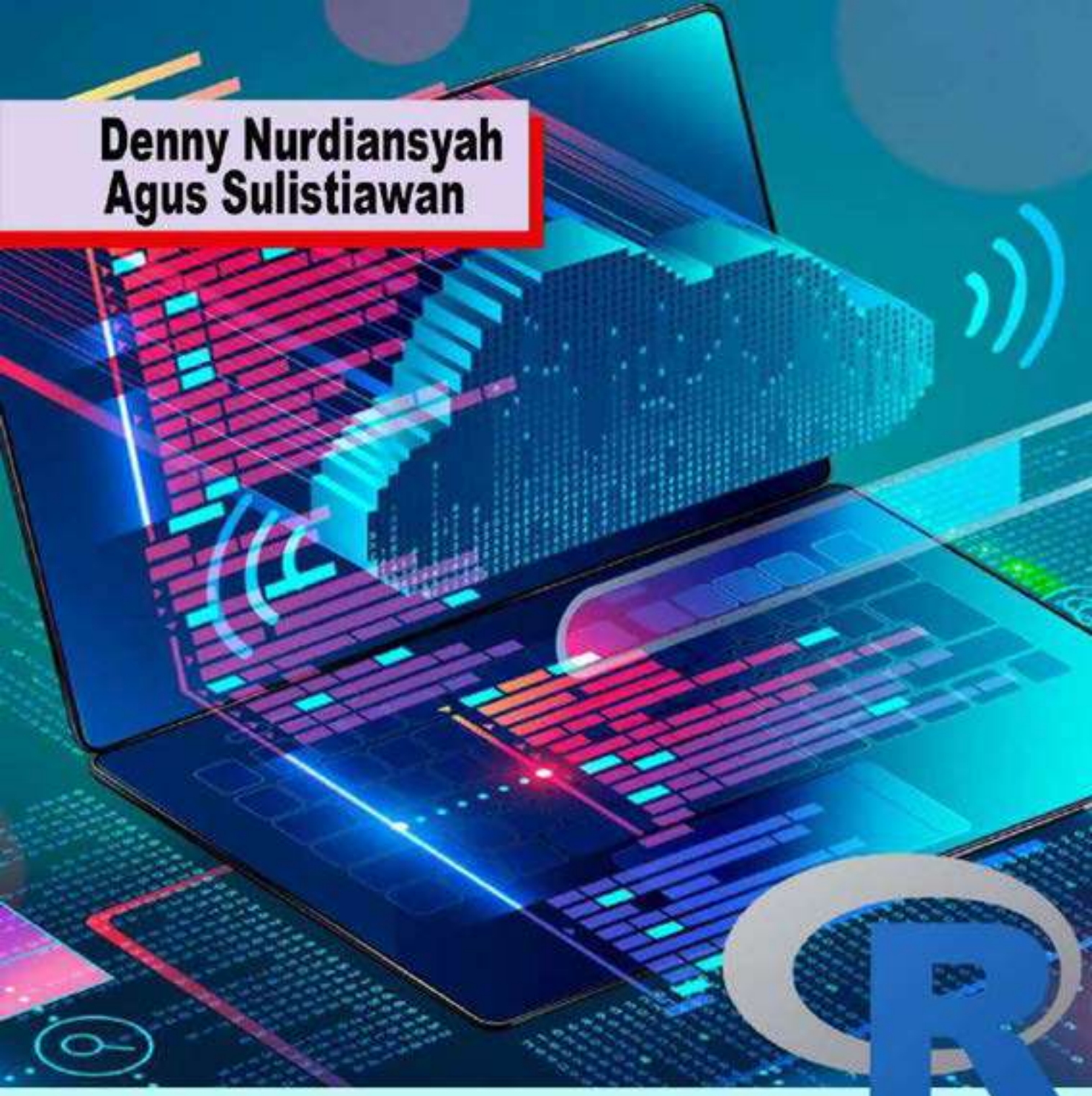
Denny Nurdiansyah
Agus Sulistiawan



DASAR PEMROGRAMAN KOMPUTER

Dengan *Open Source Software R*

(Untuk Bidang Sains dan Teknologi)

DASAR PEMROGRAMAN KOMPUTER DENGAN *OPEN SOURCE SOFTWARE R*
(Untuk Bidang Sains dan Teknologi)

Sejarah dan perkembangan software R menjadi sangat menarik karena software R hadir dengan bahasa pemrograman yang mudah dipahami. Kemampuan menggunakan bahasa R menjadi salah satu kemampuan kunci bagi seorang Data Scientist, serta kemampuan ini juga menjadi kemampuan yang dibayar mahal pada seorang Finance and Risk Analytics Manager. Software R adalah software yang bebas lisensi, sehingga pengguna mudah mendapatkannya di website resminya.

Buku ini dirancang dalam beberapa materi, hal ini dimaksudkan untuk mempermudah mahasiswa, praktisi, dan peneliti dalam mendalami pengetahuan dari konsep dasar yang dibahas dalam Dasar Pemrograman Komputer dengan bahasa R disertai pemanfaatan dan penerapan paket program R yang sudah built-in ataupun paket install. Penyajian buku ini diupayakan dalam bentuk format yang sangat sederhana agar pembaca mudah memahami dan menerapkan pengetahuan bahasa R dengan hasil pengetahuan yang maksimal.

Isi dalam buku Ini adalah sebagai berikut:

Penerbit : CV. AA. RIZKY
Alamat : Jl. Raya Ciruas Petir,
Puri Citra Blok B2 No. 34 Pipitan
Kec. Walantaka - Serang Banten
E-mail : aa.rizkypress@gmail.com
Website : www.aarizky.com

ISBN 978-623-405-212-1

DASAR PEMROGRAMAN KOMPUTER DENGAN *OPEN SOURCE SOFTWARE R*

(Untuk Bidang Sains dan Teknologi)

Undang-undang No.19 Tahun 2002 Tentang Hak Cipta
Pasal 72

1. Barang siapa dengan sengaja melanggar dan tanpa hak melakukan perbuatan sebagaimana dimaksud dalam pasal ayat (1) atau pasal 49 ayat (1) dan ayat (2) dipidana dengan pidana penjara masing-masing paling sedikit 1 (satu) bulan dan/atau denda paling sedikit Rp.1.000.000,00 (satu juta rupiah), atau pidana penjara paling lama 7 (tujuh) tahun dan/atau denda paling banyak Rp.5.000.000.000,00 (lima miliar rupiah).

2. Barang siapa dengan sengaja menyiarkan, memamerkan, mengedarkan, atau menjual kepada umum suatu ciptaan atau barang hasil pelangaran hak cipta terkait sebagai dimaksud pada ayat (1) dipidana dengan pidana penjara paling lama 5 (lima) tahun dan/atau denda paling banyak Rp.500.000.000,00 (lima ratus juta rupiah)

DASAR PEMROGRAMAN KOMPUTER DENGAN OPEN SOURCE SOFTWARE R

(Untuk Bidang Sains dan Teknologi)

Denny Nurdiansyah
Agus Sulistiawan

PENERBIT:
CV. AA. RIZKY
2023

DASAR PEMROGRAMAN KOMPUTER DENGAN OPEN SOURCE SOFTWARE R
(Untuk Bidang Sains dan Teknologi)

Penulis:
Denny Nurdiansyah
Agus Sulistiawan

Desain Cover & Tata Letak:
Tim Kreasi CV. AA. Rizky

Cetakan Pertama, Maret 2023

Penerbit:
CV. AA. RIZKY
Jl. Raya Ciruas Petir, Puri Citra Blok B2 No. 34
Kecamatan Walantaka, Kota Serang - Banten, 42183
Hp. 0819-06050622, Website : *www.aarizky.com*
E-mail: aa.rizkypress@gmail.com

Anggota IKAPI
No. 035/BANTEN/2019

ISBN : 978-623-405-212-1
x + 152 hlm, 23 cm x 15,5 cm

PRAKATA

Alhamdulillah wa syukurillah, buku ini dapat tersusun sesuai dengan rencana yang diharapkan. Buku ini merupakan buku teks yang dibuat sebagai buku terapan tentang Dasar Pemrograman Komputer dengan pemrograman bahasa R yang mana penerapannya dengan software R yang bebas lisensi. Buku ini disusun untuk memenuhi kebutuhan perkuliahan bagi mahasiswa dan implementasi bagi praktisi dan peneliti di lingkungan Data Analitik, Komputasi Statistik, dan Penelitian Ilmiah, terutama yang mempelajari pemahaman bahasa pemrograman R dalam rangka pembuatan proyek program komputer dengan bahasa R.

Buku ini dirancang dalam beberapa materi, hal ini dimaksudkan untuk mempermudah mahasiswa, praktisi, dan peneliti dalam mendalami pengetahuan dari konsep dasar yang dibahas dalam Dasar Pemrograman Komputer dengan bahasa R disertai pemanfaatan dan penerapan paket program R yang sudah built-in ataupun paket install. Penyajian buku ini diupayakan dalam bentuk format yang sangat sederhana agar pembaca mudah memahami dan menerapkan pengetahuan bahasa R dengan hasil pengetahuan yang maksimal.

Kritik dan saran sangat penulis butuhkan untuk penulisan buku di edisi berikutnya. Mohon bisa disampaikan melalui email nur.denny@gmail.com. Akhir kata dengan segala kekurangan dan kerendahan hati, penulis mengucapkan semoga buku ini bermanfaat bagi para pembaca terutama mahasiswa, praktisi, dan peneliti.

Surabaya, Maret 2023

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1

PENGENALAN SOFTWARE R

A. Deskripsi singkat

Materi pada bab ini merupakan pengenalan software R untuk para pembaca dengan memberikan pemaparan materi mengenai sejarah dan perkembangan software R, serta kebutuhan science data terkait software R. Pada bab ini juga diberikan contoh penerapan bagaimana cara mendapatkan dan menginstall software R di dalam sistem operasi Windows. Diharapkan pembaca dapat menjelaskan pengetahuan tentang software R dan proses instalasinya. Materi pada bab ini berguna untuk awalan kerja dalam studi pemrograman bahasa R.

B. Tujuan Pembelajaran

Pada bab ini, diberikan tujuan pembelajaran agar pembaca mampu menjelaskan bagaimana:

1. sejarah dan perkembangan software R.
2. kebutuhan data science terkait software R.
3. cara memperoleh dan menginstal software R.

C. Penyajian Materi

1. Sejarah dan Perkembangan Software R

   Perkembangan statistika tidak bisa lepas dari perkembangan software statistik. Penerapan statistika menjadi lebih mudah dan nyata dengan beberapa software statistik baik software komersil maupun software gratis (open source software) (Suhartono, 2008). Beberapa software statistik komersil yang sering digunakan orang Indonesia adalah Eviews, Minitab, SAS, SPSS, S-Plus, Stata, dan lainnya; sedangkan software statistik gratis bisa diwakili oleh JASP, SOFA, SCI Labs, GNU PSPP, Jamovi, dan R. Kendala pengguna software statistik komersil jelas pengguna tidak

bebas lisensi dan harganya mahal, sehingga sering terjadi penyalahgunaan lisensi software yang berujung pada denda lisensi yang sangat besar. Dengan demikian, perlu adanya alternatif software yang bebas lisensi seperti pada software R.

Software R menghasilkan syntax pemrograman komputer dalam bahasa R yang mana merupakan bahasa pemrograman tingkat tinggi untuk analisis data dan grafik. Bahasa R dipengaruhi oleh dua bahasa yang sudah ada sebelumnya, yaitu: bahasa S yang dikenalkan oleh Becker, Chambers, dan Wilks tahun 1988, dan bahasa Scheme yang dibuat Sussman tahun 1970-an. Bahasa R sangat mirip dengan tampilan bahasa S tetapi ilmu dan penerapan yang mendasari berasal dari bahasa Scheme (Crawley, 2013). Pada tahun 1992, bahasa R diperkenalkan ke dunia untuk pertama kalinya di Universitas Auckland, New Zealand. Bahasa R saat ini dikembangkan oleh R Core Team dengan dukungan para ahli statistik lain (pengguna R) dalam hal kontibusi kode, laporan bug, dan dokumentasi (Suhartono, 2008). Bahasa R merupakan bahasa pemrograman yang kuat untuk pekerjaan statistik, analisis data, analisis numerik, dan lainnya yang mana memiliki karakteristik utama yaitu kuat, stabil, gratis, dapat diprogram, perangkat lunak dengan sumber terbuka, dan diarahkan untuk visualisasi data (Sarmento & Costa, 2017). Bahasa R adalah bahasa pemrograman yang digunakan dalam lingkungan data analitik, komputasi statistik, dan penelitian ilmiah yang mana prosesnya melibatkan serangkaian tahapan meliputi: mengambil, membersihkan, menganalisis, memvisualisasikan, dan mempresentasikan data (Bell, 2020).

2. Kebutuhan Data Science terkait Software R

Kebutuhan seorang Ilmuan Data (Data Scientist) sangat melekat atau tidak bisa jauh dari beberapa software yang digunakannya, terutama pada software Statistika dan Big Data. Berikut diberikan contoh peralatan data science berupa

software-software yang dikelompokkan berdasarkan tujuan penggunaaannya seperti pada Gambar 1.1 .

Gambar 1.1 Pengelompokkan software dalam Data Scientist

Peranan software R dalam data science sangat kuat terutama dalam penerapan Data Cleaning, Data Mining, dan Data Analysis. Di samping itu kemampuan (skill) menggunakan software R menjadi penting dalam beberapa penerapan data science sebagaimana ditunjukkan dalam kebutuhan skills terhadap beberapa pekerjaan pada Tabel 1.1 (Miller & Hughes, 2017).

Tabel 1.1 Kebutuhan Skills Terhadap Beberapa Pekerjaan

Pekerjaan	***Key Skills***	***High-Paying Skills***
Data Scientist	Apache Hadoop **R** Python	*Database Administration* *Object-Oriented*

	Machine Learning *Data Science*	*Analysis and Design* *Quantitative Analysis* *Database Schemas* *Pattern Recognition*
Data Engineer	Python JAVA Apache Hadoop *Big Data* *Data Engineering*	PIG Apache Flume *Predictive Models* Oozie *Spark Programming*
Finance and Risk Analytics Manager	SQL *Project Management* *Forecasting dan Financial Modeling* *Financial Analysis and Planning* *Risk Management*	**R** *Project Management* *Data Warehousing* *Mergers and Acquisition* MATLAB

Kemampuan penggunaan software R menjadi salah satu kemampuan kunci bagi seorang Data Scientist, serta kemampuan ini juga menjadi kemampuan yang dibayar mahal pada seorang Finance and Risk Analytics Manager. Kemampuan bahasa R ini menjadi daya tarik bagi masyarakat dunia dalam mempelajarinya. Dalam indeks PYPL (PopularitY of Programming Language) pada tahun 2022, bahasa R menempati peringkat ke-7 dalam sepuluh besar bahasa pemrograman yang tutorial penggunaannya paling dicari oleh programmer dan mahasiswa di seluruh dunia melalui website Google tepatnya Google Trends yang ditunjukkan pada Gambar 1.2 (Carbonnelle, 2022).

