grup
```

```
grup %>%
summarise(
distribusi_frekuensi_jenis_kelamin = n())
```

Praktek Ke-31: Menggunakan Fungsi *na.exclude()* ()

```
Console ~/
> Data1
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1  Ugi             <NA>         <NA>             2
2  Egi             Laki-Laki      S2             3
3  Andi             Laki-Laki      S3             1
4  Udin             Laki-Laki      S1             2
5  Boy              Laki-Laki      <NA>           5
6  Sinta            <NA>          <NA>           2
7  Rini             <NA>          S2             4
8  Rani             Perempuan      S1             2
9  Fitri            Perempuan      S3             3
10 Dini             Perempuan      S2             5
> na.exclude(Data1)
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
2  Egi             Laki-Laki      S2             3
3  Andi             Laki-Laki      S3             1
4  Udin             Laki-Laki      S1             2
8  Rani             Perempuan      S1             2
9  Fitri            Perempuan      S3             3
10 Dini             Perempuan      S2             5
> |
```

Gambar 12.32

Berdasarkan Gambar 12.32, digunakan fungsi *na.exclude()* () untuk mengeliminasi baris yang mengandung NA atau mengeliminasi *cell* yang kosong (tidak terisi / NA / *not available*)

Perintah R

```
Data1
na.exclude(Data1)
```

Praktek Ke-32: Menggunakan Fungsi `group_by()` dan `n()`

The screenshot shows an R console window with the following content:

```
> Data1
  Nama Jenis_Kelamin Pendidikan Jumlah_Bersaudara
1  Ugi             <NA>         <NA>             2
2  Egi             Laki-Laki     S2              3
3  Andi            Laki-Laki     S3              1
4  Udin            Laki-Laki     S1              2
5  Boy             Laki-Laki     <NA>            5
6  Sinta           <NA>         <NA>            2
7  Rini            <NA>         S2              4
8  Rani            Perempuan     S1              2
9  Fitri           Perempuan     S3              3
10 Dini            Perempuan     S2              5

> grup <- group_by(Data1, Jenis_Kelamin, Pendidikan)
> grup %>%
+ summarise(
+ frekuensi = n())
Source: local data frame [9 x 3]
Groups: Jenis_Kelamin [?]
```

	Jenis_Kelamin	Pendidikan	frekuensi
	<fctr>	<fctr>	<int>
1	Laki-Laki	S1	1
2	Laki-Laki	S2	1
3	Laki-Laki	S3	1
4	Laki-Laki	NA	1
5	Perempuan	S1	1
6	Perempuan	S2	1
7	Perempuan	S3	1
8	NA	S2	1
9	NA	NA	2

Dalam praktek ini, digunakan fungsi `group_by()` dan `n()` untuk membentuk distribusi frekuensi berdasarkan **jenis kelamin** dan **pendidikan**.

Diketahui terdapat 1 laki-laki dengan pendidikan S1, terdapat 1 laki-laki dengan pendidikan S2, dan seterusnya.

Gambar 12.33

Perintah R

```
Data1
grup <- group_by(Data1, Jenis_Kelamin, Pendidikan)
grup %>%
summarise(
frekuensi = n())
```

Praktek Ke-33: Menggunakan Fungsi `group_by()` dan `n()`

```
Console ~/ |
> Data1
  Nama Jenis_Kelamin Kelas Nilai_Matematika Nilai_Kimia
1  Ugi      Laki-Laki   I         75           50
2  Egi      Laki-Laki   II        60           70
3  Andi     Laki-Laki   III       45           75
4  Udin     Laki-Laki   I         75           50
5  Boy      Laki-Laki   II        50           50
6  Sinta    Perempuan   III       75           85
7  Rini     Perempuan   I         70           60
8  Rani     Perempuan   II        80           90
9  Fitri    Perempuan   III       90           50
10 Dini     Perempuan   I         80           70
>
> grup <- group_by(Data1, Jenis_Kelamin, Kelas)
> grup %>%
+   summarise(
+     rata_rata_matematika = mean(Nilai_Matematika),
+     rata_rata_kimia = mean(Nilai_Kimia),
+     nilai_berbeda_matematika = n_distinct(Nilai_Matematika),
+     nilai_berbeda_kimia = n_distinct(Nilai_Kimia),
+     minimum_matematika = min(Nilai_Matematika),
+     minimum_kimia = min(Nilai_Kimia),
+     jumlah_siswa = n())
Source: local data frame [6 x 9]
Groups: Jenis_Kelamin [?]
```

	Jenis_Kelamin	Kelas	rata_rata_matematika	rata_rata_kimia	nilai_berbeda_matematika	nilai_berbeda_kimia	minimum_matematika	minimum_kimia	jumlah_siswa
	<fctr>	<fctr>	<dbl>	<dbl>	<int>	<int>	<dbl>	<dbl>	<int>
1	Laki-Laki	I	75.0	50.0	1	1	75	50	2
2	Laki-Laki	II	55.0	60.0	2	2	50	50	2
3	Laki-Laki	III	45.0	75.0	1	1	45	75	1
4	Perempuan	I	75.0	65.0	2	2	70	60	2
5	Perempuan	II	80.0	90.0	1	1	80	90	1
6	Perempuan	III	82.5	67.5	2	2	75	50	2

```
>
> |
```

Dalam praktek ini, penggunaan fungsi **`group_by()`** dan **`n()`** bertujuan untuk membentuk distribusi frekuensi berdasarkan **jenis kelamin** dan **kelas**. Serta ingin diketahui rata-rata nilai berdasarkan **jenis kelamin** dan **kelas**.

➔Diketahui jumlah siswa dengan jenis kelamin laki-laki dari kelas I sebanyak 2. Nilai rata-rata laki-laki dari kelas I, untuk pelajaran matematika 75, untuk pelajaran kimia 50.

Gambar 12.34

Perintah R

Data1

```
grup <- group_by(Data1, Jenis_kelamin, kelas)
grup %>%
summarise(
rata_rata_matematika = mean(Nilai_Matematika),
rata_rata_kimia = mean(Nilai_Kimia),
nilai_berbeda_matematika = n_distinct(Nilai_Matematika),
nilai_berbeda_kimia = n_distinct(Nilai_Kimia),
minimum_matematika = min(Nilai_Matematika),
minimum_kimia = min(Nilai_Kimia),
jumlah_siswa = n())
```

Praktek Ke-34: Menggunakan Fungsi `join()` dan `data.frame()`

```
> Data1
  Nama Jenis_Kelamin
1 Ugi      Laki-Laki
2 Egi      Laki-Laki
3 Andi     Laki-Laki
> Data3
  Nama Nilai_Kimia
1 Ugi      50
2 Egi      70
3 Andi     75
> full_join(Data1,Data3)
Joining, by = "Nama"
  Nama Jenis_Kelamin Nilai_Kimia
1 Ugi      Laki-Laki      50
2 Egi      Laki-Laki      70
3 Andi     Laki-Laki      75
> |
```

Terdapat 2 data frame, yakni **Data1** dan **Data2**, yang selanjutnya digabung dengan fungsi `join()`.

Gambar 12.35