Worldwide, Nov 2022 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	28.44 %	+2.5 %
2		Java	17.03 %	-0.3 %
3		JavaScript	9.51 %	+0.3 %
4		C#	7.05 %	-0.1 %
5		C/C++	6.51 %	-0.4 %
6		PHP	5.32 %	-1.0 %
7		R	4.12 %	+0.1 %
8	↑↑↑	TypeScript	2.89 %	+1.3 %
9	↓	Objective-C	2.37 %	+0.1 %
10		Swift	2.11 %	+0.4 %

Gambar 1.2
Peringkat Bahasa Pemrograman Berdasarkan Indeks PYPL

3. Cara Memperoleh dan Menginstall *Software* R

Pada *CRAN-archive* (*The Comprehensice R Archive Network*), *software* R dapat diperoleh secara gratis di website resminya yaitu *https://cran.r-project.org/ mirrors.html.* Jaringan *CRAN-archive* tersedia di URL ini dengan pilihan lokasi yang dekat dengan pengguna. Diberikan beberapa statistik tentang status *mirror* yang memuat halaman utama, rilis versi Windows, dan rilis lama versi Windows (Cran.R-Project, 2022). Berikut diberikan link download *software* R pada sistem operasi Windows:

a. *Software* R 4.1.3 untuk Windows dengan sistem operasi 32 bit. *https://cran.r-project.org/bin/windows/base/old/4.1.3/*
b. *Software* R 4.2.2 untuk Windows dengan sistem operasi 64 bit. *https://cran.r-project.org/bin/windows/base/old/4.2.2/*

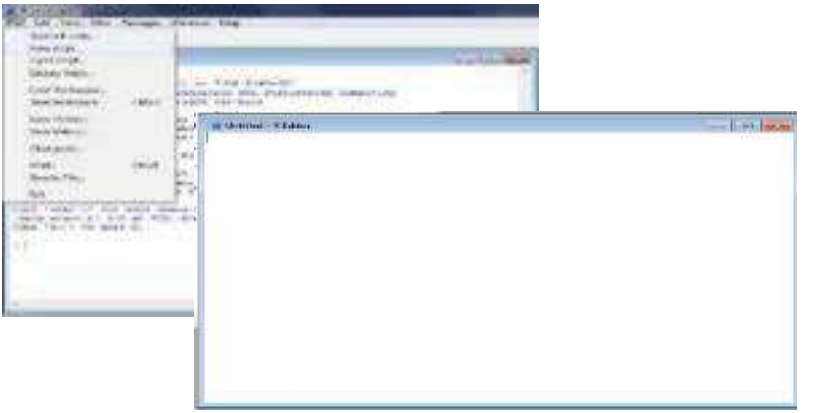
Keunggulan dan kekuatan dari R terletak dari banyaknya paket program baik yang sudah *built-in* maupun yang harus diinstal secara online ataupun offline. Pada akhir tahun 2022, sudah terdapat repositori paket CRAN sebanyak 19.097 paket yang tersedia (Cran R-Project, 2022).

Cara menginstal *software* R pertama dilakukan dengan mengunduh *software* tersebut pada link yang sudah diberikan, kemudian unduhan akan datang dalam bentuk file yang dapat dieksekusi (*.exe*), sehingga pengguna hanya perlu mengklik dua kali untuk mengeksekusi proses instalasi. Selama instalasi, pengguna bisa memilih pengaturan *default* dan cukup memilih tombol *Next* sampai proses selesai. Setelah penginstalan selesai, eksekusi *software* R dapat dilakukan melalui *R shortcut* di Dekstop atau melalui tombol *Start -> All Programs*. Pengguna akan dihadapkan pada tampilan *R Console* seperti Gambar 1.3.

Gambar 1.3 Tampilan R Console pada Software R

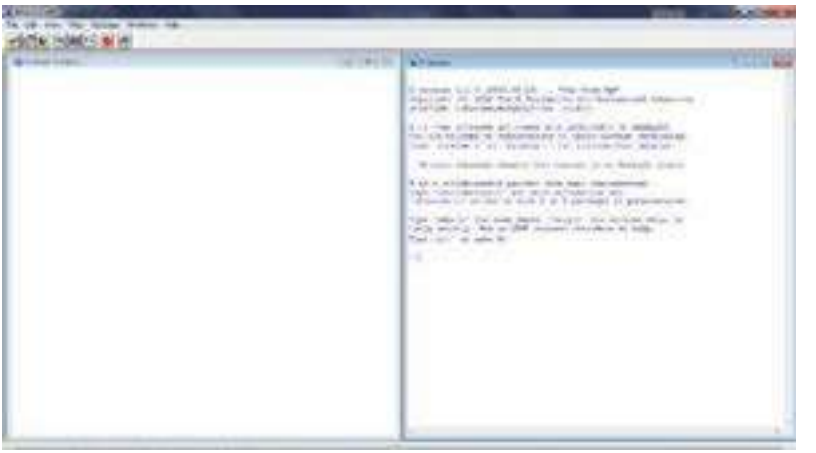
Pada Gambar 1.3, diberikan informasi software yaitu software R versi 4.1.3 yang dipublikasikan oleh The R Foundation for Statistical Computing pada tanggal 10 Maret 2022 dengan judul “One Push-Up” untuk Windows sistem operasi 32 bit. R Console ini digunakan untuk mengeksekusi setiap syntax pemrograman dalam Bahasa R. Untuk menyimpan semua syntax R, diberikan jendela skrip pemrograman dalam R Editor. Pengguna bisa membuat

jendela skrip baru dengan mengklik toolbar atas yaitu File -> New Script pada Gambar 1.4.



Gambar 1.4 Tampilan R-Editor pada *software* R

Pada buku ini dibiasakan programmer menempatkan R Editor disebelah kiri dan R Console disebelah kanan agar proses pembuatan program komputer menjadi lebih nyaman dan mudah seperti pada Gambar 1.5.



Gambar 1.5 Tampilan penenpatan R-Editor dan R-Console

D. Rangkuman

Sejarah dan perkembangan software R menjadi sangat menarik karena software R hadir dengan bahasa pemrograman yang mudah dipahami. Kemampuan menggunakan bahasa R

menjadi salah satu kemampuan kunci bagi seorang Data Scientist, serta kemampuan ini juga menjadi kemampuan yang dibayar mahal pada seorang Finance and Risk Analytics Manager. Software R adalah software yang bebas lisensi, sehingga pengguna mudah mendapatkannya di website resminya. Proses mengunduh dan mengistall software R sangat mudah, serta didukung paket program built-in dan install yang sangat banyak sebagai add-on software. Hal ini sangat membantu pengguna dalam lingkungan data analitik, komputasi statistik, dan penelitian ilmiah.

E. Soal Latihan

1. Silahkan mengunduh software R berdasarkan link website yang telah diberikan !
2. Silahkan menginstall software R sesuai tahapan yang disampaikan dalam Bab ini !

F. Sumber Referensi

Bell, D. (2020). *R Programming: A Step-by-Step Guide for Absolute Beginners (2nd Ed).* KDP Amazon Publishing.

Carbonnelle, P. (2022). *PYPL Index*. GitHub. https://pypl.github.io/PYPL.html

Cran.R-Project. (2022). *CRAN Mirrors*. https://cran.r-project.org/mirrors.html

Cran R-Project. (2022). *Contributed Packages*. https://cran.r-project.org/web/packages/

Crawley, M. J. (2013). *The R Book (2nd Ed*). John Wiley & Sons.

Miller, S., & Hughes, D. (2017). *The Quant Crunch: How The Demand For Data Science Skills Is Disrupting The Job Market*. Burning Glass Technologies.

Sarmento, R., & Costa, V. (2017). *Comparative Approaches to Using R and Python for Statistical Data Analysis*. IGI Global.

Suhartono. (2008). *Analisis Data Statistik Dengan R*. Lab. Statistik Komputasi ITS

BAB 2 JENIS DATA OBJEK SOFTWARE R

A. Deskripsi singkat

Materi pada bab ini merupakan pendalaman materi terkait jenis data objek yang ada di software R dengan memberikan pemaparan materi tentang pengenalan tipe data dan mode data, penggunaan Vektor dan Matriks beserta operator aritmatikanya, dan penggunaan Data Frame, Factor, Array, dan List. Bahasan ini diberikan dengan penjelasan singkat dan contoh penerapan bagaimana membuat syntax yang benar dalam pemrograman khususnya penggunaan Vektor, Matriks, Data Frame, Factor, Array, dan List. Pada bab ini diharapkan pembaca dapat memahami dan menerapkan bagaimana memanfaatkan penggunaan dari jenis data objek. Materi pada bab ini berguna untuk mengetahui jenis data objek apa saja yang ada di lingkungan pemrograman bahasa R.

B. Tujuan Pembelajaran

Pada bab ini, diberikan tujuan pembelajaran agar pembaca mampu menjelaskan dan menerapkan bagaimana:

1. penggunaan tipe data dan mode data.
2. penggunaan vektor dan matriks.
3. penggunaan data frame dan factor.
4. penggunaan array dan list.

C. Penyajian Materi

1. Penggunaan Tipe Data dan Mode Data

   Dalam software R, sifat data ditentukan oleh tipe data dan mode data. Tipe data yang dikenali oleh bahasa R, antara lain: Vektor, Matriks, List, Array, Data Frame, dan Factor (Bell, 2020). Mode Data yang digunakan dalam R ada empat

macam yang mana contoh penerapannya dapat diberikan di dalam R Console, sebagai berikut:

a. Numeric

Ketika pengguna mengetikkan angka 88, maka angka ini dikenali dengan mode data numerik, kemudian diberikan vektor yang berisi tujuh bilangan prima pertama yang juga memiliki mode data numerik. Vektor ini bisa disimpan dalam variabel prima1 yang mana secara automatis akan menjadi tipe data vektor yang bernama prima1. Sebagai tambahan, diberikan contoh penggunaan fungsi is.numeric untuk menanyakan apakah variabel prima1 berisi data yang memiliki mode data numerik atau bukan.

```
> 88
[1] 88
> c(2,3,5,7,9,11,13)
[1]  2  3  5  7  9 11 13
> prima1 = c(2,3,5,7,9,11,13)
> prima1
[1]  2  3  5  7  9 11 13
> is.numeric(prima1)
[1] TRUE
```

b. Complex

Diberikan bilangan 2+3i adalah bilangan kompleks yang bukan merupakan bilangan numerik. Hal ini bisa dibuktikan dengan penerapan fungsi is.numeric dan is.complex pada bahsaan ini. Bilangan kompleks di software R dapat diambil nilai realistisnya dengan fungsi Re, sedangkan nilai imajinernya diambil dengan fungsi Im.

```
> 2+3i
[1] 2+3i
> is.numeric(2+3i)
[1] FALSE
> is.complex(2+3i)
[1] TRUE
> a = 2+3i
> Re(a)
[1] 2
> Im(a)
[1] 3
```

Diberikan juga contoh penggunaan fungsi sqrt di software R untuk menghitung akar kuadrat pada suatu bilangan. Terlihat sqrt(4) atau √4 jelas sama dengan 2, namun kenapa sqrt(-4) atau √(-4) tidak dikenali. Hal ini bisa diatasi dengan merubah mode data numerik menjadi mode data kompleks dengan fungsi as.complex, sehingga diperoleh √(-4) sama dengan 2i. Disamping itu, bilangan -4 yang diubah menjadi mode data kompleks akan bernilai -4+0i.

```
> sqrt(4)
[1] 2
> sqrt(-4)
[1] NaN
Warning message:
In sqrt(-4) : NaNs produced
> sqrt(as.complex(-4))
[1] 0+2i
> as.complex(-4)
[1] -4+0i
```

c. Logika

Dalam software R, variabel T dan F ditetapkan menjadi suatu nilai logika yang masing-masing bernilai TRUE dan FALSE. Misalkan vektor A berisi data dengan mode data logika bisa dirubah menjadi mode data numerik dan karakter masing-masing dengan fungsi as.numeric dan as.character . Dengan demikian, numerik untuk T sama dengan 1 dan numerik untuk F adalah 0. Di sisi lain, semua mode data yang dirubah menjadi karakter akan memiliki nilai karakter dengan tanda petik dua (“”).

```
> T
[1] TRUE
> F
[1] FALSE
> c(T,T,F,T,F,T)
[1]  TRUE  TRUE FALSE  TRUE FALSE  TRUE
> A = c(T,T,F,T,F,T)
```

```
> as.numeric(A)
[1] 1 1 0 1 0 1
> as.character(A)
[1] "TRUE"  "TRUE"  "FALSE" "TRUE"  "FALSE" "TRUE"
```

d. Character

Ketika kata Budi diketikkan di R Console, maka akan dikenali sebagai data objek yang tidak ditemukan. Agar kata Budi dikenali sebagai mode data karakter maka perlu diberikan tanda petik dua, sehingga penulisannya menjadi "Budi". Penulisan ini juga dapat diterapkan pada suatu vektor seperti pada vektor B. Sebagai tambahan, diberikan contoh penggunaan fungsi mode untuk mengetahui mode data dari suatu objek data.

```
> Budi
Error: object 'Budi' not found
> "Budi"
[1] "Budi"
> B = c("Saya","Ibu","Budi")
> B
[1] "Saya" "Ibu"  "Budi"
> mode(B)
[1] "character"
```

Penulisan objek data bisa saja diterima atau tidak dalam suatu syntax pemrograman R. Berikut diberikan contoh-contoh nama objek data yang valid dan yang tidak valid (Suhartono, 2008):

a. Nama objek data yang valid

Penulisan nama suatu objek data bisa huruf kecil (b) atau huruf besar (B). Bisa juga sebuah kata seperti Budi. Setelah kata ini bisa ditambahkan nomor, titik, dan garis bawah. Hal ini berlaku juga ketika ada dua kata atau lebih yang boleh ditambahkan nomor, titik, dan garis bawah seperti pada variabel Budi.2, Budi_3, Budi.Tobat.4, dan Budi_Tobat_5.

```
> b = -2
> B = -1
> Budi = 0
> Budi1 = 1
> Budi.2 = 2
> Budi_3 = 3
> Budi.Tobat.4 = 4
> Budi_Tobat_5 = 5
```

b. Nama objek data yang tidak valid

Penulisan nama suatu objek data tidak boleh diawali dengan angka (1Budi), operator aritmatika (-Budi), garis bawah (_Budi), dan operator penugasan (=Budi). Ketika ada dua kata atau lebih tidak boleh menggunakan operator aritmatika sebagai pemisahnya (Budi-Tobat).

```
> 1Budi = 1
Error: unexpected symbol in "1Budi"
> -Budi = 1
Error in -Budi = 1 : could not find function "-<-"
> _Budi = 1
Error: unexpected input in "_"
> =Budi = 1
Error: unexpected '=' in "="
> Budi-1 = 1
Error in Budi - 1 = 1 : could not find function "-<-"
> Budi-Tobat = 1
Error in Budi - Tobat = 1 : could not find function "-<-"
```

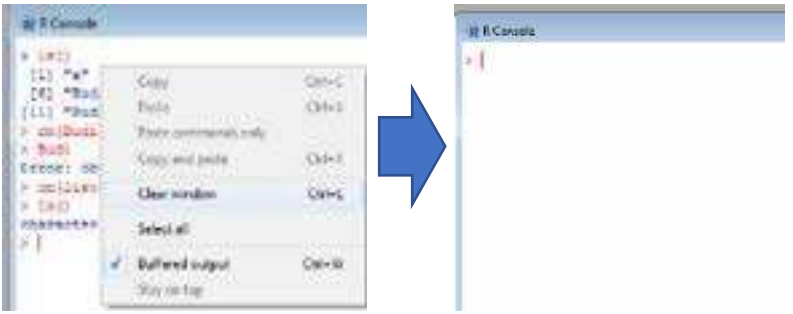
Dalam suatu bahasa pemrograman, ada dua istilah yaitu Konstanta dan Variabel. Variabel digunakan untuk penyimpanan data dan nilainya dapat berubah berdasarkan kebutuhan programmer (Bell, 2020), sedangkan konstanta memiliki nilai yang tetap atau tidak bisa berubah. Dalam bahasan software R ini, variabel bisa digambarkan dalam suatu objek data yang mana variabel bisa berupa Vektor, Matriks, Data Frame, Factor, Array, dan List. Ada beberapa nama yang tidak bisa diberikan pada suatu objek data, melainkan nama tersebut sudah ditetapkan sebagai konstanta di dalam software R. Konstanta ini memiliki sifat yang tidak bisa berubah nilainya yang mana bisa diberikan contoh penerapannya sebagai berikut.

```
> LETTERS
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S"
[20] "T" "U" "V" "W" "X" "Y" "Z"
> letters
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
[20] "t" "u" "v" "w" "x" "y" "z"
> pi
[1] 3.141593
> month.name
 [1] "January"   "February"  "March"     "April"     "May"       "June"
 [7] "July"      "August"    "September" "October"   "November"  "December"
> month.abb
 [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"
>
```

Dalam contoh ini, konstanta yang ada di software R adalah LETTERS, letters, pi, month.name, dan month.abb. Semua variabel yang sudah dijalankan dalan R Console dapat ditampilkan dengan fungsi ls yang mana variabel tersebut bisa dihapus dengan fungsi rm. Misalkan dilakukan penghapusan variabel Budi dengan syntax rm(Budi), maka variabel tersebut tidak akan ditemukan lagi. Ketika pengguna ingin menghapus semua variabel yang dijalankan di R Console, maka cukup dilakukan dengan mengetikkan syntax rm(list=ls()).

```
> ls()
 [1] "a"           "A"           "b"           "B"           "Budi"
 [6] "Budi.2"      "Budi.Anak.3" "Budi.Tobat.4" "Budi_3"      "Budi_Tobat_5"
[11] "Budi1"       "prima1"      "Senang"
> rm(Budi)
> Budi
Error: object 'Budi' not found
> rm(list=ls())
> ls()
character(0)
```

Kemudian misalkan pengguna ingin menghapus semua syntax di R Console, maka tinggal mengklik kanan cursor mouse dan pilih opsi Clear windows.