```
Console ~/
> Data1
  Nama Jenis_Kelamin
1 Ugi      Laki-Laki
2 Egi      Laki-Laki
3 Andi     Laki-Laki
> Data2
  Kelas Nilai_Matematika
1 I      75
2 II     60
3 III    45
> data.frame(Data1,Data2)
  Nama Jenis_Kelamin Kelas Nilai_Matematika
1 Ugi      Laki-Laki   I      75
2 Egi      Laki-Laki  II     60
3 Andi     Laki-Laki  III    45
> |
```

Terdapat 2 data frame, yakni **Data1** dan **Data2**, yang selanjutnya digabung dengan fungsi `data.frame()`.

Gambar 12.36

Perintah R

`Data1`

```
grup <- group_by(Data1, Jenis_Kelamin, kelas)
grup %>%
summarise(
  rata_rata_matematika = mean(Nilai_Matematika),
  rata_rata_kimia = mean(Nilai_Kimia),
  nilai_berbeda_matematika = n_distinct(Nilai_Matematika),
  nilai_berbeda_kimia = n_distinct(Nilai_Kimia),
  minimum_matematika = min(Nilai_Matematika),
  minimum_kimia = min(Nilai_Kimia),
  jumlah_siswa = n())
```

Tidy Data

Data yang rapi (*tidy data*) merupakan pondasi utama dalam memanipulasi data menggunakan *package dplyr* dan *tidyr*.

Gambar 12.37

Merubah Tampilan Daset

- `data_frame(a=1:3,b=4:6)`
Mengombinasikan vektors menjadi data frame
- `arrange(mtcars,mpg)`
Mengurutkan baris dari rendah ke tinggi berdasarkan variabel mpg
- `arrange(mtcars, desc(mpg))`
Mengurutkan baris dari tinggi ke rendah
- `rename(tb,y = year)`
Mengganti nama kolom atau variabel year menjadi y

Subset Observasi

Gambar 12.38

- `filter(iris, Sepal.Length > 7)`
Mengekstrak baris data sesuai kriteria yang ditentukan

- `distinct(iris)`
Menghilangkan baris yang nilai-nilainya sama
- `sample_frac(iris, 0.5, replace=TRUE)`
Mengambil random setengah baris data secara random
- `sample_n(iris, 10, replace=TRUE)`
Memilih n baris data secara random
- `slice(iris, 10:15)`
Mengambil baris data sesuai posisi yang diinginkan.
- `top_n(df, 2, x)`
Memilih dan mengurutkan 2 observasi dataset df berdasarkan variabel x

Subset Variabel

Gambar 12.39

- `select(iris, Sepal.Width, Petal.Length, Species)`
Memilih kolom berdasarkan nama atau fungsi bantuan (*helper*):
- `select(iris, contains("."))`
Memilih kolom yang memiliki nama variabelnya memiliki unsur karakter yang diinginkan.
Pada kasus ini titik(.)
- `select(iris, ends_with("Length"))`
Memilih kolom yang memiliki nama berakhiran *Length*
- `select(iris, everything())`
Memilih semua kolom
- `select(iris, matches(".t."))`
Memilih kolom yang namanya ada huruf *t*.
- `iris, num_range("x", 1:5)`
Memilih kolom yang bernama x_1, x_2, x_3, x_4, x_5
- `select(iris, one_of(c("Species", "Genus")))`
Memilih kolom dari salah satu *Species* atau *Genus*
- `select(iris, start_with("Sepal"))`
Memilih kolom yang namanya dimulai dengan kata *Sepal*

- `select(iris, Sepal.Length:Petal.Width)`
Memilih semua kolom diantara *Sepal.Length* dan *Petal.Width*
- `select(iris, -Species)`
Memilih semua kolom selain *Species*

Summarise Data

Gambar 13.40

- `summarise(iris, avg=mean(Sepal.Length))`
Meringkas data menjadi menjadi satu baris
 - `summarise_each(iris, funs(mean))`
Menerapkan fungsi *summary* yakni mean ke masing-masing kolom
 - `count(iris, Species, wt=Sepal.Length)`
Mencacah banyaknya kategori species dengan kriteria *Sepal.Length*
- Beberapa fungsi *summary* yang sering digunakan,

first	min
last	max
nth	median
n	mean
n_distinct	var
IQR	sd

Membuat Variabel Baru

Gambar 13.41

- `mutate(iris, sepal = Sepal.Length + Sepal.Width)`
Membuat dan menggabungkan variabel *sepal* yang merupakan jumlahan antara Length dan Width
- `mutate_each(iris, funs(min_rank))`
Menerapkan fungsi *window* ke masing-masing kolom
- `transmute(iris, sepal=Sepal.Length + Sepal.Width)`
Membuat variabel baru *sepal* dan menghilangkan kolom lainnya

Mutate menggunakan fungsi *window*, fungsi yang mengambil vektor sebagai masukan dan menghasilkan vektor juga, seperti

lead	cumall
lag	cumany
dense_rank	cummean
min_rank	cumsum
percent_rank	cummax
row_number	cummin
ntile	cumprod
between	pmax
cume_dist	pmin

Grouping Data

- `group_by(iris, Species)`
Mengelompokkan data menjadi baris yang bersesuaian dengan data Species
- `ungroup(iris)`
Menghilangkan informasi pengelompokan dari dataframe
- `iris %>% group_by(Species) %>% summarise(...)`

Gambar 13.42

Mengelompokkan data iris berdasarkan Species dan diringkas berdasarkan fungsi *summarise*

- `iris %>% group_by(Species) %>% mutate(...)`

Gambar 13.43

Combine Datasets

Gambar 13.44

⇒ Mutating Join

- `left_join(a,b,by="x1")`
Menggabungkan baris dari data b ke a\

x1	x2	x3
A	1	T
B	2	F
C	3	NA

- `right_join(a,b,by="x1")`
Menggabungkan baris dari a ke b

x1	x3	x2
A	T	1
B	F	2
D	T	NA

- `inner_join(a,b,by="x1")`
Menggabungkan data yang beririsan.

x1	x2	x3
A	1	T
B	2	F

- `full_join(a,b,by="x1")`
Menggabungkan data. Semua data yang ada.

x1	x2	x3
A	1	T
B	2	F
C	3	NA
D	NA	T

⇒ Filtering Join

- `semi_join(a, b, by="x1")`
Semua baris di a yang memiliki sama dengan b
- `anti_join(a, b, by="x1")`
Semua data yang tidak beririsan ditampilkan.

x1	x2
A	1
B	2

x1	x2
C	3

⇒ Operasi Himpunan

Gambar 13.45

- `intersect(y, z)`
Baris yang ada tepat di y dan z
- `union(y, z)`
Menampilkan dengan menggabungkan dua-duanya
- `setdiff(y, z)`
Baris yang ada di y tapi tidak di z
- `bind_rows(y, z)`
Menambahkan z ke y sebagai baris baru
- `bind_cols(y, z)`
Menambahkan z ke y sebagai kolom baru

x1	x2
B	2
C	3

x1	x2
A	1
B	2
C	3
D	4

x1	x2
A	1

x1	x2
A	1
B	2
C	3
B	2
C	3
D	4

x1	x2	x1	x2
A	1	B	2
B	2	C	3
C	3	D	4

BAB 13

SEKILAS MEMBUAT APLIKASI OLAH DATA SEDERHANA DENGAN SHINY

13.1 Menginstal Package Shiny

Pada Bab 13 ini, akan diberikan contoh bagaimana membuat aplikasi olah data dengan shiny. Shiny adalah salah satu *package* R (*open source R package*) dari RStudio yang dapat digunakan untuk membuat *interactive web application*. Berikut alamat *website* resmi shiny <https://shiny.rstudio.com/>. Gambar 13.1 dan Gambar 13.2 disajikan cuplikan gambar dari *website* <https://shiny.rstudio.com/>. Untuk membuat program aplikasi dengan shiny, terlebih dahulu menginstal *package* shiny. Perhatikan Gambar 13.3. Pada Gambar 13.3 akan diinstal *package* shiny (pastikan telah terkoneksi internet!!!). Pada Gambar 13.4 diperlihatkan sedang proses menginstal *package* shiny. Pada Gambar 13.5 diperlihatkan telah sukses menginstal *package* shiny.

Gambar 13.1

Here is a Shiny app

Shiny apps are easy to write. No web development skills are required.

Gambar 13.2

Gambar 13.3

Gambar 13.4

Gambar 13.5

13.2 Membuat Program Aplikasi Shiny Paling Sederhana (Praktek 1)

Pada Gambar 13.6 diperlihatkan program aplikasi shiny paling sederhana, yakni **tampilan kosong**. Sekarang, akan akan dijelaskan langkah demi langkah untuk membuat program aplikasi shiny tersebut.

Gambar 13.6 (Program Aplikasi Shiny Paling Sederhana, Tampilan Kosong)

Langkah Ke-1

Jalankan Rstudio terlebih dahulu. Selanjutnya pilih *File =>New File =>R Script* (Gambar 13.7), sehingga muncul tampilan seperti pada Gambar 13.8.

Gambar 13.7

Langkah Ke-2

Selanjutnya tuliskan kode R seperti pada Gambar 13.9.

Gambar 13.8

Gambar 13.9

Langkah Ke-3

Setelah kode R ditulis seperti pada Gambar 13.9, selanjutnya pilih *File => Save* (Gambar 13.10). Pada bagian *File name*, beri nama **ui.r**. Kemudian simpan.

Gambar 13.10

Gambar 13.11

Langkah Ke-4

Pada langkah ke-3, telah dibuat *file* dengan nama **ui.r**. Sekarang pilih *File => New File => R Script* (Gambar 13.12), sehingga muncul tampilan seperti pada Gambar 13.12.

Gambar 13.12

Selanjutnya tuliskan kode R seperti pada Gambar 13.13.

Gambar 13.13

Setelah kode R ditulis seperti pada Gambar 13.13, selanjutnya pilih *File* => *Save* (Gambar 13.14). Pada bagian *File name*, beri nama **server.r**. Kemudian simpan.

Gambar 13.14

Gambar 13.15

Perhatikan bahwa telah dibentuk dua buah *file* dengan nama **ui** dan **server** (Gambar 13.15).

Gambar 13.16

Langkah Ke-5 (Jalankan Aplikasi)

Langkah terakhir, jalankan program aplikasi dengan meng-*click* **Run App** (Gambar 13.17). Maka hasilnya seperti pada Gambar 13.18.

Gambar 13.17

Gambar 13.18 (Program Aplikasi Shiny Paling Sederhana, Tampilan Kosong)

Penjelasan Kode `ui.r` dan `server.r`

Ada dua komponen untuk membentuk program aplikasi dengan shiny, yakni:

- ⇒ *user-interface script.*
- ⇒ *server script.*

Pada proses pembuatan program aplikasi shiny, telah dibuat *file* dengan nama **`ui.r`** dan **`server.r`**. Perhatikan bahwa “ui” singkatan dari ***user-interface***. Pada kode R pada bagian ***user-interface***(`ui.r`), bertujuan untuk **mendesain tampilan dari program aplikasi**. *File* **`ui.r`** dan **`server.r`** disimpan dalam **folder yang sama**. Berikut kutipan dari halaman *website* <https://shiny.rstudio.com/tutorial/lesson1/>.

“Shiny apps have two components:

*a user-interface script
a server script*

The user-interface (ui) script controls the layout and appearance of your app. It is defined in a source script named ui.R”

Gambar 13.19 disajikan kode R untuk **`ui.r`** yang paling sederhana (tampilan kosong).

Gambar 13.19

Berdasarkan Gambar 13.19:

- ⇒ Kode R bagian 1 bertujuan untuk mengaktifkan (*load*) *package* shiny. Hal ini perlu dilakukan karena akan dibuat program aplikasi dengan menggunakan shiny.

- ⇒ Kode R bagian 2 merupakan format dari shiny untuk *user-interface* di shiny.
- ⇒ Pada contoh selanjutnya, di dalam fungsi *fluidPage*, tampilan dari program aplikasi shiny diatur.

Berikut kutipan dari halaman website <https://shiny.rstudio.com/tutorial/lesson2/>.

“Shiny ui.R scripts use the function `fluidPage` to create a display that automatically adjusts to the dimensions of your user’s browser window. You lay out your app by placing elements in the `fluidPage` function.”

For example, the ui.R script below creates a user-interface that has a title panel and then a sidebar layout, which includes a sidebar panel and a main panel. Note that these elements are placed within the `fluidPage` function.”

Gambar 13.20

Pada Gambar 13.20 disajikan kode R untuk **server.r**. Berdasarkan Gambar 13.20, diketahui:

- ⇒ Kode R bagian 1 bertujuan untuk mengaktifkan (*load*) *package* shiny. Hal ini perlu dilakukan karena akan dibuat program aplikasi dengan menggunakan shiny.
- ⇒ Kode R bagian 2 merupakan format dari shiny untuk server di shiny.
- ⇒ Pada contoh selanjutnya, di dalam fungsi *function(input,output)* nantinya berisi instruksi-instruksi mengenai, misalkan instruksi-instruksi untuk perhitungan, penyajian data ke dalam grafik, dan sebagainya.

Berikut kutipan dari halaman website <https://shiny.rstudio.com/tutorial/lesson1/>.

“The server.R script contains the instructions that your computer needs to build your app. Here is the server.R”

13.2 Membuat Program Aplikasi Penjumlahan (Praktek 2)

Pada Gambar 13.21 dan Gambar 13.22 diperlihatkan program aplikasi yang dapat digunakan untuk melakukan penjumlahan dua buah bilangan. Pada program aplikasi tersebut, diminta untuk memberikan dua buah bilangan, yakni bilangan pertama dan bilangan kedua. Setelah diinput bilangan pertama dan bilangan kedua, maka akan secara otomatis muncul hasil penjumlahan dari dua buah bilangan tersebut.

Program Aplikasi untuk Penjumlahan Dua Bilangan

Silahkan Berikan Bilangan Pertama:

Silahkan Berikan Bilangan Kedua:

Di Bawah Ini Hasil Penjumlahan Bilangan 1 dan Bilangan 2

3.000000

Gambar 13.21

Program Aplikasi untuk Penjumlahan Dua Bilangan

Silahkan Berikan Bilangan Pertama:

Silahkan Berikan Bilangan Kedua:

Di Bawah Ini Hasil Penjumlahan Bilangan 1 dan Bilangan 2

4.600000

Gambar 13.22

Untuk membuat program aplikasi shiny tersebut, tulis kode R untuk **ui.r** dan **server.r** seperti pada Gambar 13.23 dan 13.24. Simpan **ui.r** dan **server.r** dalam satu folder. Kemudian jalankan aplikasi dengan meng-*click* **Run App**, seperti pada Gambar 13.25. Akan muncul program aplikasi shiny seperti pada Gambar 13.26.

Gambar 13.23 (Kode R untuk ui.r)

Gambar 13.24 (Kode R untuk server.r)

Gambar 13.25

Program Aplikasi untuk Penjumlahan Dua Bilangan

Silahkan Berikan Bilangan Pertama:

Silahkan Berikan Bilangan Kedua:

Di Bawah Ini Hasil Penjumlahan Bilangan 1 dan Bilangan 2

Input bilangan sesuai yang diinginkan.

Gambar 13.26

Program Aplikasi untuk Penjumlahan Dua Bilangan

Silahkan Berikan Bilangan Pertama:

Silahkan Berikan Bilangan Kedua:

Di Bawah Ini Hasil Penjumlahan Bilangan 1 dan Bilangan 2

Gambar 13.27

Penjelasan Kode ui.R

Ada dua komponen untuk membentuk program aplikasi dengan shiny, yakni:

- ⇒ *user-interface script*.
- ⇒ *server script*.

Pada proses pembuatan program aplikasi untuk penjumlahan dua bilangan, telah dibuat *file* dengan nama **ui.r** dan **server.r**. Perhatikan bahwa “ui” singkatan dari **user-interface**. Pada kode R pada bagian **user-interface**, bertujuan untuk **mendesain tampilan dari program aplikasi**.

Berdasarkan Gambar 13.28:

- ⇒ Pada bagian 1 bertujuan untuk membuat judul “Program Aplikasi untuk Penjumlahan Dua Bilangan”.
- ⇒ Pada bagian 2, diminta untuk menginput bilangan, yakni bilangan pertama dan bilangan kedua.
- ⇒ Pada bagian 3 akan mencetak tulisan “Di Bawah ini Hasil Penjumlahan 1 dan Bilangan 2”.
- ⇒ Pada bagian 4 akan menampilkan hasil dari **hitung**, yang mana proses perhitungan dari **hitung** dapat dilihat pada *file server.r*.

Gambar 13.28

Penjelasan Kode server.R

Berdasarkan Gambar 13.29:

⇒ Pada bagian 1, kode R

Simpan_bilangan1 <- input\$bilangan1

dapat diartikan **simpan_bilangan1** ditugaskan untuk **menyimpan bilangan** yang diinput pada **bilangan1**. Kode R

Simpan_bilangan2 <- input\$bilangan2

dapat diartikan **simpan_bilangan2** ditugaskan untuk **menyimpan bilangan** yang diinput pada **bilangan2**.

⇒ Pada bagian 2, kode R

jumlah = simpan_bilangan1 + simpan_bilangan2

dapat diartikan **jumlah** ditugaskan untuk **menyimpan bilangan** dari hasil penjumlahan antara **simpan_bilangan1** dan **simpan_bilangan2**.

⇒ Pada bagian 3, kode R

cat(sprintf("%f", jumlah))

dapat diartikan untuk menampilkan bilangan pecahan (%f menandakan untuk bilangan pecahan) pada jumlah.