2. Penggunaan Vektor dan Matriks

Vektor merupakan suatu array atau himpunan bilangan dengan mode data apapun yang mana vektor adalah objek data yang paling dasar dalam bahasa R (Suhartono, 2008). Vektor merupakan array berdimensi satu yang seharusnya memiliki mode data yang sama (Bell, 2020). Sebelum masuk ke pembahasan vektor, perlu dipahami terkait Skalar. Skalar merupakan bilangan individual yang tidak benar-benar ada dalam R, melainkan bilangan tersebut diwujudkan dalam vektor satu elemen (Matloff, 2011).

```
> a = 8
> a
[1] 8
```

Dengan kata lain, skalar termasuk dalam vektor yang elemennya hanya satu. Cara membuat vektor sangat mudah karena tinggal mengumpulkan beberapa bilangan dan dimasukkan ke dalam fungsi concatenate (c) (Crawley, 2013), seperti berikut.

```
> c(1,2,3,4,5,6,7,8,9,0)
 [1] 1 2 3 4 5 6 7 8 9 0
> A = c(1,2,3,4,5,6,7,8,9,0)
> A
 [1] 1 2 3 4 5 6 7 8 9 0
```

Vektor bisa ditulis langsung atau disimpan ke dalam suatu variabel seperti pada variabel **A**. Vektor juga dapat

dibuat dengan operator titik dua (:), fungsi replikasi (**rep**), dan fungsi *sequence* (**seq**).

```
> A = 1:10
> A
 [1]  1  2  3  4  5  6  7  8  9 10
> B = rep(0,10)
> B
 [1] 0 0 0 0 0 0 0 0 0 0
> C = seq(1,10,2)
> C
[1] 1 3 5 7 9
```

Vektor yang berisi bilangan asli dari 1 sampai 10 dapat dibuat dengan *syntax* R, yaitu **1:10**. Vektor juga dapat dihasilkan dengan nilai nol sebanyak sepuluh kali dengan *syntax* **rep(0,10)**. *Syntax* ini biasa digunakan untuk nilai awalan sebelum masuk proses perulangan dalam pemrograman komputer. Disisi lain, *syntax* **seq(1,10,2)** menghasilkan vektor terurut mulai dari 1 sampai 10 dengan selisih atau beda 2, sehingga menghasilkan bilangan ganjil 1 sampai 9. Di samping itu, vektor bisa dibuat dari bermacam mode data yang diinginkan pengguna.

```
> c(1,4,5,T,F,T)
[1] 1 4 5 1 0 1
> c(1,4,5,T,F,2+3i)
[1] 1+0i 4+0i 5+0i 1+0i 0+0i 2+3i
> c(1,4,5,T,F,"Budi")
[1] "1"     "4"     "5"     "TRUE"  "FALSE" "Budi"
> c(1,4,5,T,F,2+3i,"Budi")
[1] "1"     "4"     "5"     "TRUE"  "FALSE" "2+3i"  "Budi"
```

Ketika bilangan dengan mode data numerik dan logika dimasukkan ke dalam vektor, maka akan dihasilkan vektor dengan mode data numerik. Pada hasil yang lain, ketika bilangan dengan mode data numerik dan kompleks dimasukkan ke dalam vektor, maka akan dihasilkan vektor dengan mode data kompleks. Disisi lain, ketika ada minimal satu bilangan dengan mode data karakter yang dimasukkan

ke dalam vektor, maka akan dihasilkan vektor dengan mode data karakter.

Indexing vektor bisa digunakan untuk mengenali nilai elemen-elemen di dalam vektor. Ada beberapa cara untuk melakukan *indexing* vektor, yaitu

```
> A = seq(2,20,2)
> A
 [1]  2  4  6  8 10 12 14 16 18 20
> A[4]
[1] 8
> A[2:4]
[1] 4 6 8
> A[c(2,5,7)]
[1]  4 10 14
> A[A>10]
[1] 12 14 16 18 20
> A[-6]
[1]  2  4  6  8 10 14 16 18 20
> A[-(2:4)]
[1]  2 10 12 14 16 18 20
> A[-c(2,5,7)]
[1]  2  6  8 12 16 18 20
```

Misalkan diberikan bilangan genap dari 2 sampai 20, maka nilai elemen vektor ke-4 adalah **8** yang mana diberikan dari *indexing* vektor dengan *syntax* yaitu **A[4]**. *Syntax* **A[2:4]** akan menampilkan nilai vektor pada elemen ke-2 sampai ke-4, yaitu **4, 6, dan 8**. Dengan menerapkan fungsi *concatenate* (**c**), pengguna bisa memilih nilai elemen pada vektor ke berapa saja yang ingin ditampilkan. Misalkan *syntax* **A[c(2,5,7)]** akan menampilkan nilai vektor pada elemen ke-2, ke-5, dan ke-7, yaitu **4, 10, dan 14**. *Indexing* vektor juga dapat dimasukkan operator logika, seperti *syntax* **A[A>10]** yang mana akan menampilkan nilai vektor pada elemen yang nilainya diatas 10, yaitu **12, 14, 16, 18, dan 20**. Pada *indexing* vektor ini, elemen di dalam vektor dapat dihapus dengan tanda negatif (**-**) pada elemen berapa nilai itu dihapus. Misalkan *syntax* **A[-6]** berarti menampilkan vektor A dengan

nilai elemen ke-6 yang dihapus (**12**), sehingga menjadi **2, 4, 6, 8, 10, 14, 16, 18, dan 20**. Sama halnya, *syntax* **A[-(2:4)]** yang menampilkan vektor A dengan penghapusan elemen ke-2 sampai ke-4, kemudian *syntax* **A[-c(2,5,7)]** juga dapat menampilkan vektor A dengan penghapusan elemen ke-2, ke-5, dan ke-7 saja.

Setiap elemen di dalam vektor bisa diberikan nama atau *header* dengan fungsi **names**.

```
> Nilai = c(80,70,60,90,50)
> Nilai
[1] 80 70 60 90 50
> names(Nilai) = c("A","B","C","D","E")
> Nilai
 A  B  C  D  E
80 70 60 90 50
> Nilai["D"]
 D
90
> names(Nilai) = NULL
> Nilai
[1] 80 70 60 90 50
> length(Nilai)
[1] 5
```

Misalkan diberikan vektor **Nilai** yang bisa diberikan nama **"A"** sampai **"E"** dengan bantuan fungsi **names**. Dengan demikian *indexing* vektor cukup dengan mengetikkan namanya saja seperti **Nilai["D"]** yang akan menampilkan nilai vektor **Nilai** yang memiliki nama **"D"**, yaitu **90**. Untuk menghapus nama vektor bisa menggunakan *syntax* **names(Nilai) = NULL**. Sebagai tambahan, diberikan contoh penerapan fungsi **length** yang berguna untuk menghitung banyaknya elemen di dalam suatu vektor. Setelah vektor, bahasan berikutnya adalah Matriks.

Matriks adalah kumpulan data yang direpresentasikan dalam bentuk array dua dimensi yang mana memiliki mode data yang sama dengan orde dimensi tertentu (Bell, 2020).

Matriks juga merupakan salah satu tipe objek data yang sering digunakan dalam pemrograman statistik (Suhartono, 2008). Secara teknik, matriks adalah suatu vektor yang memiliki atribut tambahan yaitu banyaknya baris dan banyaknya kolom (Matloff, 2011). Untuk membuat matriks, digunakan suatu vektor yang akan dimasukkan ke dalam fungsi **matrix**.

```
> vektor1 = c(2,3,5,7,9,11,13,17)
> matriks1 = matrix(vektor1,nrow=2,ncol=4)
> matriks1
     [,1] [,2] [,3] [,4]
[1,]    2    5    9   13
[2,]    3    7   11   17
> matriks2 = matrix(vektor1,nrow=2,ncol=4,byrow=T)
> matriks2
     [,1] [,2] [,3] [,4]
[1,]    2    3    5    7
[2,]    9   11   13   17
> matriks3 = matrix(vektor1,2,4,T)
> matriks3
     [,1] [,2] [,3] [,4]
[1,]    2    3    5    7
[2,]    9   11   13   17
```

Misalkan **vektor1** merupakan vektor yang berisi delapan bilangan prima pertama yang dimasukkan ke dalam fungsi **matrix** untuk membuat matriks yang diinginkan, seperti **matriks1**, **matriks2**, dan **matriks3**. Ada tiga macam cara untuk membuat matriks. Cara pertama ditunjukkan pada **matriks1** yang membuat matriks dengan banyaknya baris 2 (**nrow=2**) dan banyaknya kolom 4 (**ncol=4**) yang berisi nilai elemen-elemen **vektor1** yang diinputkan menurun. Jika pengguna bermaksud menginputkan **vektor1** secara mendatar maka digunakan cara kedua dengan menambahkan opsi **byrow=T** seperti pada **matriks2**. Cara ketiga adalah bentuk ringkas dari cara kedua, yaitu tanpa mengetikkan opsi **nrow**, **ncol**, dan **byrow** seperti pada **matriks3**.

```
> matriks3
     [,1] [,2] [,3] [,4]
[1,]    2    3    5    7
[2,]    9   11   13   17
> dim(matriks3)
[1] 2 4
> length(matriks3)
[1] 8
> mode(matriks3)
[1] "numeric"
```

Pada **matriks3**, dimensi matriks bisa diketahui dengan fungsi **dim**, dan banyaknya data dalam matriks bisa juga ditunjukkan oleh fungsi **length** seperti penerapannya pada vektor.

Indexing matriks bisa digunakan untuk mengenali nilai di dalam elemen-elemen matriks yang dua dimensi. Ada beberapa cara untuk melakukan *indexing* matriks, sebagai berikut.

```
> matriks3
     [,1] [,2] [,3] [,4]
[1,]    2    3    5    7
[2,]    9   11   13   17
> matriks3[2,3]
[1] 13
> matriks3[1,4]
[1] 7
> matriks3[2,]
[1]  9 11 13 17
> matriks3[,3]
[1]  5 13
> matriks3[,2:4]
     [,1] [,2] [,3]
[1,]    3    5    7
[2,]   11   13   17
> matriks3[,c(2,4)]
     [,1] [,2]
[1,]    3    7
[2,]   11   17
```

Indexing matriks pada *syntax* **matriks3[2,3]** akan menampilkan nilai vektor dari nilai elemen matriks baris ke-2 dan kolom ke-3, yaitu **13**. Sama halnya, pada *syntax* **matriks3[4,1]** akan menampilkan nilai vektor dari nilai elemen matriks baris ke-4 dan kolom ke-1, yaitu **7**.

Selanjutnya pada *syntax* **matrik3[2,]** akan menampilkan vektor dari nilai elemen matriks baris ke-2 saja, yaitu **9, 11, 13, dan 17**. Sejauh ini bisa disimpulkan bahwa elemen-elemen yang diambil dalam satu baris/kolom akan membentuk suatu vektor tertentu. Jika ingin membentuk suatu matriks seharusnya elemen-elemen yang diambil dari matriks haruslah lebih dari 1 kolom/baris seperti pada *syntax* **matriks3[,2:4]** atau **matriks3[,c(2,4)]**.

```
> matriks3
     [,1] [,2] [,3] [,4]
[1,]    2    3    5    7
[2,]    9   11   13   17
> matriks3[-2,]
[1] 2 3 5 7
> matriks3[,-3]
     [,1] [,2] [,3]
[1,]    2    3    7
[2,]    9   11   17
> matriks3[,-(2:4)]
[1] 2 9
> matriks3[,-c(2,4)]
     [,1] [,2]
[1,]    2    5
[2,]    9   13
```

Misalkan diberikan *syntax* **matriks3[-2,]** yang menampilkan **matrik3** tanpa baris ke-2, sedangkan *syntax* **matriks3[,-3]** berarti menampilkan **matrik3** tanpa kolom ke-3. Ketika diterapkan *syntax* **matriks3[,-(2:4)]**, maka akan ditampilkan **matrik3** tanpa kolom ke-2 sampai kolom ke-4, sehingga menghasilkan suatu vektor. Selanjutnya *syntax* **A[,-c(2,4)]** juga akan menampilkan **matrik3** tanpa kolom ke-2 dan ke-4 saja.

Penamaan setiap baris dan kolom pada matriks bisa diberikan masing-masing dengan fungsi **rownames** dan **colnames**.

```
> matriks3
     [,1] [,2] [,3] [,4]
[1,]    2    3    5    7
[2,]    9   11   13   17
> rownames(matriks3) = rownames(matriks3,do.NULL=FALSE,prefix="Nilai.")
> nama.mahasiswa = c("Agus","Adi","Ayu","Anggi")
> colnames(matriks3) = nama.mahasiswa
> matriks3
        Agus Adi Ayu Anggi
Nilai.1    2   3   5     7
Nilai.2    9  11  13    17
> rownames(matriks3) = NULL
> colnames(matriks3) = NULL
> matriks3
     [,1] [,2] [,3] [,4]
[1,]    2    3    5    7
[2,]    9   11   13   17
```

Misalkan **matriks3** diberinama baris dan kolomnya masing-masing dengan fungsi **rownames** dan **colnames**. Penamaan ini memberikan tampilan output yang lebih informatif untuk suatu variabel. Untuk menghapus nama matriks pada baris dan kolomnya, masing-masing digunakan *syntax* **rownames (matriks3) = NULL** dan **colnames (matriks3) = NULL**.

Ada beberapa operator yang sering digunakan pada operasi vektor dan matriks dalam pemrograman matematis yang mana diringkas dalam Tabel 2.1.

Tabel 2.1. Operator yang digunakan dalam operasi vektor dan matriks.