```
1 library(shiny)
2
3 shinyServer(
4
5
6   function(input, output) {
7
8
9     output$hitung <- renderPrint({ #proses perhitungan pada hitung
10
11       simpan_bilangan1 <- input$bilangan1 #bagian 1
12       simpan_bilangan2 <- input$bilangan2 #bagian 1
13       jumlah = simpan_bilangan1 + simpan_bilangan2 #bagian 2
14
15       cat(sprintf("%f", jumlah)) #bagian 3
16
17     })
18
19   }
20
21 )
```

Gambar 13.29

13.3 Membuat Program Aplikasi Penjumlahan, Pengurangan, Perkalian, dan Pembagian (Praktek 3)

Program aplikasi shiny praktek 3 ini, merupakan pengembangan dari program aplikasi shiny pada praktek 2, yang mana selain penjumlahan, juga dapat melakukan pengurangan, perkalian, dan pembagian. Perhatikan Gambar 13.30 dan Gambar 13.31. Pada program aplikasi tersebut, diminta untuk memberikan dua buah bilangan, yakni bilangan pertama dan bilangan kedua. Setelah diinput bilangan pertama dan bilangan kedua, maka akan secara otomatis muncul hasil penjumlahan, pengurangan, perkalian, dan pembagian dari dua buah bilangan tersebut.

Program Aplikasi untuk Penjumlahan, Pengurangan, Perkalian, dan Pembagian dari Dua Bilangan

Silahkan Berikan Bilangan Pertama:

Silahkan Berikan Bilangan Kedua:

Di Bawah Ini Hasil Penjumlahan Bilangan 1 dan Bilangan 2

7.000000

Di Bawah Ini Hasil Pengurangan Bilangan 1 dan Bilangan 2

1.000000

Di Bawah Ini Hasil Perkalian Bilangan 1 dan Bilangan 2

12.000000

Di Bawah Ini Hasil Pembagian Bilangan 1 dan Bilangan 2

1.333333

Gambar 13.30

Program Aplikasi untuk Penjumlahan, Pengurangan, Perkalian, dan Pembagian dari Dua Bilangan

Silahkan Berikan Bilangan Pertama:

10

Silahkan Berikan Bilangan Kedua:

2

Di Bawah Ini Hasil Penjumlahan Bilangan 1 dan Bilangan 2

12.000000

Di Bawah Ini Hasil Pengurangan Bilangan 1 dan Bilangan 2

8.000000

Di Bawah Ini Hasil Perkalian Bilangan 1 dan Bilangan 2

20.000000

Di Bawah Ini Hasil Pembagian Bilangan 1 dan Bilangan 2

5.000000

Gambar 13.31

```
1 library(shiny)
2
3 shinyUI(
4
5   fluidPage(
6
7     headerPanel("Program Aplikasi untuk Penjumlahan, Pengurangan,
8                 Perkalian, dan Pembagian dari Dua Bilangan"),
9
10    numericInput("bilangan1", label="Silahkan Berikan Bilangan Pertama:", value=""),
11    numericInput("bilangan2", label="Silahkan Berikan Bilangan Kedua:", value=""),
12
13    h1("Di Bawah Ini Hasil Penjumlahan Bilangan 1 dan Bilangan 2"), #Bagian 1
14    verbatimTextoutput("jumlah"), #Bagian 2
15
16    h2("Di Bawah Ini Hasil Pengurangan Bilangan 1 dan Bilangan 2"), #Bagian 3
17    verbatimTextoutput("kurang"), #Bagian 4
18
19    h3("Di Bawah Ini Hasil Perkalian Bilangan 1 dan Bilangan 2"), #Bagian 5
20    verbatimTextoutput("kali"), #Bagian 6
21
22    h4("Di Bawah Ini Hasil Pembagian Bilangan 1 dan Bilangan 2"), #Bagian 7
23    verbatimTextoutput("bagi") #Bagian 8
24
25  )
26
27 )
28
29 )
30 )
```

Perhatian!!! Harap teliti dalam pengetikan kode R terhadap tanda titik, koma, buka kurung kurawal.

Gambar 13.32 (Kode R untuk ui.r)

```

1 library(shiny)
2
3 shinyServer(
4
5
6   function(input, output) {
7
8     #####
9     #####Penjumlahan#####
10    #####
11
12    output$jumlah <- renderPrint({
13
14      simpan_bilangan1 <- input$bilangan1
15      simpan_bilangan2 <- input$bilangan2
16      jumlah = simpan_bilangan1 + simpan_bilangan2
17      cat(sprintf("%f", jumlah))
18
19    })
20
21    #####
22    #####Pengurangan#####
23    #####
24
25    output$kurang <- renderPrint({
26
27      simpan_bilangan1 <- input$bilangan1
28      simpan_bilangan2 <- input$bilangan2
29      selisih = simpan_bilangan1 - simpan_bilangan2
30      cat(sprintf("%f", selisih))
31
32    })
33
34

```

Perhatian!!! Harap teliti dalam pengetikan kode R terhadap tanda titik, koma, buka kurung kurawal.

Gambar 13.33 (Kode R untuk server.r)

```

29      selisih = simpan_bilangan1 - simpan_bilangan2
30      cat(sprintf("%f", selisih))
31
32    })
33
34
35    #####
36    #####Perkalian#####
37    #####
38
39    output$kali <- renderPrint({
40
41      simpan_bilangan1 <- input$bilangan1
42      simpan_bilangan2 <- input$bilangan2
43      perkalian = simpan_bilangan1 * simpan_bilangan2
44      cat(sprintf("%f", perkalian))
45
46    })
47
48
49    #####
50    #####Pembagian#####
51    #####
52
53    output$bagi <- renderPrint({
54
55      simpan_bilangan1 <- input$bilangan1
56      simpan_bilangan2 <- input$bilangan2
57      dibagi = simpan_bilangan1 / simpan_bilangan2
58      cat(sprintf("%f", dibagi))
59
60    })
61
62  }
63 )

```

60:7 (Untitled) ↕

Gambar 13.34 (Sambungan Kode R untuk server.r)

Untuk membuat program aplikasi shiny tersebut, tulis kode R untuk **ui.r** dan **server.r** seperti pada Gambar 13.32 dan 13.34. Simpan **ui.r** dan **server.r** dalam satu folder. Kemudian jalankan aplikasi dengan meng-*click* **Run App**. Akan muncul program aplikasi shiny seperti pada Gambar 13.30 dan Gambar 13.31.

Berdasarkan Gambar 13.35, perhatikan bahwa `h1("kalimat")`, `h2("kalimat")`, `h3("kalimat")`, dan `h4("kalimat")` bertujuan untuk menampilkan kalimat di antara tanda petik ganda, dengan ukuran kalimat yang berbeda-beda.

- ⇒ Pada bagian 2 akan menampilkan hasil dari **jumlah**, yang mana proses perhitungan dari **jumlah** dapat dilihat pada *file server.r*.
- ⇒ Pada bagian 4 akan menampilkan hasil dari **kurang**, yang mana proses perhitungan dari **kurang** dapat dilihat pada *file server.r*.
- ⇒ Pada bagian 6 akan menampilkan hasil dari **kali**, yang mana proses perhitungan dari **kali** dapat dilihat pada *file server.r*.
- ⇒ Pada bagian 8 akan menampilkan hasil dari **bagi**, yang mana proses perhitungan dari **bagi** dapat dilihat pada *file server.r*.

Gambar 13.35

Berikut disajikan bagian dari kode R pada **server.r**.

```

output$kali <- renderPrint({

  simpan_bilangan1 <- input$bilangan1

```

```

simpan_bilangan2 <- input$bilangan2
perkalian = simpan_bilangan1 * simpan_bilangan2
cat(sprintf("%f", perkalian))

})

```

Berdasarkan kode R di atas,

```

Simpan_bilangan1 <- input$bilangan1

```

dapat diartikan **simpan_bilangan1** ditugaskan untuk **menyimpan bilangan** yang diinput pada **bilangan1**. Kode R

```

Simpan_bilangan2 <- input$bilangan2

```

dapat diartikan **simpan_bilangan2** ditugaskan untuk **menyimpan bilangan** yang diinput pada **bilangan2**. Kode R

```

perkalian = simpan_bilangan1 * simpan_bilangan2

```

dapat diartikan **perkalian** ditugaskan untuk **menyimpan bilangan** dari **hasil perkalian** antara **simpan_bilangan1** dan **simpan_bilangan2**. Kode R

```

cat(sprintf("%f", perkalian))

```

dapat diartikan untuk menampilkan bilangan pecahan (%f menandakan untuk bilangan pecahan) pada **perkalian**.

13.4 Membuat Program Aplikasi Penjumlahan dengan Tampilan Berbeda (Praktek 4)

Tampilan program aplikasi shiny pada praktek 2 sebagai berikut (Gambar 13.36).

Program Aplikasi untuk Penjumlahan Dua Bilangan

Silahkan Berikan Bilangan Pertama:

Silahkan Berikan Bilangan Kedua:

Di Bawah Ini Hasil Penjumlahan Bilangan 1 dan Bilangan 2

5.000000

Gambar 13.36

Sekarang akan dibuat program aplikasi shiny seperti pada Gambar 13.37.

Program Aplikasi untuk Penjumlahan Dua Bilangan

Silahkan Berikan Bilangan Pertama:

2

Silahkan Berikan Bilangan Kedua:

4

Di Bawah Ini Hasil Penjumlahan Bilangan 1 dan Bilangan 2

6.000000

Gambar 13.37

Untuk membuat program aplikasi shiny tersebut, tulis kode R untuk **ui.r** dan **server.r** seperti pada Gambar 13.38 dan 13.39. Simpan **ui.r** dan **server.r** dalam satu folder. Kemudian jalankan aplikasi dengan meng-click **Run App**.

```
1 library(shiny)
2
3 shinyUI(
4   pagewithsidebar( #Bagian 1
5     headerPanel("Program Aplikasi untuk Penjumlahan Dua Bilangan"),
6
7     sidebarPanel( #Bagian 2
8       numericInput("bilangan1", label="Silahkan Berikan Bilangan Pertama:", value=""),
9       numericInput("bilangan2", label="Silahkan Berikan Bilangan Kedua:", value="")
10    ),
11
12    mainPanel( #Bagian 3
13      h3("Di Bawah Ini Hasil Penjumlahan Bilangan 1 dan Bilangan 2"),
14      verbatimTextOutput("hitung")
15    )
16  )
17 ) #tutup kurung dari pagewithsidebar
18
19 ) #tutup kurung dari shinyUI
```

Gambar 13.38

Berdasarkan Gambar 13.38, pada bagian 1, yang sebelumnya **fluidPage** sekarang menjadi **sidebarPanel**. Pada bagian 2, **sidebarPanel** untuk tampilan pada bagian kiri (perhatikan Gambar 13.40). Sementara bagian 3, **mainPanel** untuk tampilan pada bagian kanan (perhatikan Gambar 13.41).