Operator	Keterangan
*	Perkalian elemen pada vektor atau matriks
%*%	Perkalian matriks
%o%	Outer proses
t	Transpose matriks
crossprod	Crossproduct matriks yaitu t(y)%*%y
solve	Invers matriks

Misalkan diberikan vektor **a** yang bisa digunakan untuk menampilkan penerapan dari perkalian elemen dan crossproduct. Pada crossproduct, vektor **a** dengan 5 elemen akan secara automatis menjadi matriks kolom dengan orde

5x1, sehingga bisa dilakukan operasi **t(a)%*%a** yang jelas akan menghasilkan matriks dengan orde 1x1 karena **t(a)** memiliki orde 1x5 yang mana dikalikan matriks dengan **a** yang memiliki orde 5x1, sehingga akan menghasilkan matriks dengan orde 1x1 dengan nilai **55**.

```
> a = c(1,2,3,4,5)
> a*a
[1]  1  4  9 16 25
> crossprod(a)
     [,1]
[1,]   55
```

Misalkan diberikan matriks **b**, kemudian ditunjukkan penerapan dari perkalian elemen, perkalian matriks, crossproduct, transpose, dan invers. Khusus untuk operator **solve**, matriks harus simetris atau memiliki orde yang sama seperti matriks **b** dengan orde 2x2.

```
> b = matrix(1:4,2)
> b
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> b*b
     [,1] [,2]
[1,]    1    9
[2,]    4   16
> b%*%b
     [,1] [,2]
[1,]    7   15
[2,]   10   22
> crossprod(b)
     [,1] [,2]
[1,]    5   11
[2,]   11   25
> t(b)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
> solve(b)
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
```

Diberikan suatu operator yang bisa digunakan untuk menggabungkan baris dan kolom pada vektor atau matriks.

operator ini dikenali masing-masing dengan fungsi **rbind** dan **cbind**.

```
> a = matrix(c(3,4,5,6,7,8),2,3)
> a
     [,1] [,2] [,3]
[1,]    3    5    7
[2,]    4    6    8
> a1 = cbind(a,c(1,2))
> a1
     [,1] [,2] [,3] [,4]
[1,]    3    5    7    1
[2,]    4    6    8    2
> a2 = cbind(c(1,2),a)
> a2
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
```

Misalkan diberikan matriks **a**, maka diperoleh matriks **a1**, **a2**, **a3**, dan **a4** dari operator penggabungan. Variabel **a1** menyimpan matriks gabungan kolom dari matriks **a** dengan vektor yang elemennya berisi bilangan 1 dan 2. Disisi lain, variabel **a2** menyimpan matriks gabungan kolom dari vektor yang elemennya berisi bilangan 1 dan 2 dengan matriks **a**.

```
> a3 = rbind(a,c(1,2,3))
> a3
     [,1] [,2] [,3]
[1,]    3    5    7
[2,]    4    6    8
[3,]    1    2    3
> a4 = rbind(c(1,2,3),a)
> a4
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    3    5    7
[3,]    4    6    8
```

Variabel **a3** menyimpan matriks gabungan baris dari matriks **a** dengan vektor yang elemennya berisi bilangan 1, 2, dan 3. Di hasil yang lain, variabel **a4** menyimpan matriks gabungan baris dari vektor yang elemennya berisi bilangan 1, 2, dan 3 dengan matriks **a**.

Berikut akan diberikan contoh penerapan identifikasi dan penentuan suatu variabel sebagai vektor atau matriks dengan penerapan fungsi **is.vector**, **as.vector**, **is.matrix**, dan **as.matrix**.

```
> a = 1:5
> a
[1] 1 2 3 4 5
> is.vector(a)
[1] TRUE
> a = as.matrix(a)
> is.matrix(a)
[1] TRUE
> a = crossprod(a)
> a
     [,1]
[1,]   55
> is.vector(a)
[1] FALSE
> a = as.vector(a)
> a
[1] 55
> is.vector(a)
[1] TRUE
```

Syntax **is.vector(a)** digunakan untuk menanyakan apakah variabel **a** merupakan vektor atau bukan kemudian mengembalikan nilai TRUE atau FALSE. *Syntax* **as.vector(a)** digunakan untuk menyatakan atau menentukan bahwa variabel **a** sebagai vektor. Disisi lain, *syntax* **is.matrix(a)** digunakan untuk menanyakan apakah variabel **a** merupakan matriks atau bukan kemudian mengembalikan nilai TRUE atau FALSE. *Syntax* **as.matrix(a)** digunakan untuk menyatakan atau menentukan bahwa variabel **a** sebagai matriks.

3. Penggunaan Data Frame dan Factor.

   Data frame merupakan objek data yang memiliki bentuk serupa dengan Matriks yang mempunyai baris dan kolom, tetapi matriks hanya memuat beberapa data dengan mode data yang sama sedangkan data frame bisa berbeda

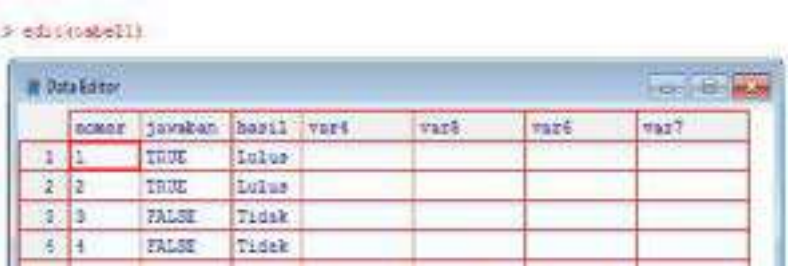
(Suhartono, 2008). Misalkan diberikan data frame dengan kolom pertama adalah vektor numerik, kolom kedua adalah vektor karakter, dan kolom ketiga adalah vektor logika.

```
> nomer = c(1:4)
> jawaban = c(T,T,F,F)
> hasil = c("Lulus","Lulus","Tidak","Tidak")
> tabel1 = data.frame(nomer,jawaban,hasil)
> tabel1
  nomer jawaban hasil
1     1    TRUE Lulus
2     2    TRUE Lulus
3     3   FALSE Tidak
4     4   FALSE Tidak
```

Sebagai pengganti matriks, *programmer* dapat menggunakan data frame yang setiap komponennya adalah suatu vektor yang berhubungan dengan sebuah kolom dalam suatu matriks (Matloff, 2011). Dengan demikian, *indexing* data frame serupa dengan matriks. Untuk melihat struktur dari data frame, digunakan fungsi **str** sebagai berikut.

```
> str(tabel1)
'data.frame':   4 obs. of  3 variables:
 $ nomer  : int  1 2 3 4
 $ jawaban: logi  TRUE TRUE FALSE FALSE
 $ hasil  : chr  "Lulus" "Lulus" "Tidak" "Tidak"
```

Untuk memperbaiki nilai data frame yang terbentuk, *programmer* bisa menggunakan proses pengeditan data dengan fungsi **edit**.



Indexing data frame memiliki kesamaan dengan indexing pada matriks yang mana penerapannya diberikan, sebagai berikut:

a. Matriks

Pada contoh matriks, syntax matriks[2,2] mengembalikan nilai matriks pada elemen baris ke-2 dan kolom ke-2 yaitu 5.

```
> matriks1 = matrix(1:9,3,3)
> matriks1
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> matriks1[2,2]
[1] 5
```

b. Data frame

Pada contoh data frame, syntax tabel2[2,2] mengembalikan nilai data frame pada elemen baris ke-2 dan kolom ke-2 yaitu "Budi". Di samping itu, data frame juga bisa menggunakan head¬¬er-nya untuk indexing seperti pada syntax tabel2["nama"].

```
> nomer = 1:4
> nama = c("Adi","Budi","Cika","Dony")
> nilai = 7:10
> tabel2 = data.frame(nomer,nama,nilai)
> tabel2
  nomer nama nilai
1     1  Adi     7
2     2 Budi     8
3     3 Cika     9
4     4 Dony    10
> tabel2[2,2]
[1] "Budi"
> tabel2["nama"]
  nama
1  Adi
2 Budi
3 Cika
4 Dony
```

Dalam data frame, bisa dilakukan proses *filtering* pada tabel yang terbentuk. Berikut diberikan contoh penerapannya sebagai berikut.

```
> tabel2$nilai > 8
[1] FALSE FALSE  TRUE  TRUE
> tabel2$nama[tabel2$nilai > 8]
[1] "Cika" "Dony"
```

Misalkan pada *syntax* **tabel2$nilai > 8** menampilkan nilai logika apakah **nilai** pada data frame **tabel2** bernilai diatas 8. *Syntax* ini menghasilkan nilai logika yang menjadi rujukan dalam menentukan siapa saja yang memiliki **nilai** diatas 8 dalam data frame **tabel2** yang mana diberikan dalam *syntax* **tabel2$nama[tabel2$nilai > 8]**. Hasilnya siapa saja yang memiliki nilai diatas 8 dalam data frame **tabel2**, yaitu: **Cika** dan **Dony**. Selanjutya akan diberikan pembahasan tentang Factor.

Factor merupakan objek data yang digunakan untuk tujuan mengkategorikan data, kemudian menyimpannya di dalam level baik string maupun bilangan bulat (Bell, 2020). Cara menginputkan factor sama saja dengan vektor, kemudian vektor tersebut diformat ke dalam bentuk factor.

```
> Nilai = c("A","B","A","C","C","A")
> Nilai
[1] "A" "B" "A" "C" "C" "A"
> is.factor(Nilai)
[1] FALSE
> Nilai = as.factor(Nilai)
> Nilai
[1] A B A C C A
Levels: A B C
> is.factor(Nilai)
[1] TRUE
```

Awalnya variabel **Nilai** merupakan suatu vektor karakter, kemudian dirubah menjadi suatu factor dengan fungsi **as.factor**. *Indexing* factor memiliki kemiripan dengan *indexing* pada vektor.

```
> Nilai
[1] A B A C C A
Levels: A B C
> Nilai[2]
[1] B
Levels: A B C
> Nilai[2:4]
[1] B A C
Levels: A B C
> Nilai[c(2,4)]
[1] B C
Levels: A B C
> Nilai[Nilai=="A"]
[1] A A A
Levels: A B C
> Nilai[-2]
[1] A A C C A
Levels: A B C
> Nilai[-(2:4)]
[1] A C A
Levels: A B C
> Nilai[-c(2,4)]
[1] A A C A
Levels: A B C
```

Perbedaan antara factor dan vektor yang paling jelas adalah terletak pada factor yang memiliki level yang sifatnya kategorik. Untuk *generate* factor, bisa digunakan fungsi **gl** ('*generate levels*') yang berguna ketika *programmer* ingin membuat vektor panjang dengan level factor (Crawley, 2013).

```
> gl(4,3)
 [1] 1 1 1 2 2 2 3 3 3 4 4 4
Levels: 1 2 3 4
```

Syntax **gl(4,3)** men-*generate* 4 level dengan setiap levelnya ada 3 bilangan. Jika menginginkan bentuk *generate* seperti ini ditampilkan 2 kali, maka *programmer* bisa menentukan banyak pengamatannya menjadi **24**.

```
> gl(4,3,24)
 [1] 1 1 1 2 2 2 3 3 3 4 4 4 1 1 1 2 2 2 3 3 3 4 4 4
Levels: 1 2 3 4
```

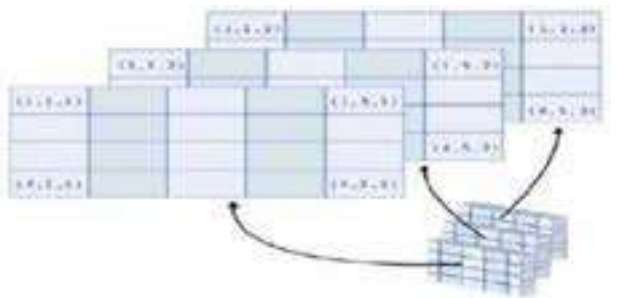
4. Penggunaan Array dan List.

Matriks terdiri dari dua dimensi, sedangkan Array dapat memiliki dimensi berapapun yang mana atribut array

ditentukan oleh "**dim**" yang menentukan banyaknya dimensi array yang dibuat (Bell, 2020). Misalkan diberikan contoh array tiga dimensi dimana urutan dimensinya (baris, kolom, sub-tabel) menjadi (2x3x4) yang berisi angka 1 sampai 24 secara berurutan (Crawley, 2013):

```
> data = array(1:24,dim=c(2,3,4))
> data
, , 1

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

, , 2

     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12

, , 3

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

, , 4

     [,1] [,2] [,3]
[1,]   19   21   23
[2,]   20   22   24
```

Diperoleh array tiga dimensi yang menggambarkan 4 sub-tabel yang masing-masing dengan 2 baris dan 3 kolom. Array tiga dimensi bisa juga digambarkan sebagai tabel muldimensional yaitu tiga dimensi (Micheaux et al., 2013), sebagai berikut.



Gambar 2.1
Ilustrasi Dari Tabel Multidimensional Untuk Array Tiga Dimensi.

Disamping itu, array juga bisa diberikan nama untuk tingkat faktor di masing-masing dari tiga dimensi dengan fungsi **dimnames.**

```
> dimnames(data)[[1]] = c("Laki-laki","Perempuan")
> dimnames(data)[[2]] = c("<30","30-60",">60")
> dimnames(data)[[3]] = c("Grup A","Grup B","Grup C","Grup D")
> data
, , Grup A

          <30 30-60 >60
Laki-laki   1     3   5
Perempuan   2     4   6

, , Grup B

          <30 30-60 >60
Laki-laki   7     9  11
Perempuan   8    10  12

, , Grup C

          <30 30-60 >60
Laki-laki  13    15  17
Perempuan  14    16  18

, , Grup D

          <30 30-60 >60
Laki-laki  19    21  23
Perempuan  20    22  24
```

Misalkan array tersebut dirubah dengan empat grup (A-D) menjadi kolom-kolom pada masing-masing sub-tabel, kemudian sub-tabel yang terpisah mewakili dua jenis kelamin. Kondisi ini bisa diterapkan dengan fungsi **aperma**, dimana urutan dimensinya (baris, kolom, sub-tabel) yang diawal (2,3,4) dirubah menjadi (2,3,1) dengan mudah.

```
> new.data = aperm(data,c(2,3,1))
> new.data
, , Laki-laki

      Grup A Grup B Grup C Grup D
<30         1      7     13     19
30-60       3      9     15     21
>60         5     11     17     23

, , Perempuan

      Grup A Grup B Grup C Grup D
<30         2      8     14     20
30-60       4     10     16     22
>60         6     12     18     24
```

Setelah bahasan factor, selanjutnya diberikan bahasan tentang List. List merupakan objek data yang sangat penting dalam R karena list dapat memuat empat objek data dengan mode data yang berbeda, seperti: vektor numerik, vektor logika, vektor karakter dan vektor kompleks (Crawley, 2013). Selain itu, list dapat menampung elemen-elemen berupa objek data yang berbeda di dalamnya dan tidak berhubungan, seperti: vektor, matriks, data frame, factor, array, fungsi, dan list yang lain (Suhartono, 2008). Contoh penerapan list secara sederhana diberikan sebagai berikut.

```
> nomer = c(1:3)
> jawaban = c(T,F,T,T)
> tabel1 = data.frame(nama=c("Budi","Cika","Dony"),nilai=c(8:10))
> daftar1 = list(nomer,jawaban,tabel1)
> daftar1
[[1]]
[1] 1 2 3

[[2]]
[1]  TRUE FALSE  TRUE  TRUE

[[3]]
  nama nilai
1 Budi     8
2 Cika     9
3 Dony    10
```

Misalkan diberikan vektor numerik **nomer**, vektor logika **jawaban**, data frame **tabel1** yang ditampung ke dalam

list yang bernama **daftar1**. Untuk merubah indeks list dari angka menjadi karakter, diberikan contoh *syntax* berikut.

```
> daftar2 = list(nomer=nomer,jawaban=jawaban,tabel1=tabel1)
> daftar2
$nomer
[1] 1 2 3

$jawaban
[1]  TRUE FALSE  TRUE  TRUE

$tabel1
  nama nilai
1 Budi     8
2 Cika     9
3 Dony    10
```

Indexing list atau ekstraksi sebagian data pada list dapat pula dilakukan dengan berbagai cara.

```
> daftar1[[3]]
  nama nilai
1 Budi     8
2 Cika     9
3 Dony    10
> daftar1[[3]][[1]]
[1] "Budi" "Cika" "Dony"
> daftar2$tabel1
  nama nilai
1 Budi     8
2 Cika     9
3 Dony    10
> daftar2$tabel1$nama
[1] "Budi" "Cika" "Dony"
```

Misalkan diberikan *syntax* **daftar1[[3]]** akan mengembalikan nilai dari list ketiga pada list **daftar1**. Selanjutnya hasil yang lain, *syntax* **daftar1[[3]][[1]]** akan mengembalikan nilai vektor **nama** dari data frame **tabel1** pada list **daftar1**. Pada list **daftar2**, dapat digunakan operator dollar (**$**) untuk ekstrasi list. Contohnya pada *syntax* **daftar2$tabel1** akan mengembalikan nilai data frame **tabel1** pada list **daftar2**. Hasil yang lain, *syntax* **daftar2$tabel1$nama** akan mengembalikan nilai vektor **nama** dari data frame **tabel1** pada list **daftar2**.