```
1 library(shiny)
2
3 shinyServer(
4
5
6   function(input, output) {
7
8
9     output$hitung <- renderPrint({ #proses perhitungan pada hitung
10
11       simpan_bilangan1 <- input$bilangan1 #bagian 1
12       simpan_bilangan2 <- input$bilangan2 #bagian 1
13       jumlah = simpan_bilangan1 + simpan_bilangan2 #bagian 2
14
15       cat(sprintf("%f", jumlah)) #bagian 3
16
17     })
18
19   }
20
21 )
```

Gambar 13.39

Program Aplikasi u

Gambar 13.40 (Bagian sidebarPanel)

Di Bawah Ini Hasil Penjumlahan Bilangan 1 dan Bilangan 2

5.000000

Gambar 13.41 (Bagian mainPanel)

13.5 Berbagai Jenis Style Tulisan (Praktek 5)

Berikut disajikan berbagai jenis *style* tulisan dalam shiny.

Gambar 13.42

Gambar 13.43

Hai

Hai

Hai

Prana Ugiana Gio

Prana Ugiana Gio

Hai Prana Ugiana Gio & Ronnie

Prana Ugiana Gio Prana Ugiana Gio Prana Ugiana Gio

Prana Ugiana Gio, lahir di Tanjung Pinang

```
1 library(shiny)
2
3 shinyUI(
4   fluidPage(
5     h1("Hai", style = "color:red;"),
6     h1("Hai", style = "color:red; text-align:center;"),
7     h1("Hai", style = "color:red; text-align:right;"),
8
9     h1("Prana Ugiana Gio",
10      style = "font-family: 'Lobster', cursive;
11      font-weight: 500; line-height: 1.1;
12      color: red;"),
13
14     h1("Prana Ugiana Gio",
15      style = "font-family: 'Lobster', cursive;
16      color: green; text-align:center;"),
17
18     h1( span("Hai", style = "color:red;"), "Prana Ugiana",
19        span("Gio", style="color:green;"), "&",
20        span("Ronnie", style="font-style:italic; color:blue;"),
21        style="text-align:center" ),
22
23     h1(span("Prana Ugiana Gio", style="font-size:10px"),
24        span("Prana Ugiana Gio", style="font-size:20px"),
25        span("Prana Ugiana Gio", style="font-size:30px") ),
26
27     p("Prana Ugiana Gio, ", span("lahir di Tanjung Pinang",
28      style="font-weight:bold; font-style:italic;") ),
29
30     br()
31   ) #akhir fluidPage
32 ) #tutup kurung dari shinyUI
```

```
1 library(shiny)
2
3 shinyServer(
4   function(input, output) {
5
6   }
7 )
```

Gambar 13.44

Prana Ugiana Gio

Prana Ugiana Gio

Prana Ugiana Gio

Prana Ugiana Gio

Prana Ugiana Gio

Prana Ugiana Gio

Prana Ugiana Gio

Prana Ugiana Gio

Prana Ugiana Gio

```

1 library(shiny)
2
3 shinyUI(
4
5   fluidPage(
6
7     h1("Prana Ugiana Gio", style="text-shadow:2px 2px; color:red;" ),
8     h1("Prana Ugiana Gio", style="text-shadow:5px 2px; color:green;" ),
9     h1("Prana Ugiana Gio", style="text-shadow:2px 5px; color:blue;" ),
10    h1("Prana Ugiana Gio", style="text-shadow:2px 2px 5px purple;" ),
11    h1("Prana Ugiana Gio", style="text-shadow:2px 20px 1px orange;font-style:italic;" ),
12    h1("Prana Ugiana Gio", style="text-shadow: -1px 0 black,
13      0 1px black, 1px 0 black, 0 -1px black; color:yellow"),
14
15    h1("Prana Ugiana Gio", style="box-shadow:5px 5px; width:700px; background-color:yellow;
16      text-shadow: -1px 0 black,
17      0 1px black, 1px 0 black, 0 -1px black; color:green"),
18
19    h1("Prana Ugiana Gio", style="box-shadow:5px 5px; width:700px; background-color:yellow;
20      text-shadow: -1px 0 black,
21      0 1px black, 1px 0 black, 0 -1px black; color:red;
22      font-size:80px;"),
23
24    h1("Prana Ugiana Gio", style="box-shadow:5px 15px blue; width:700px; background-color:yellow;
25      text-shadow: -1px 0 black,
26      0 1px black, 1px 0 black, 0 -1px black; color:red;
27      font-size:80px;"),
28
29
30    br()
31  ) #akhir fluidPage
32 ) #tutup kurung dari shinyUI

```

```

1 library(shiny)
2
3 shinyServer(
4
5   function(input, output) {
6
7   }
8 )

```

Gambar 13.45

Gambar 13.46

13.6 Membuat Program Aplikasi Shiny (Praktek 6)

Sekarang akan dibuat program aplikasi shiny seperti pada Gambar 13.47. Program aplikasi shiny pada Gambar 13.37 dirancang untuk menyajikan data dari dua variabel (variabel **X** dan variabel **Y**) ke dalam grafik.

Program Aplikasi untuk Grafik Pencar

Gambar 13.47

Untuk membuat program aplikasi shiny tersebut, tulis kode R untuk **ui.r** dan **server.r** seperti pada Gambar 13.48 dan 13.49. Simpan **ui.r** dan **server.r** dalam satu folder. Kemudian jalankan aplikasi dengan meng-click **Run App**.

Berdasarkan Gambar 13.48, kode R

```
ambil_data <- read.csv(text=input$grafik_pencar, sep="", header = TRUE )
```

bertujuan untuk menampilkan tampilan seperti pada Gambar 13.50. Pada Gambar 15.50, diminta untuk mengisi data pada variabel **X** dan **Y**. Setelah mengisi data pada variabel **X** dan **Y**, maka secara otomatis akan muncul grafik pencar. Sementara kode R

```
plotOutput("grafik_pencar")
```

bertujuan untuk menampilkan grafik pencar. Instruksi dari **grafik_pencar** dapat dilihat pada bagian **server.r**, yakni

```

grafik_pencar <- function(){

ambil_data <- read.csv(text=input$grafik_pencar, sep="", header = TRUE )
ggplot(data=ambil_data, aes(x=X, y=Y)) + geom_point(size=7)

}

output$grafik_pencar <- renderPlot({
print(grafik_pencar())
})

```

Gambar 13.48 Kode ui.r

Gambar 13.49 Kode server.r

Gambar 13.50

Kode R

```
ambil_data <- read.csv(text=input$grafik_pencar, sep=";", header = TRUE )
```

bertujuan mendapatkan data variabel **X** dan **Y**, kemudian ditugaskan disimpan di **ambil_data**.
Selanjutnya kode R

```
ggplot(data=ambil_data, aes(x=X, y=Y)) + geom_point(size=7)
```


bertujuan untuk menampilkan grafik pencar, dengan ukuran titik sebesar 7 (**size = 7**).

13.7 Membuat Program Aplikasi Shiny (Praktek 7)

Sekarang akan dibuat program aplikasi shiny seperti pada Gambar 13.51. Program aplikasi shiny pada Gambar 13.51 merupakan pengembangan dari program aplikasi shiny pada Gambar 13.47.

Gambar 13.51

Untuk membuat program aplikasi shiny tersebut, tulis kode R untuk **ui.r** dan **server.r** seperti berikut.