D. Rangkuman

Pengenalan tipe data dan mode data dalam software R memberikan banyak pengetahuan tentang tipe data yaitu: Vektor, Matriks, Data Frame, Factor, Array, dan List; serta diberikan pengetahuan mode data yang bersifat numerik, kompleks, logika, dan karakter. Penulisan nama pada objek data haruslah valid sesuai ketentuan yang ada di software R. Dalam ilmu pemrograman dasar, suatu variabel digunakan untuk penyimpanan data dan nilainya dapat berubah berdasarkan kebutuhan programmer, sedangkan suatu konstanta memiliki nilai yang tetap atau tidak bisa berubah. Variabel bisa digambarkan dalam suatu objek data yang mana variabel bisa berupa Vektor, Matriks, Data Frame, Factor, Array, dan List. Penggunaan vektor dan matriks sangat mudah dalam hal pembuatannya, indexing, dan operasi aritmatikanya. Vektor adalah array dengan satu dimensi, sedangkan matriks adalah array dengan dua dimensi. Dalam buku ini, dikenalkan juga Array dalam bentuk yang umum dengan dimensi berapapun. Selain itum diberikan juga objek data yaitu Data Frame yang memiliki bentuk seperti tabel dan indexing-nya mirip seperti indexing pada matriks. Diberikan juga bahasan Factor yang hampir sama dengan vektor, tetapi factor lebih digunakan pada pengkategorian dan memiliki suatu level tertentu. Objek data yang paling general adalah List karena dapat memuat atau menampung semua tipe objek data dan fungsi.

E. Soal Latihan

Buatlah syntax di R Console sesuai dengan algoritma program berikut:

1. Buatlah Vektor A yang berisi bilangan bulat ganjil 1 sampai 9 !
2. Buatlah Matrik B yang berukuran 3x3 yang mana baris pertama bernilai 1, 2, 3 ; baris kedua bernilai 4, 5, 6 ; dan baris ketiga bernilai 7, 8, 9 !

3. Buatlah Data Frame C yang mana kolom pertama berisi nomor 1 sampai 4 ; kolom kedua berisi nama yaitu “Nisa”, “Laelatul”, “Teguh”, “Sri” ; dan kolom ketiga berisi nilai numerik 85, 80, 70, 75 !
4. Buatlah Factor D yang mana berisi nilai kategori yaitu “Benar”, “Salah”, “Benar”,“Salah”, ’’Benar”, “Benar”, ”Salah” !
5. Buatlah Array E dengan tiga dimensi yang elemennya bernilai 1 sampai 12 dimana dimensinya (2x3x2) !
6. Buatlah List F yang berisi Vektor A, Matriks B, Data Frame C, Factor D, dan Array E !

F. Sumber Referensi

Bell, D. (2020). *R Programming: A Step-by-Step Guide for Absolute Beginners (2nd Ed).* KDP Amazon Publishing.

Crawley, M. J. (2013). *The R Book (2nd Ed).* John Wiley & Sons.

Matloff, N. (2011). *The Art of R Programming: A Tour of Statistical Software Design*. No Starch Press.

Micheaux, P. L. de, Drouilhet, E., & Liquet, B. (2013). *The R Software: Fundamentals of Programming and Statistical Analysis*. Spinger.

Suhartono. (2008). *Analisis Data Statistik Dengan R*. Lab. Statistik Komputasi ITS.

*****

BAB 3

MANAJEMEN DATA DAN PENULISAN FUNGSI

A. Deskripsi singkat

Materi pada bab ini merupakan pendalaman materi terkait proses manajemen data di dalam software R dengan memberikan pemaparan materi tentang pernyataan import dan export, pernyataan insert dan browse data frame, dan pengenalan struktur penulisan fungsi di software R baik penerapannya di R Console maupun di R Editor. Bahasan ini diberikan dengan penjelasan singkat dan contoh penerapan bagaimana membuat syntax yang benar dalam pemrograman khususnya penggunaan perintah import, export, insert, dan browse di software R, serta pemahaman struktur fungsi yang ada. Pada bab ini diharapkan pembaca dapat memahami dan menerapkan bagaimana memanfaatkan penggunaan dari manajemen data dan penulisan fungsi. Materi pada bab ini berguna untuk mengetahui bagaimana manajemen dataset dan membuat fungsi sesuai keinginan pengguna di lingkungan pemrograman bahasa R.

B. Tujuan Pembelajaran

Pada bab ini, diberikan tujuan pembelajaran agar pembaca mampu menjelaskan dan menerapkan bagaimana:

1. pernyataan import dan export.
2. pernyataan insert dan browse data frame.
3. struktur penuisan fungsi di software R.

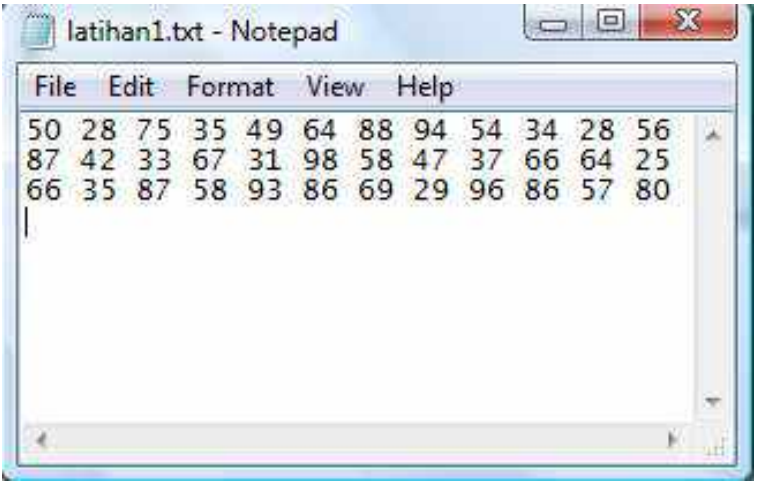
C. Penyajian Materi

1. Pernyataan Import dan Export

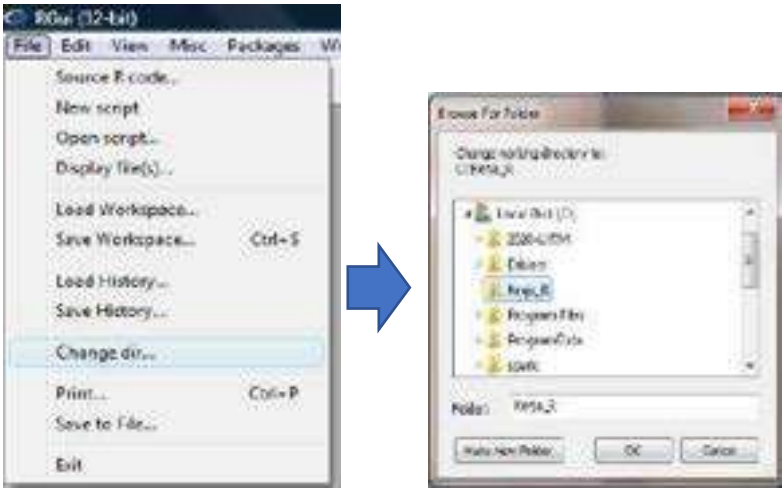
   Bahasan mengenai manajemen data dalam software R berkaitan dengan manajemen directori kerja. Software R akan

menyimpan semua file kerja di dalam direktori ini yang mana secara default disimpan di lokasi C:\Program Files\R\R-4.1.3 untuk software R versi 4.1.3. Direktori kerja bisa dirubah sesuai keinginan pengguna. Misalkan pengguna berkeinginan menempatkan direktori kerja di folder dan lokasi tertentu, maka diberikan conto penerapan dengan tahapan berikut:

a. membuat folder baru dengan nama Kerja_R di Local Disk (C:).
b. membuat file ASCII dengan *software* Notepad, kemudian diisi dengan dataset yang diinginkan dan disimpan di folder **Kerja_R** dengan nama **latihan1.txt**.



c. memindahkan direktori kerja R dari lokasi default ke lokasi folder Kerja_R dengan mengklik toolbar atas yaitu: File -> Change dir, kemudian pilih lokasi dari folder Kerja_R.



Perintah import dalam software R dilakukan dengan fungsi scan, read.tabel, dan read.csv. Berikut diberikan contoh penerapan fungsi scan dengan direktori kerja dan tanpa direktori kerja:

a. Fungsi scan dengan direktori kerja.

Penggunaan fungsi scan tinggal memilih dataset jika direktori sudah ditentukan di folder Kerja_R. Dataset yang masuk ke dalam software R dengan fungsi scan akan dikenali sebagai suatu vektor pada variabel datascan1, sehingga pada penerapan ini bisa juga diubah ke dalam suatu matriks yaitu seperti pada variabel matriks1.

```
> scan("latihan1.txt")
Read 36 items
 [1] 50 28 75 35 49 64 88 94 54 34 28 56 87 42 33 67 31 98
[19] 58 47 37 66 64 25 66 35 87 58 93 86 69 29 96 86 57 80
> datascan1 = scan("latihan1.txt")
Read 36 items
> datascan1
 [1] 50 28 75 35 49 64 88 94 54 34 28 56 87 42 33 67 31 98
[19] 58 47 37 66 64 25 66 35 87 58 93 86 69 29 96 86 57 80
> matriks1 = matrix(datascan1,6,6,T)
> matriks1
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   50   28   75   35   49   64
[2,]   88   94   54   34   28   56
[3,]   87   42   33   67   31   98
[4,]   58   47   37   66   64   25
[5,]   66   35   87   58   93   86
[6,]   69   29   96   86   57   80
```

b. Fungsi scan tanpa direktori kerja.
Penggunaan fungsi scan tanpa direktori kerja bisa diterapkan oleh pengguna dengan menambahkan URL terkait dari lokasi dataset tersebut, yaitu misalkan "c:\\Kerja_R\\latihan1.txt". Sama halnya, semua dataset yang masuk ke dalam software R dengan fungsi scan akan dikenali sebagai suatu vektor seperti pada variabel datascan2.

```
> datascan2 = scan("c:\\Kerja_R\\latihan1.txt")
Read 36 items
> datascan2
 [1] 50 28 75 35 49 64 88 94 54 34 28 56 87 42 33 67 31 98
[19] 58 47 37 66 64 25 66 35 87 58 93 86 69 29 96 86 57 80
> matriks2 = matrix(datascan2,6,6,T)
> matriks2
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   50   28   75   35   49   64
[2,]   88   94   54   34   28   56
[3,]   87   42   33   67   31   98
[4,]   58   47   37   66   64   25
[5,]   66   35   87   58   93   86
[6,]   69   29   96   86   57   80
```

Perintah import juga bisa dilakukan dengan read.table dan read.csv. Berikut diberikan contoh penerapannya secara step-by-step:

a. Membuat *dataset* pertama untuk penerapan fungsi **read.csv** dengan *software* Microsoft Excel, kemudian disimpan di folder **Kerja_R** dengan format *commad delimited* (*.csv), yaitu: **latihan2.csv .**

	A	B	C	D	E
1	**Responden**	**Matematika**	**B.Indonesia**	**B.Inggris**	**IPK**
2	Adi	8	9.1	8.4	3.35
3	Budi	7.6	8.8	8.5	3.02
4	Dany	6.9	8.1	7.2	2.9
5	Eka	8.9	9.2	9	3.42
6	Fery	9.5	9.6	9.5	3.75
7	Nuri	7.3	8.7	7.9	3.26
8	Rury	6.5	7.5	8.2	2.76
9					

b. Membuat *dataset* kedua untuk penerapan fungsi **read.table** dengan *copy-paste* dari dataset pertama ke file

Notepad kemudian disimpan di folder **Kerja_R** dengan nama **latihan2.txt .**

c. Menerapkan fungsi **read.csv** tanpa direktori kerja.
Semua d*ataset* yang masuk ke dalam *software* R dengan fungsi **read.csv** akan dikenali sebagai suatu data frame seperti pada variabel **datatable1.**

```
> datatable1 = read.csv("c:\\Kerja_R\\latihan2.csv", header=TRUE)
> datatable1
  Responden Matematika B.Indonesia B.Inggris  IPK
1      Adi        8.0         9.1       8.4 3.35
2     Budi        7.6         8.8       8.5 3.02
3     Dany        6.9         8.1       7.2 2.90
4      Eka        8.9         9.2       9.0 3.42
5     Fery        9.5         9.6       9.5 3.75
6     Nuri        7.3         8.7       7.9 3.26
7     Rury        6.5         7.5       8.2 2.76
```

d. Menerapkan fungsi **read.table** tanpa direktori kerja.
Sama halnya, semua d*ataset* yang masuk ke dalam *software* R dengan fungsi **read.table** juga akan dikenali sebagai suatu data frame seperti pada variabel **datatable2.**

```
> datatable2 = read.table("c:\\Kerja_R\\latihan2.txt", header=TRUE)
> datatable2
  Responden Matematika B.Indonesia B.Inggris  IPK
1      Adi        8.0         9.1       8.4 3.35
2     Budi        7.6         8.8       8.5 3.02
3     Dany        6.9         8.1       7.2 2.90
4      Eka        8.9         9.2       9.0 3.42
5     Fery        9.5         9.6       9.5 3.75
6     Nuri        7.3         8.7       7.9 3.26
7     Rury        6.5         7.5       8.2 2.76
```

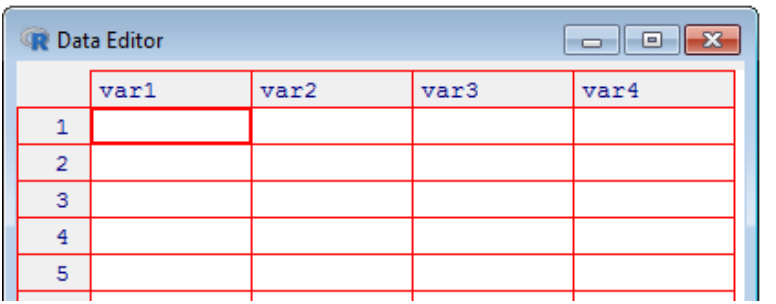
2. Pernyataan Insert dan Browse Data Frame.