```
server.R % ui.r %
1 library(shiny)
2 library(shinyAce)
3 library(ggplot2)
4
5 shinyUI(
6
7   pagewithsidebar(
8
9     headerPanel(span("Program Aplikasi untuk Grafik Pencar", style="color:blue;
10                      font-family:Lucida Console;
11                      box-shadow:5px 5px; width:700px; background-color:pink;
12                      font-size:40px")),
13
14     sidebarPanel(
15
16       h3("Silahkan Input Data Anda", style="color:red;text-shadow: -1px 0 black,
17         0 1px black, 1px 0 black, 0 -1px black;font-size:30px"),
18
19       aceEditor("grafik_pencar", value="x\t y", mode="r", theme="cobalt", height = 160),
20
21       br(),
22
23       sliderInput("ukuran_titik", label="ukuran Titik", min=0, max=10, value=3, step=0.2),
24
25       br(),
26
27       radioButtons("warna_titik", "Warna Titik", c("Merah"="red", "Biru"="blue",
28         "Hijau"="green", "Kuning"="yellow", "Pink"="pink", "Ungu"="purple", "Orange"="orange",
29         "Abu-Abu"="grey", "Hijau Gelap"="darkgreen", "Merah Gelap"="darkred", "Biru Gelap"="darkblue",
30         "Hitam"="black"), inline = TRUE ),
31
32     )
33
34   )
35 )
```

Gambar 13.52 Kode ui.r

```

r.R x ui.r x
br(),

sliderInput("ukuran_titik", label="Ukuran Titik", min=0, max=10, value=3, step=0.2),

br(),

radioButtons("warna_titik", "Warna Titik", c("Merah"="red", "Biru"="blue",
"Hijau"="green", "Kuning"="yellow", "Pink"="pink", "Ungu"="purple", "Orange"="orange",
"Abu-Abu"="grey", "Hijau Gelap"="darkgreen", "Merah Gelap"="darkred", "Biru Gelap"="darkblue",
"Hitam"="black"), inline = TRUE ),

radioButtons("bentuk", "Bentuk", c("I"="21",
"II"="22", "III"="23", "IV"="24", "V"="25"), inline = TRUE ),

br()
),
mainPanel(
  h3("Grafik Pencar", style="color:orange;text-shadow: -1px 0 black,
0 1px black, 1px 0 black, 0 -1px black;font-size:60px; text-align:center"),
  plotOutput("grafik_pencar")
)
)
)
(Top Level) R Script

```

Gambar 13.53 Sambungan Kode ui.r

```

server.R x ui.r x
1 library(shiny)
2 library(shinyAce)
3 library(ggplot2)
4
5 shinyServer(
6
7   function(input, output) {
8
9
10    grafik_pencar <- function(){
11
12
13      ambil_data <- read.csv(text=input$grafik_pencar, sep="", header = TRUE )
14
15      ukuran_titik <- input$ukuran_titik
16      warna_titik <- input$warna_titik
17      bentuk <- input$bentuk
18      bentuk = as.numeric(bentuk)
19
20      ggplot(data=ambil_data, aes(x=X, y=Y)) + geom_point(size=ukuran_titik,
21      colour=c(warna_titik), shape=c(bentuk), fill=c(warna_titik) )
22
23    }
24
25    output$grafik_pencar <- renderPlot({
26      print(grafik_pencar())
27    })
28
29
30
21:65 grafik_pencar()

```

Gambar 13.54 Kode server.r

Simpan **ui.r** dan **server.r** dalam satu folder. Kemudian jalankan aplikasi dengan meng-*click* **Run App**.

Berdasarkan Gambar 13.52, kode R

```
sliderInput("ukuran_titik", label="Ukuran Titik", min=0, max=10, value=3, step=0.2),
```

bertujuan untuk menampilkan tampilan seperti pada Gambar 13.55, yang bertujuan untuk mengatur besar kecil ukuran titik.

Gambar 13.55

Kode R

```
radioButtons("warna_titik", "Warna Titik", c("Merah"="red", "Biru"="blue",  
"Hijau"="green", "Kuning"="yellow", "Pink"="pink",  
"Ungu"="purple", "Orange"="orange",  
"Abu-Abu"="grey", "Hijau Gelap"="darkgreen", "Merah Gelap"="darkred", "Biru  
Gelap"="darkblue",  
"Hitam"="black"), inline = TRUE ),
```

bertujuan untuk menampilkan tampilan seperti pada Gambar 13.56, yang bertujuan untuk mengatur warna titik.

Gambar 13.56

Kode R

```
radioButtons("bentuk", "Bentuk", c("I"="21",  
"II"="22", "III"="23", "IV"="24", "V"="25"), inline = TRUE ),
```

Bertujuan untuk menampilkan tampilan seperti pada Gambar 13.57, yang bertujuan untuk mengatur bentuk (titik, segiempat, segitiga, dan sebagainya).

Gambar 13.57

13.8 Dasar-dasar Shiny

Shiny merupakan aplikasi berupa halaman web(UI) yang terhubung dengan sebuah komputer yang menjalankan aplikasi R(Server). Pengguna dapat memanipulasi, mengubah tampilan UI yang akan mengakibatkan server mengupdate tampilan UI dengan menjalankan sintak R.

Gambar 13.58

13.9 Template

Untuk mempelajari shiny, langkah pertama yang dapat dilakukan adalah memulai menulis dengan template yang sudah ada. Jalankan sintak berikut di command line R.

```
library(shiny)
ui <- fluidpage()
server <- function(input,output){}
shinyApp(ui = ui, server = server)
```

Berikut penjelasan dari sintaks di atas,

- ui** - fungsi R yang menghimpun komponen-komponen HTML untuk tampilan depan aplikasi
- server** - sebuah fungsi dengan instruksi-instruksi yang akan merekayasa objek-objek yang ditampilkan pada UI.
- shinyApp** - mengombinasikan **ui** dan **server** menjadi sebuah aplikasi.

13.10 Membuat Sebuah Aplikasi

Selanjutnya, akan diuraikan bagaimana menyempurnakan sintak di sub-bab 2.2. Pada bagian UI, telah dilakukan penambahan fungsi masukan ***Input()**, dan penambahan keluaran dengan fungsi ***Output()**. Untuk memberi komando kepada server supaya merubah (*render*) tampilan output, maka diberi perintah di bagian Server,

1. Mengacu ke *outputs* dengan `output$<id>`
2. Mengacu ke *inputs* dengan `input$<id>`

Sebelum outputnya di save, sintaks2 yang merupakan *output* di *wrap* dalam sintak **render*()**

Gambar 13.59

Simpan template anda sebagai **app.R**. Selain itu, kita juga dapat memisahkan template menjadi dua file terpisah. Namakan **ui.R** dan **server.R**. **ui.R** berisi segala sesuatu yang dibutuhkan untuk membangun tampilan, sementara **server.R** berisi fungsi-fungsi yang dibutuhkan untuk mengubah tampilan ketika diberi perubahan inputan.

Gambar 13.60

13.11 Inputs

Mengumpulkan nilai-nilai dari pengguna(*user*) dengan **input\$<inputId>**. Nilai-nilai dari input merupakan *reactive*, digambarkan berikut:

Gambar 13.61

13.12 Outputs

Fungsi **render*()** dan ***Output()** berjalan bersama untuk menampilkan output R ke UI.

Gambar 13.62

13.13 Reactivity

Nilai-nilai *reactive* berjalan bersama-sama dengan fungsi *reactive*. Mengakses nilai *reactive* dari argument fungsi-fungsi di bawah ini untuk menghindari bunyi error **Operation not allowed without an active reactive context**.

Gambar 13.63

13.13.1 Membuat Reactive Value

Pada shiny, reactive merupakan cara untuk membuat variabel pada bagian server. Fungsi ***Input()** menghasilkan reactive value yang tersimpan di `input$<inputId>`. Selain menggunakan `*input`, cara lain mendefinisikan variabel di server yakni menggunakan **reactiveValues()**, yang akan menghasilkan list variabel yang kita inginkan.