Untuk memasukan dataset secara langsung ke dalam software R, digunakan pernyatan insert data frame yang akan memunculkan kotak Data Editor (Suhartono, 2008). Berikut diberikan contoh penerapan dari pernyataan insert secara bertahap:

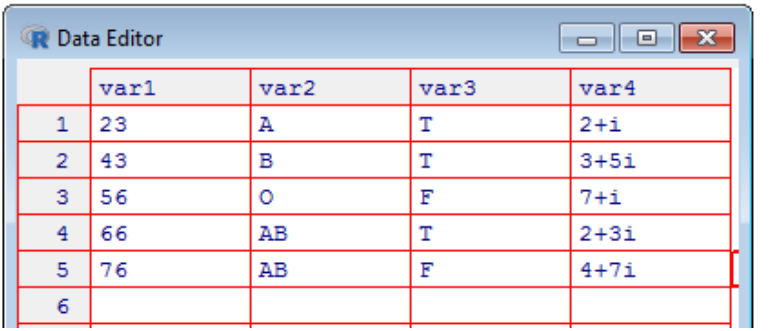
a. Memunculkan Data Editor.

Syntax **dataset1 = edit(as.data.frame(NULL))** akan memunculkan *Data Editor* yang memuat data frame kosong yang bisa diisi oleh pengguna.

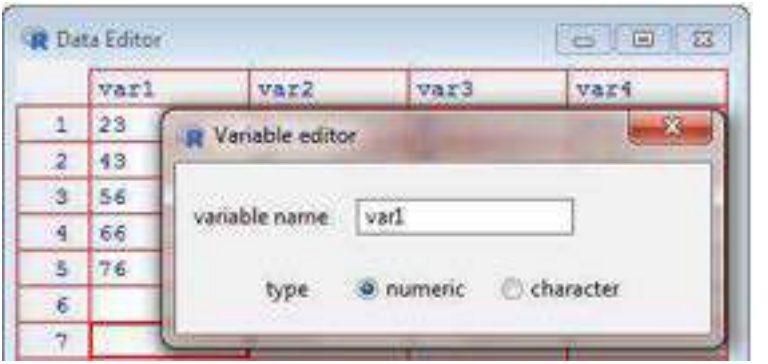
```
> dataset1 = edit(as.data.frame(NULL))
```



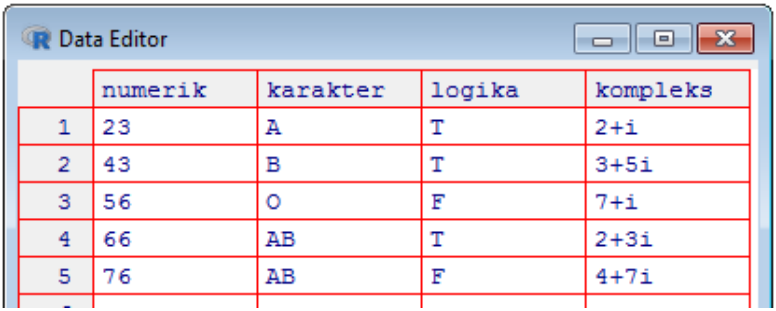
b. Menginputkan data secara manual.



c. Menentukan nama *header* data frame dengan cara mengklik *header* tersebut (misal **var1**) dan *rename* menjadi **numerik**.



Lakukan perubahan nama yang lain dengan cara yang sama untuk setiap kolom tabel.



	numerik	karakter	logika	kompleks
1	23	A	T	2+i
2	43	B	T	3+5i
3	56	O	F	7+i
4	66	AB	T	2+3i
5	76	AB	F	4+7i

Kemudian klik **X** untuk menutup *Data Editor*.

d. Menampilkan data frame baru.

```
> dataset1
  numerik karakter logika kompleks
1      23        A      T      2+i
2      43        B      T    3+5i
3      56        O      F     7+i
4      66       AB      T    2+3i
5      76       AB      F    4+7i
```

Untuk memasukan *dataset* dengan cara menelusuri data, digunakan pernyataaan *browse* data frame yang contoh penerapannya, sebagai berikut:

a. Mengetikkan pernyatan *browse* data frame dan memilih file Excel yang berekstensi *.csv.

```
> dataset2 = read.csv(file.choose(), header=TRUE)
```

Misalkan dipilih file **latihan2.csv** .

b. Menampilkan data frame baru.

```
> dataset2
  Responden Matematika B.Indonesia B.Inggris  IPK
1       Adi        8.0         9.1       8.4 3.35
2      Budi        7.6         8.8       8.5 3.02
3      Dany        6.9         8.1       7.2 2.90
4       Eka        8.9         9.2       9.0 3.42
5      Fery        9.5         9.6       9.5 3.75
6      Nuri        7.3         8.7       7.9 3.26
7      Rury        6.5         7.5       8.2 2.76
```

Penerapan pernyataan *browse* data frame ini merupakan cara yang paling praktis untuk memasukkan *dataset* ke dalam *software* R tanpa menggunakan direktori data.

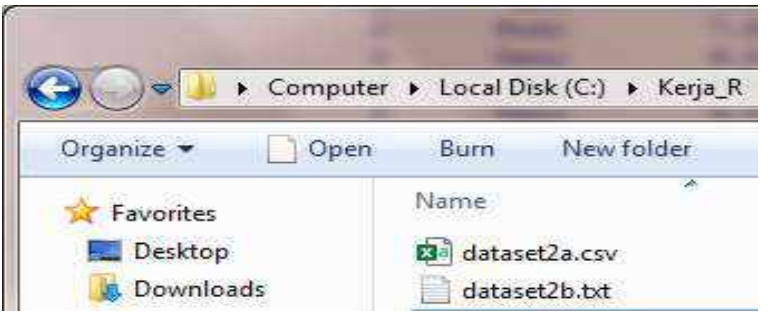
Proses *export* output dalam *software* R dapat diberikan dalam output file yang berekstensi *csv dan *txt dengan bantuan fungsi seperti **write.csv** dan **write.txt** (Team, 2021).

```
> write.csv(dataset2,file="c:\\Kerja_R\\dataset2a.csv")
```

Sebagai contoh, diberikan *export* **dataset2** dalam output file yang berekstensi *csv dengan URL yaitu **"c:\\Kerja_R\\dataset2a.csv"** .

```
> write.table(dataset2,file="c:\\Kerja_R\\dataset2b.txt",sep="\t",row.names=FALSE)
```

Contoh yang lain, diberikan *export* **dataset2** dalam output file yang berekstensi *txt dengan URL yaitu **"c:\\Kerja_R\\dataset2b.txt"** dengan separator **"\t"** dan tanpa nama baris.



Demikian hingga, dua *export* data tersebut dapat ditemukan di folder **Kerja_R**.

3. Struktur Penulisan Fungsi di *Software* R.

Fungsi sangat berguna untuk membantu *programmer* agar tidak menulis ulang semua *syntax* di dalam suatu program setiap kali *programmer* ingin menggunakan *syntax*-nya lagi (Sarmento & Costa, 2017). Fungsi didefinisikan dengan domain input dengan beberapa prosedur internal (proses) yang mana akan menghasilkan suatu output yang sesuai dengan keinginan *programmer*. Secara umum, struktur penulisan fungsi di dalam *software* R diberikan sebagai berikut.

```
nama_fungsi = function(input_fungsi)
{
//argumen fungsi yang diinginkan berisi proses dan output
}
```

Untuk memahami kinerja fungsi dalam bahsa R diberikan contoh penerapannya. Misalkan diketahui jari-jari suatu lingkaran adalah 10 cm, maka *programmer* dapat mengetahui luas dan keliling lingkaran tersebut dengan cara membuatkan fungsi **Lingkaran** dengan input jari-jari lingkaran (**r**) yang digunakan untuk melakukan beberapa proses seperti menghitung **Luas** dan **Keliling** lingkaran, sehingga dihasilkan fungsi yang mengembalikan nilai output berupa **Luas** dan **Keliling** lingkaran. Penerapan fungsi ini dapat diberikan contoh penulisannya dalam *R Console* sebagai berikut.

```
> Lingkaran = function(r){
+ Luas = pi*r*r
+ Keliling = 2*pi*r
+ Hasil = list(Luas=Luas,Keliling=Keliling)
+ Hasil
+ }
>
```

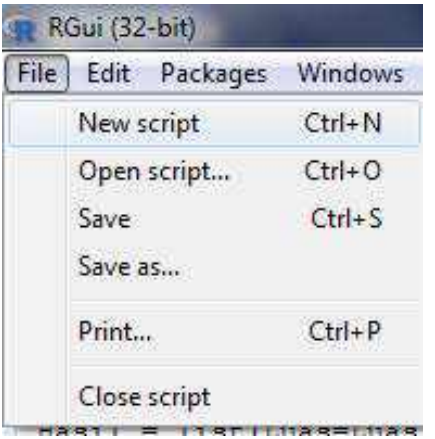
Fungsi **Lingkaran** didefinisikan dengan domain input **r**, kemudian input **r** tersebut dipakai untuk menghitung luas dan keliling lingkaran masing-masing dengan rumusan matematis yaitu **Luas** = $\boldsymbol{\pi r^2}$ dan **Keliling** = $\boldsymbol{2 \pi r}$. Simbol $\boldsymbol{\pi}$ mewakili

konstanta **pi** yang telah ditentukan di dalam *software* R dengan nilai 3.141593. Proses berikutnya adalah memasukkan variabel **Luas** dan **Keliling** ke dalam suatu list yang bernama **Hasil**, sehingga fungsi dapat mengembalikan nilai output **Hasil** yang berisi luas dan keliling lingkaran.

```
> Lingkaran(10)
$Luas
[1] 314.1593

$Keliling
[1] 62.83185
```

Demikian pengguna tinggal memanggil fungsi **Lingkaran** tersebut. Misalkan pengguna mengetikkan fungsi **Lingkaran** dengan domain input **r** sebesar 10, maka akan dihasilkan list Hasil yang berisi **Luas** dan **Keliling** lingkaran masing-masing sebesar **341.1593** dan **62.83185**. Agar *syntax* fungsi **Lingkaran** dapat disimpan dan dijalankan kembali, *programmer* bisa menuliskan *syntax* fungsi tersebut di *R Editor*. Tampilan *R Console* dan *R Editor* dapat dilihat di **Bab 1**. Penulisan fungsi di *R Editor* diberikan secara *step-by-step* sebagai berikut:

a. Membuat skrip baru dengan mengklik toolbar atas yaitu *File -> New script*.

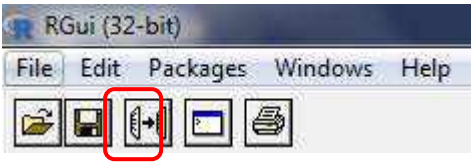


b. Menuliskan *syntax* fungsi **Lingkaran** di *R Editor*.

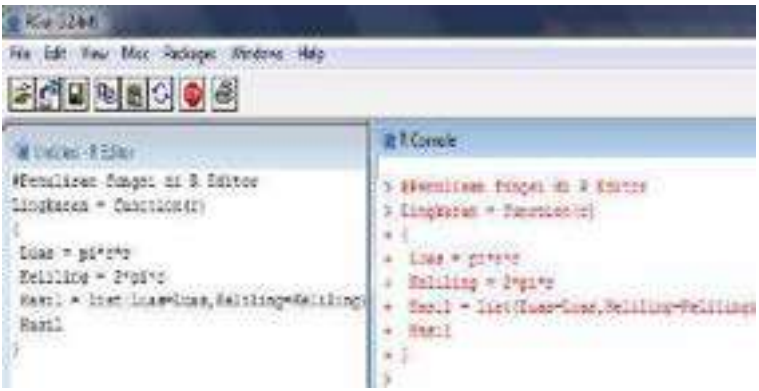
```
#Penulisan fungsi di R Editor
Lingkaran = function(r)
{
 Luas = pi*r*r
 Keliling = 2*pi*r
 Hasil = list(Luas=Luas,Keliling=Keliling)
 Hasil
}
```

c. Menyimpan dan mengeksekusi skrip R dengan cara mengklik tombol **Save** untuk menyimpan skrip R,

kemudian blok semua *syntax* fungsi di *R Editor* dengan tombol *keyboard* (**Ctrl** + **A**), selanjutnya mengklik tombol **Run** untuk mengeksekusi skrip R.



Demikian hasil eksekusi tersebut dihasilkan dalam visualisasi berikut.



d. Memanggil fungsi **Lingkaran** di *R Console*.

```
> Lingkaran(10)
$Luas
[1] 314.1593

$Keliling
[1] 62.83185
```

Di dalam *software* R terdapat fungsi-fungsi yang sudah *built-in* untuk matematika yang diberikan, seperti: **log**, **exp**, **sqrt**, **abs**, **factorial**, **round**, **cos**, **sin**, **cos**, **tan**, a**sin**, a**cos**, a**tan**, dan lainnya (Crawley, 2013).

```
> Z0 = pi
> Z0
[1] 3.141593
> Z1 = log(Z0)
> Z1
[1] 1.14473
> Z2 = exp(Z1)
> Z2
[1] 3.141593
```

Misalkan diberikan variabel **Z0** = **pi** = **3.141593**, maka nilai logaritma natural variabel **Z1** = **log(Z0)** = **ln(pi)** = **1.14473**. Demikian hingga nilai invers dari logaritma natural atau nilai eksponen, diperoleh variabel **Z2** = **exp(Z1)** = **3.141593.**

```
> Z3 = log(Z2,2)
> Z3
[1] 1.651496
> Z4 = sqrt(Z3)
> Z4
[1] 1.285105
> Z5 = abs(Z4-2*Z4)
> Z5
[1] 1.285105
```

Diberikan contoh misalkan nilai logaritma dengan basis **2** variabel **Z3** = ^{2}**log(Z2)** = **1.651496**. Salah satu fungsi lain yang sering digunakan adalah **sqrt** dan **abs** yang masing-masing merupakan fungsi yang mengembalikan nilai akar kuadrat dan nilai mutlak.

```
> Z6 = round(Z5,3)
> Z6
[1] 1.285
> Z7 = round(Z6,0)
> Z7
[1] 1
> Z8 = factorial(3*Z7)
> Z8
[1] 6
```

Fungsi yang digunakan untuk meringkas bilangan dengan berapa angka dibelakan koma adalah **round**. Misalkan **Z6 = round(Z5,3)**, maka diartikan maksudnya adalah meringkas bilangan **Z5** dengan tiga angka dibelakang koma kemudian disimpan di variabel **Z6**. Demikian juga untuk **Z7** digunakan meringkas bilangan **Z6** menjadi bilangan bulat. Fungsi yang sudah *built-in* akhir-akhir ini adalah **factorial**, sehingga pengguna bisa menggunakannya untuk menghitung faktorial misalkan **Z8 = factorial(3*Z7) = 3! = 3 * 2 * 1 = 6**.

```
> Y = 30
> Y1 = sin(Y*(pi/180))
> Y1
[1] 0.5
> Y2 = cos(Y*(pi/180))
> Y2
[1] 0.8660254
> Y3 = tan(Y*(pi/180))
> Y3
[1] 0.5773503
```

Untuk masalah Trigonometri, juga sudah ada fungsi yang sudah *built-in*. Misalkan **Y = 30,** maka **Y1 = sin(30^o)** diukur dengan **Y1 = sin(30*(pi/180)),** sehingga diperoleh nilainya sama dengan **0.5** . Hal ini serupa penerapannya untuk fungsi-fungsi trigonometri yang lainnya.

```
> Z1 = asin(Y1)/(pi/180)
> Z1
[1] 30
> Z2 = acos(Y2)/(pi/180)
> Z2
[1] 30
> Z3 = atan(Y3)/(pi/180)
> Z3
[1] 30
```

Untuk nilai invers dari fungsi trigometri, misalkan diberikan **Z1 = arcsin(30^{o})** diukur dengan **Z1 = asin(0.5)/(pi/180),** sehingga diperoleh nilainya sama dengn **30** . Hal ini serupa penerapannya untuk fungsi-fungsi invers trigonometri yang lainnya.

Selain fungsi Matematika, terdapat juga fungsi Statistika yang bisa dimanfaatkan oleh pengguna *software* R, seperti: **length**, **min**, **max**, **mean**, **sd**, **print**, **summary**, **table**, dan lainnya (Sarmento & Costa, 2017).

```
> X = seq(5,50,5)
> X
 [1]  5 10 15 20 25 30 35 40 45 50
> length(X)
[1] 10
> min(X)
[1] 5
> max(X)
[1] 50
> mean(X)
[1] 27.5
> sd(X)
[1] 15.13825
> print(X)
 [1]  5 10 15 20 25 30 35 40 45 50
> summary(X)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.00   16.25   27.50   27.50   38.75   50.00
```

Diberikan vektor numerik **X** yang dapat dihitung banyak datanya dengan fungsi **length**, kemudian nilai minimal, maksimal, rata-rata, dan standar deviasi dari X dapat ditampilkan masing-masing dengan fungsi **min**, **max**, **mean**, dan **sd**. Untuk menampilkan vektor **X** juga dapat digunakan fungsi **print**. Disamping itu, terdapat fungsi yang dapat menampilkan ringkasan tabel deskriptif statistik dari *dataset* vektor **X** yatu fungsi **summary**.

```
> mahasiswa = c("Faqih","Manaf","Zulvi","Septi","Angga","Zulvi","Zulvi","Manaf","Angga")
> kelas = c("A","A","A","B","C","B","C","D","D")
> table(mahasiswa)
mahasiswa
Angga Faqih Manaf Septi Zulvi
    2     1     2     1     3
> table(mahasiswa,kelas)
          kelas
mahasiswa A B C D
    Angga 0 0 1 1
    Faqih 1 0 0 0
    Manaf 1 0 0 1
    Septi 0 1 0 0
    Zulvi 1 1 1 0
```

Untuk dua vektor karakter, bisa ditampilkan ringkasan tabel distribusi frekuensi dan tabel tabulasi silang dengan menggunakan fungsi **table**. Fungsi lain yang sering digunakan adalah fungsi **sort** yang berguna untuk mengurutkan data, fungsi **rank** yang digunakan untuk memberi peringkat data, dan fungsi **order** yang berguna untuk menampilkan indeks pengurutan.

```
> y <- c(8,3,5,7,6,6,8,9,2,3,9,4,10,4,11)
> sorted = sort(y)
> ranked = rank(y)
> ordered = order(y)
> data = data.frame(y,sorted,ranked,ordered)
> data
    y sorted ranked ordered
1   8      2   10.5       9
2   3      3    2.5       2
3   5      3    6.0      10
4   7      4    9.0      12
5   6      4    7.5      14
6   6      5    7.5       3
7   8      6   10.5       5
8   9      6   12.5       6
9   2      7    1.0       4
10  3      8    2.5       1
11  9      8   12.5       7
12  4      9    4.5       8
13 10      9   14.0      11
14  4     10    4.5      13
15 11     11   15.0      15
```

Diberikan contoh *syntax* **order(y)** menampilkan indeks pengurutan dari vektor **y**, sehingga jika ingin menggunakannya untuk pengurutan **y**, maka dapat diterapkan *syntax* **y[order(y)]**.

```
> y <- c(8,3,5,7,6,6,8,9,2,3,9,4,10,4,11)
> y
 [1]  8  3  5  7  6  6  8  9  2  3  9  4 10  4 11
> order(y)
 [1]  9  2 10 12 14  3  5  6  4  1  7  8 11 13 15
> y[order(y)]
 [1]  2  3  3  4  4  5  6  6  7  8  8  9  9 10 11
```

D. Rangkuman

Pernyataan import dan export bisa dilakukan dengan mudah pada file yang berektensi *csv dan *txt. Proses ini dapat dengan mudah diterapkan baik dengan direktori kerja maupun tanpa direktori kerja. Pernyataan insert bisa digunakan ketika pengguna ingin mengetikkan dataset secara langsung dan manual, sedangkan pernyataan browse bisa digunakan untuk menelusuri file data frame yang ada di komputer secara langsung dan cepat. Dengan demikian, pernyataan browse menjadi pilihan terbaik bagi programmer yang ingin memasukkan dataset dengan mudah dan cepat. Fungsi memiliki peran penting dalam pemrograman karena membantu programmer untuk tidak menulis ulang semua syntax di dalam suatu program. Fungsi didefinisikan dengan domain input, beberapa proses, dan nilai output. Fungsi bisa diketikkan di R Console dan R Editor, tetapi pilihan terbaik jatuh pada penulisan fungsi di R Editor karena bisa disimpan dan dijalankan kembali. Di dalam software R ada beberapa fungsi yang sudah built-in seperti fungsi-fungsi Matematika dan Statistika.

E. Soal Latihan

1. Lakukan penerapan terkait perintah import dan export pada dataset yang anda miliki !
2. Buatlah fungsi yang menampilkan vektor yang berisi:
   a. sepuluh bilangan genap !
   b. tujuh bilangan asli kelipatan 7 !
3. Buatlah fungsi Bola yang didefinisikan dengan domain input jari-jari bola untuk menghitung Luas Permukaan dan Volume bola !
4. Dengan menggunakan data latihan2.txt, buatlah fungsi StatDescrip yang menampilkan tabel statistika deskriptif yang berisi nilai banyaknya data, minimal, maksimal, rata-rata, dan standar deviasi !
5. Gunakan fungsi StatDescrip untuk menampilkan statistika deskriptif dari nilai Matematika, B.Indonesia, B.Inggris, dan IPK pada data latihan2.txt !

F. Sumber Referensi

Crawley, M. J. (2013). *The R Book (2nd Ed).* John Wiley & Sons.

Sarmento, R., & Costa, V. (2017*). Comparative Approaches to Using R and Python for Statistical Data Analysis*. IGI Global.

Suhartono. (2008). *Analisis Data Statistik Dengan R*. Lab. Statistik Komputasi ITS.

Team, D. S. (2021). *How to Export Data from R.* https://universeofdatascience.com/how-to-export-data-from-r/

*****

BAB 4 PERNYATAAN INPUT-OUTPUT DAN OPERATOR SOFTWARE R

A. Deskripsi Singkat

Materi pada bab ini merupakan pendalaman materi terkait proses manajemen data di dalam *software* R dengan memberikan pemaparan materi tentang perintah *output* dan *input*, serta pengenalan operator R meliputi: operator aritmatika, operator relasi, operator logika, dan operator penugasan. Bahasan ini diberikan dengan penjelasan singkat dan contoh penerapan bagaimana membuat *syntax* yang benar dalam pemrograman khususnya penggunaan perintah *output* dan *input* dalam beberapa fungsi yang tersedia di *software* R, serta pemahaman operator-operator di dalam *software* R. Pada bab ini diharapkan pembaca dapat memahami dan menerapkan bagaimana memanfaatkan penggunaan dari pernyataan input-output dan operator di *software* R. Materi pada bab ini berguna untuk mengetahui bagaimana manajemen *dataset* dan memanfaatkan operator yang sesuai dengan keinginan *programmer* di lingkungan pemrograman bahasa R.

B. Tujuan Pembelajaran

Pada bab ini, diberikan tujuan pembelajaran agar pembaca mampu menjelaskan dan menerapkan bagaimana:

1. Perintah output.
2. Perintah input.
3. Penggunaan operator R.

C. Penyajian Materi

1. Perintah Output

   Output suatu program komputer adalah tujuan utama *programmer* dalam membuat program komputer yang dituangkan ke dalam suatu algoritma dan *syntax* program.

Dalam *software* R, output program bisa berupa pernyataan string, nilai suatu variabel (atau objek data), dan tampilan suatu grafik.

```
> cat("Selamat Belajar Bahasa R \n")
Selamat Belajar Bahasa R
> print(pi)
[1] 3.141593
```

Fungsi yang sering digunakan untuk menampilkan output pernyataan atau *statement* dalam *Software* R adalah fungsi **cat**, sedangkan untuk menampilkan suatu variabel adalah fungsi **print**. Untuk mengetahui penggunaan fungsi ini lebih dalam, diberikan contoh penerapan sederhana dalam algoritma, *syntax*, dan output program berikut.

Algoritma program untuk Fungsi **Output** :

1) menentukan nilai variabel **a = 25.77** dan **b = 23.45**
2) menampilkan output variabel **a**
3) menampilkan output variabel **b**
4) melakukan operasi aritmatika yaitu **c = a * b**
5) menampilkan output variabel **c**

Syntax program yang diketikkan di *R Editor*

```
Output = function()
{
 a = 25.77
 b = 23.45
 cat(" Nilai A : ",a)
 cat("\n Nilai B : ",b)
 c = a*b
 cat("\n Nilai C = A*B = ",c)
 cat("\n Hasil print(c) : \n")
 print(c)
}
```

Output program yang ditampilkan di *R Console*

```
> Output()
 Nilai A :  25.77
 Nilai B :  23.45
 Nilai C = A*B =  604.3065
 Hasil print(c) :
[1] 604.3065
>
```

Fungsi **Output** diberikan tanpa inputan karena ditentukan langsung nilai variabel **a** dan **b** dalam fungsi tersebut sesuai pada Langkah ke-1. Pada fungsi **cat** pertama, diberikan dua output berurutan dengan dipisahkan tanda koma (**,**) yaitu: *statement* **" Nilai A : "** dan variabel **a** . Fungsi **cat** ini menyelesaikan Langkah ke-2 yang mirip dengan Langkah ke-3. Penulisan fungsi **cat** kedua dan seterusnya ditambahkan operator pindah baris yaitu **\n** agar tampilan output program bisa tertata dengan baik (*escape sequences*). Pada Langkah ke-4, digunakan operasi aritmatika yaitu **c = a * b** . Dengan demikian, Langkah ke-5 yang menampillkan variabel **c** dapat diwakili dengan penggunaan fungsi **cat** dan **print**. Fungsi **print** ini menampilkan output variabel **c** secara individu. Sebagai tambahan, *programmer* dapat memasukkan suatu komentar di dalam *syntax*-nya setelah tanda pagar (#). Komentar ini tidak akan dieksekusi dalam suatu program, namun bermanfaat sebagai penanda dan penjelas suatu *syntax* pemrograman.

```
> #Ini hanya komentar yang tidak akan dieksekusi
> cat("Pernyataan Komentar \n")
Pernyataan Komentar
>
```

2. Perintah Input.

Input adalah masukan yang diberikan oleh pengguna dalam rangka mengetahui output program berdasarkan proses atau operasi yang telah terprogram. Input data secara cepat dengan beberapa data dalam suatu vektor di awal penulisan buku ini sudah dilakukan dengan fungsi *concatenate* (**c**) dan disimpan dalam suatu variabel seperti contoh di *R Console* ini.

```
> G = c(2,8,4,7,3,6,5)
> G
[1] 2 8 4 7 3 6 5
```

serta input data juga bisa dilakukan dengan fungsi **scan** yang memasukkan data secara satu per satu kemudian berhenti ketika tidak diisi (Crawley, 2013).

```
> x = scan()
1: 3
2: 5
3: 3
4: 6
5: 7
6: 8
7:
Read 6 items
> x
[1] 3 5 3 6 7 8
```

Jika input data dilakukan hanya satu kali saja dalam suatu momen, maka bisa digunakan fungsi **readline**. Fungsi ini menyimpan data yang masuk setelah pernyataan tertentu, kemudian input akan dikenali sebagai data dengan mode data karakter. Jika input data berupa numerik, maka perlu ditambahkan fungsi **as.numeric** agar input dikenali dengan mode data numerik. Berikut diberikan contoh penerapan sederhana dalam algoritma, *syntax*, dan output program.

Algoritma program untuk Fungsi **IO** :

1) menginputkan nilai variabel **nm**.
2) menginputkan nilai variabel **a**.
3) menginputkan nilai variabel **b**.
4) melakukan operasi aritmatika yaitu **c = a / b**.
5) menampilkan output variabel **nm**.
6) menampilkan output variabel **c**.

Output program yang ditampilkan di *R Console*

```
IO = function()
{
 nm = readline(" Nama : ")
 a = as.numeric(readline(" Masukkan Nilai A : "))
 b = as.numeric(readline(" Masukkan Nilai B : "))
 c = a/b
 cat("\n Hai : ",nm)
 cat("\n Nilai C = A/B = ",c)
 cat("\n")
}
```

Output program yang ditampilkan di *R Console*

```
> IO()
 Nama : Suranto
 Masukkan Nilai A : 7
 Masukkan Nilai B : 3

 Hai :  Suranto
 Nilai C = A/B =  2.333333
>
```

Misalkan diberikan fungsi **IO** dengan Langkah ke-1 yaitu memasukkan nilai nama ke dalam variabel **nm** dengan memanfaatkan fungsi **readline** yang akan memasukkan data setelah *statement* **" Nama : "** . Nilai variabel **nm** jelas akan memiliki mode data karakter. Pada Langkah ke-2 dan ke-3 diberikan penambahan fungsi **as.numeric** agar data yang diinput disimpan dalam mode data numerik, sehingga bisa dilakukan Langkah ke-4 yaitu operasi aritmatika **c = a/b** yang dilanjutkan dengan output program sesuai Langkah ke-5 dan ke-6. Sebagai tambahan, diberikan *syntax* **cat("\n")** yang berguna agar panah *command* di *R Console* tidak berhenti di kanan output (bisa pindah ke bawah). Jika fungsi **IO** mengembalikan suatu nilai saja tanpa begitu dibutuhkan suatu *statement*, maka fungsi **IO** dapat disertai input fungsi dan disederhanakan sebagai berikut.

Syntax program yang diketikkan di *R Editor*

```
IO = function(a,b)
{
  return(a/b)
}
```

Output program yang ditampilkan di *R Console*

```
> IO(7,3)
[1] 2.333333
>
```

Contoh perubahan ini diawali dengan *syntax* **IO = function(a,b)** dengan **a** dan **b** adalah input *general* yang diberikan untuk fungsi **IO** yang mana akan dikembalikan nilainya dengan operasi **a/b** dengan bantuan fungsi **return**.

Demikian hasil fungsi, ditampilkan perintah di *R Console* yaitu **IO(7,3)** yang berarti input **a = 7** dan input **b = 3** akan mengembalikan nilai **a/b = 7/3** yang bernilai **2.333333** .