```
#Create Your Own Reactive Values
ui <- fluidPage(
  textInput("a",""),
  textOutput(outputId = "siang")
)

server <- function(input,output){
  rv <- reactiveValues()
  rv$number <- 5
  rv$saya <- 10

  output$siang <- renderText({
    paste(input$a,rv$saya, rv$number)
  })
}

shinyApp(ui = ui, server = server)
```

Gambar 13.64

13.13.2 Trigger Arbitrary Code

Kita dapat melihat hasil dari perintah yang dibuat supaya muncul di console R, seperti di bawah ini.

```
ui <- fluidPage(
  textInput("a",""),
  actionButton("go","Klik Ini")
)

server <- function(input,output){
  observeEvent(input$go,
    print(input$a))

  values <- reactiveValues(A = 1)
  obsB <- observe({
    print(values$A + 1)
  })

  obsC <- observe(quote({
    print(values$A + 2)
  }), quoted = TRUE)
}

shinyApp(ui, server)
```

Gambar 13.65

13.13.3 Delay Reaction

Untuk menghasilkan suatu program yang memiliki tampilan(*interface*) yang interaktif, kita dapat menggunakan fitur *delay reaction*. Fitur ini akan menjalankan code setelah menunggu *action* dari user.

13.13.4 Modularize Reactions

Bedanya **reactiveValues()** dengan **reactive()** ialah hasil tipe data yang dihasilkan. Jika **reactiveValues()** akan menghasilkan list, sementara **reactive()** dapat digunakan untuk membuat perintah-perintah yang akan digunakan, sehingga tidak perlu diulang pada setiap badan code Output.

```

ui <- fluidPage(
  textInput("a",""),
  textInput("z",""),
  textOutput("b")
)

server <- function(input, output){
  re <- reactive({
    paste(input$a, input$z)
  })

  output$b <- renderText({
    re()
  })
}

shinyApp(ui = ui, server = server)

```

Gambar 13.66

13.13.5 Render Reactive Output

Fitur ini akan menghasilkan object yang akan ditampilkan. Sesuai namanya yakni *render*, akan menjalankan ulang code untuk menghasilkan ulang object kapanpun nilai reactive berubah.

```

ui <- fluidPage(
  textInput(inputId = "a",label = "Masukkan Teks"),
  textOutput("b")
)

server <- function(input, output){
  output$b <- renderText({
    input$a
  })
}

shinyApp(ui = ui, server = server)

```

Gambar 13.67

13.14 Layout

Ada banyak tampilan yang dapat digunakan pada shiny. Jika membicarakan tampilan, maka fokus kita tertuju pada code di bagian UI. Inputan-inputan yang digunakan dinamakan element. Kita dapat mengkombinasikan beberapa element menjadi satu elemen dengan fungsi Panel, seperti contoh berikut,

Gambar 13.68

Selain `wellpanel`, ada banyak yang penulis sarankan pembaca untuk mencoba sendiri hal-hal berikut ini. Cukup ketikkan di `help Rstudio`.

<code>absolutePanel()</code>	<code>inputPanel()</code>	<code>tabPanel()</code>
<code>conditionalPanel()</code>	<code>mainPanel()</code>	<code>tabsetPanel()</code>
<code>fixedPanel()</code>	<code>navlistPanel()</code>	<code>titlePanel()</code>
<code>headerPanel()</code>	<code>sidebarPanel()</code>	<code>wellPanel()</code>

Gambar 13.69

Kita juga dapat mengatur panel dan element dengan menggunakan fungsi `layout`. Dengan menjadikan elemen-elemen sebagai argumen dari fungsi `layout`.

Gambar 13.70

13.15 Membuat Kalkulator Sederhana

Pada bagian ini akan dibahas langkah-langkah membuat kalkulator yang sederhana seperti gambar di bawah ini,

Kalkulator Sederhana

Nilai 1

Nilai 2

Pilih Operator

☒ +
☐ -
☐ *
☐ :

Hasil

Gambar 13.71

13.15.1 Mendesain UI

Untuk membuat program kalkulator seperti gambar di atas, elemen-elemen yang dibutuhkan ialah,

- Model Layout dengan *sidebarPanel* dan *MainPanel*
- Elemen *input* yakni *numericInput*

- Kedua *input*-an di jadikan satu dengan menggunakan *wellPanel*
- Untuk pilihan operator perhitungan, digunakan element *radioButton*
- Supaya ada interaksi antara pengguna dan program, maka digunakan element *actionButton*

13.15.2 Menyusun Sintaks

Sintaks pada bagian server disajikan di bawah berikut ini,

```
ui <- fluidPage(
  h1("Kalkulator Sederhana"),
  sidebarLayout(
    sidebarPanel(
      wellPanel(
        numericInput("masukan1", "Nilai 1", value = 50),
        numericInput("masukan2", "Nilai 2", value = 100),
        radioButtons(inputId = "operator", label = "Pilih operator",
          choices = c("+", "-", "*", ":") ),
        actionButton("a", "Hitung")),
    mainPanel(
      h3("Hasil"),
      textOutput("ab")
    )
  )
)
server <- function(input, output){
  ide <- eventReactive(
    input$a,{
      A = input$masukan1
      B = input$masukan2
      if(input$operator == "+"){
        A + B
      }else if(input$operator == "-"){
        A - B
      }else if(input$operator == "*"){
        A*B
      } else{ A / B}
    })
  output$ab <- renderText({
    ide()
  })
}
shinyApp(ui, server)
```

Gambar 13.72

Referensi Bacaan

- ⇒ Andrie de Vries dan Joris Meys, 2012, *R for Dummies*, John Willey & Sons
- ⇒ Garrett Grolemund, 2014, *Hands-On Programming with R*, O'Reilly Media, Inc.,
- ⇒ Nina Zumel dan John Mount, 2014, *Practical Data Science with R*, Manning Publications
- ⇒ Norman Matloff, 2009, *The Art of R Programming*
- ⇒ Phil Spector, 2008, *Data Manipulation with R*, Springer
- ⇒ Prana Ugiana Gio dan Dasapta Erwin Irawan, 2016, Belajar Statistika dengan R
- ⇒ W. John Braum dan Duncan J. Murdoch, 2007, *A First Course in Statistical Programming with R*, Cambridge University Press
- ⇒ <https://shiny.rstudio.com/>
- ⇒ <http://deanattali.com/>
- ⇒ <http://langtest.jp/shiny/npt/>