3. Penggunaan Operator R.

Dalam *software* R, operator merupakan suatu simbol yang memberikan perintah untuk melakukan operasi matematika atau logika tertentu. *Software* R dilengkapi dengan beberapa operator yang dapat digunakan untuk tugas-tugas tersebut. Operator R diklasifikasikan sebagai berikut (Bell, 2020):

a. Operator Aritmatika, ini merupakan jenis operator yang digunakan untuk melakukan operasi matematika seperti, penambahan (+), pengurangan (-), pembagian (/), perkalian (*), eksponen (^), modulus (**%%**), dan pembagi intejer (%/%). Misalkan diberikan contoh penerapannya sebagai berikut.

```
> x = 7; y = 3
> cat("x + y = ",x+y,"\n")
x + y =  10
> cat("x - y = ",x-y,"\n")
x - y =  4
> cat("x * y = ",x*y,"\n")
x * y =  21
> cat("x / y = ",x/y,"\n")
x / y =  2.333333
> cat("x %% y = ",x%%y,"\n")
x %% y =  1
> cat("x %/% y = ",x%/%y,"\n")
x %/% y =  2
```

Urutan operasi matematika yang dikerjakan terlebih dulu adalah dari kiri ke kanan dengan eksponen dikerjakan dulu baru lanjut pembagian/perkalian setelah itu operasi penjumlahan/pengurangan dikerjakan. Diketahui **x = 7** dan **y = 3**, maka **x %% y = 1** ini diartikan bahwa **7** modulo **3** sama dengan **1** karena **7 / 3 = 2.333333** yang mana hanya **2** yang bisa membagi (pembagi bulat) dan **3 * 2 = 6** kemudian hasil bagi sisanya diperoleh dari **7 – 6 = 1.**

```
> 2+3*3^2/2-1
[1] 14.5
> (2+3)*3^2/2-1
[1] 21.5
```

Namun, jika dikehendaki penjumlahan dikerjakan dulu maka diberikan tanda kurung **( )**.

b. Operator Relasi, ini membantu pengguna membuat perbandingan antar nilai. Daftar operator relasi yang didukung dalam R, yaitu: lebih dari (>), kurang dari (<), lebih dari atau sama dengan (>=), kurang dari atau sama dengan (<=), sama dengan (==), dan tidak sama dengan (**!**=). Misalkan diberikan contoh penerapannya sebagai berikut.

```
> x = 8; y = 24
> cat(" x > y adalah ",x>y,"\n")
 x > y adalah  FALSE
> cat(" x < y adalah ",x<y,"\n")
 x < y adalah  TRUE
> cat(" x >= y adalah ",x>=y,"\n")
 x >= y adalah  FALSE
> cat(" x <= y adalah ",x<=y,"\n")
 x <= y adalah  TRUE
> cat(" x == y adalah ",x==y,"\n")
 x == y adalah  FALSE
> cat(" x != y adalah ",x!=y,"\n")
 x != y adalah  TRUE
```

c. Operator Logika, ini digunakan untuk melakukan operasi Boolean seperti operasi logika AND (**&** atau **&&**), OR (| atau ||) dan NOT (**!**). Operator logika hanya berlaku untuk numerik, logika, atau vektor. Misalkan diberikan contoh penerapannya sebagai berikut.

```
> x = c(TRUE,TRUE,FALSE,FALSE)
> y = c(TRUE,FALSE,TRUE,FALSE)
> cat(" x & y adalah ",x&y,"\n")
 x & y adalah  TRUE FALSE FALSE FALSE
> cat(" x && y adalah ",x&&y,"\n")
 x && y adalah  TRUE
> cat(" x | y adalah ",x|y,"\n")
 x | y adalah  TRUE TRUE TRUE FALSE
> cat(" x || y adalah ",x||y,"\n")
 x || y adalah  TRUE
> cat(" ! x adalah ",!x,"\n")
 ! x adalah  FALSE FALSE TRUE TRUE
```

Operator **&** dan | masing-masing adalah operator AND dan OR yang digunakan pada suatu vektor, sedangkan jika data hanya memuat satu elemen saja atau skalar maka operator yang cocok digunakan masing-masing adalah operator **&&** dan || (Matloff, 2011).

d. Operator Penugasan, ini digunakan untuk memberikan nilai pada variabel. *Software* R mendukung operator penugasan yaitu: operator penugasan ke kiri (=, <-, <<-) dan operator penugasan ke kanan (->, ->>).

```
> x <- 4
> x
[1] 4
> 10 -> x
> x
[1] 10
> x = 7
> x
[1] 7
```

D. Rangkuman

Perintah output akan menghasilkan output yang menjadi tujuan utama *programmer* dalam membuat program komputer yang dituangkan ke dalam suatu algoritma dan *syntax* program. Dalam *software* R, output bisa dihasilkan dengan bantuan fungsi **cat** dan **print**. Perintah input adalah masukan yang diberikan oleh pengguna dalam rangka mengetahui output program

berdasarkan proses atau operasi yang telah terprogram. Dalam *software* R, input pengguna bisa dilakukan dengan bantuan fungsi **c**, **scan**, dan **readline**. Fungsi **readline** umumnya akan membaca masukan data dengan mode data karakter, sehingga perlu ditambahkan fungsi **as.numeric** agar mode data menjadi numerik. Operator dalam *software* R dikenali dalam empat jenis operator yaitu operator aritmatika, operator relasi, operator logika, dan operator penugasan.

E. Soal Latihan

Misalkan diberikan nilai variabel **A**, **B**, **C**, **K**, **L**, dan **M** yang berupa input pengguna dalam data numerik .Tentukan apa hasil logika dari ekspresi relasi dan logika untuk variabel **D**, **N**, **O**, **P** dan **Q** !

D = (4 + 2 > A) & (B – 2 > 3 +2) | (C + 2 <= 6 + 2)
N = (K + 5 < M) | (C * M < L) & (2 * M – L > 0)
O = (L + 5 < M) | (C * K < L) & (2 * K – L > 0)
P = A * 4 <= 3 * M + B
Q = (K + 10 > A) & (L – 2 > 4 * C)

F. Sumber Referensi

Bell, D. (2020). *R Programming: A Step-by-Step Guide for Absolute Beginners* (2nd Ed). KDP Amazon Publishing.

Crawley, M. J. (2013). *The R Book* (2nd Ed). John Wiley & Sons.

Matloff, N. (2011). *The Art of R Programming: A Tour of Statistical Software Design*. No Starch Press.

*****

BAB 5

OPERASI PENYELEKSIAN KONDISI

A. Deskripsi Singkat

Materi pada bab ini merupakan pendalaman materi terkait proses penyeleksian kondisi di dalam *software* R dengan memberikan pemaparan materi tentang pernyataan pengandaian seperti: penyataan *if*, penyataan *if-else*, penyataan *nested if*, dan penyataan *nested if* majemuk. Bahasan ini diberikan dengan penjelasan singkat dan contoh penerapan bagaimana membuat *syntax* yang benar dalam pemrograman khususnya penggunaan operasi penyeleksian kondisi yang bisa dilakukan di *software* R. Pada bab ini diharapkan pembaca dapat memahami dan menerapkan bagaimana memanfaatkan penggunaan dari operasi penyeleksian kondisi di *software* R. Materi pada bab ini berguna untuk mengetahui bagaimana menyeleksi kondisi-kondisi dari suatu operasi atau proses yang diinginkan oleh *programmer* di lingkungan pemrograman bahasa R.

B. Tujuan Pembelajaran

Pada bab ini, diberikan tujuan pembelajaran agar pembaca mampu menjelaskan dan menerapkan bagaimana:

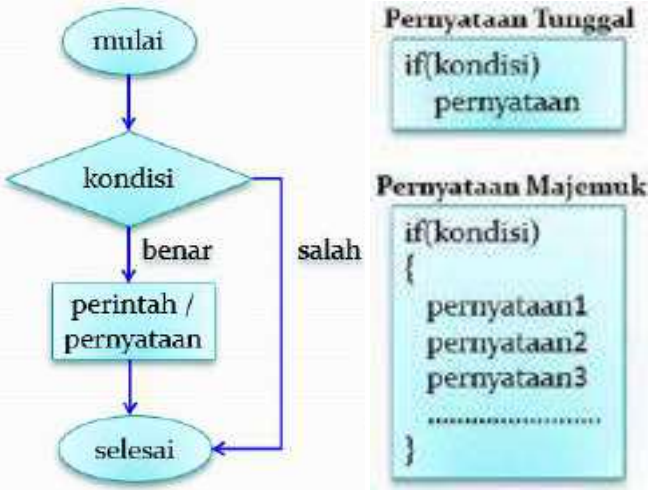
1. Pernyataan if
2. Pernyataan if-else
3. Pernyataan nested if
4. Pernyataan nested if majemuk

C. Penyajian Materi

1. Pernyataan *If*

   Pada bagian ini, diberikan bahasan bagaimana cara menggunakan berbagai pernyataan pengambilan keputusan dalam *software* R yang sangat penting dalam bahasa pemrograman. Pengambilan keputusan melibatkan pembuatan beberapa kondisi yang akan diuji oleh program bersama dengan pernyataan yang akan dieksekusi ketika suatu kondisi bernilai benar dan sebagai alternatif lainnya pernyataan lain akan dieksekusi ketika kondisinya salah

(Bell, 2020). Pernyataan *if* merupakan operasi penyeleksian kondisi yang paling sederhana. Pernyataan ini hanya memiliki satu kondisi saja yang jika kondisi benar terpenuhi, maka akan dilakukan suatu perintah/pernyataan; sedangkan jika kondisi tidak terpenuhi, maka proses akan selesai. Pernyataan ini digambarkan dalam diagram jalur berikut.



Gambar 5.1 Diagram Jalur Untuk Pernyataan *If*.

Dari Gambar 5.1, proses dimulai dengan satu kondisi/ syarat. Jika kondisi benar terpenuhi, maka akan dilakukan suatu perintah/pernyataan kemudian proses selesai; Namun jika kondisi tidak terpenuhi, maka proses akan langsung selesai. Jika suatu pernyataan ada lebih dari satu pernyataan atau pernyataan majemuk, maka hendaklah diberikan tanda kurung kurawal { }.

Untuk lebih memahami pernyataan *if*, diberikan contoh penerapan. Misalkan ada seorang penjual yang memberikan nilai potongan harga untuk barang yang dibeli dengan ketentuan/kondisi sebagai berikut:

a. Jika nilai pembelian barang lebih dari atau sama dengan Rp. 500.000 , maka akan diberikan potongan harga sebesar 10% dari nilai pembelian. Untuk menyelesaikan

kondisi ini, diberikan contoh penerapan sederhana dalam algoritma, syntax, dan output program berikut.

Algoritma Program untuk Fungsi **Contoh1** :

1) Menentukan nilai variabel **potongan = 0**
2) Menginputkan nilai variabel **beli**
3) Jika nilai variabel **beli >= 500000**, maka menentukan variabel **potongan = 0.10 * beli**
4) Menampilkan output variabel **potongan**
5) Melakukan operasi aritmatika yaitu **bayar = beli - potongan**
6) Menampilkan output variabel **bayar**

Syntax program yang diketikkan di *R Editor*

```
Contoh1 = function()
{
 potongan = 0
 beli = as.numeric(readline(" Total Pembelian Rp. "))
 if(beli >= 500000)
   potongan = 0.10 * beli
 cat(" Besarnya Potongan Rp. ",potongan,"\n")
 bayar = beli - potongan
 cat(" Jumlah yang harus dibayarkan Rp. ",bayar,"\n")
}
```

Output program yang ditampilkan di *R Console*

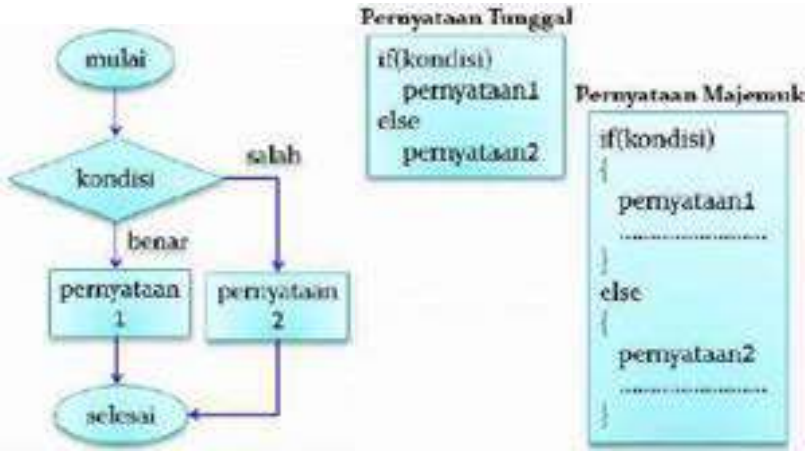
```
> Contoh1()
 Total Pembelian Rp. 600000
 Besarnya Potongan Rp.  60000
 Jumlah yang harus dibayarkan Rp.  540000
> Contoh1()
 Total Pembelian Rp. 400000
 Besarnya Potongan Rp.  0
 Jumlah yang harus dibayarkan Rp.  4e+05
```

Syntax if(beli >= 500000) akan menyeleksi nilai variabel beli lebih dari atau sama dengan 500000. Jika kondisi benar terpenuhi, maka akan ada potongan sebesar 10% dari total pembelian; tidak ada potongan harga jika kondisi tidak terpenuhi. Sebagai tambahan, nilai 4e+05 diartikan sebagai nilai sainstifik dari 4 * 105 atau 400000. Ketidaknyamanan penulisan ini bisa diatasi dengan memasukan syntax options("scipen"=100, "digits"=4) di R Console.

```
> options("scipen"=100, "digits"=4)
> Contoh1()
 Total Pembelian Rp. 400000
 Besarnya Potongan Rp.  0
 Jumlah yang harus dibayarkan Rp.  400000
```

2. Pernyataan *If-Else*.

Pernyataan *if-else* adalah operasi penyeleksian kondisi yang memiliki satu kondisi dengan dua pilihan penyataan. Jika kondisi benar terpenuhi, maka akan dilakukan pernyataan pertama; Sedangkan pernyataan kedua akan dilakukan jika kondisi tidak terpenuhi. Pernyataan ini digambarkan dalam diagram jalur berikut.



Gambar 5.2 Diagram Jalur Untuk Pernyataan *if-else*.

Dari Gambar 5.2, proses dimulai dengan satu kondisi. Jika kondisi benar terpenuhi, maka akan dilakukan suatu pernyataan pertama kemudian proses selesai; Namun pernyataan kedua akan dilakukan jika kondisi tidak terpenuhi, kemudian proses selesai. Jika suatu pernyataan ada lebih dari satu pernyataan atau pernyataan majemuk, maka hendaklah diberikan tanda kurung kurawal { }.

Untuk lebih memahami pernyataan *if-else*, diberikan contoh penerapan. Misalkan ada seorang penjual yang

memberikan nilai potongan harga untuk barang yang dibeli dengan ketentuan/kondisi sebagai berikut:

a. Jika nilai pembelian barang kurang dari Rp. 500.000 , maka akan diberikan potongan harga sebesar 2.5% dari nilai pembelian.
b. Jika nilai pembelian barang lebih dari atau sama dengan Rp. 500.000 , maka akan diberikan potongan harga sebesar 10% dari nilai pembelian.

Untuk menyelesaikan kondisi ini, diberikan contoh penerapan sederhana dalam algoritma, *syntax*, dan output program berikut.

Algoritma Program untuk Fungsi **Contoh2** :

1) menginputkan nilai variabel **beli**
2) jika nilai variabel **beli >= 500000**, maka menentukan
   BAB 1. variabel **potongan = 0.10 * beli**
3) jika nilai variabel beli < 500000, maka menentukan
   BAB 2. variabel **potongan = 0.025 * beli**
4) menampilkan output variabel **potongan**
5) melakukan operasi aritmatika yaitu **bayar = beli - potongan**
6) menampilkan output variabel **bayar**.

Syntax program yang diketikkan di *R Editor*

```
Contoh2 = function()
{
 beli = as.numeric(readline(" Total Pembelian Rp. "))
 if(beli >= 500000)
   potongan = 0.10 * beli
 else
   potongan = 0.025 * beli
 cat(" Besarnya Potongan Rp. ",potongan,"\n")
 bayar = beli - potongan
 cat(" Jumlah yang harus dibayarkan Rp. ",bayar,"\n")
}
```

Output program yang ditampilkan di *R Console*

```
> Contoh2()
 Total Pembelian Rp. 500000
 Besarnya Potongan Rp.  50000
 Jumlah yang harus dibayarkan Rp.  450000
> Contoh2()
 Total Pembelian Rp. 400000
 Besarnya Potongan Rp.  10000
 Jumlah yang harus dibayarkan Rp.  390000
```

Syntax **if(beli >= 500000)** akan menyeleksi nilai variabel **beli** lebih dari atau sama dengan **500000**. Jika kondisi benar terpenuhi, maka akan ada potongan sebesar **10%** dari total pembelian; Namun jika kondisi tidak terpenuhi, maka akan ada potongan sebesar **2.5%** dari total pembelian. Sebagai tambahan, diberikan contoh penetapan fungsi **ifelse** yang berguna untuk pengandaian biner sederhana (Crawley, 2013).

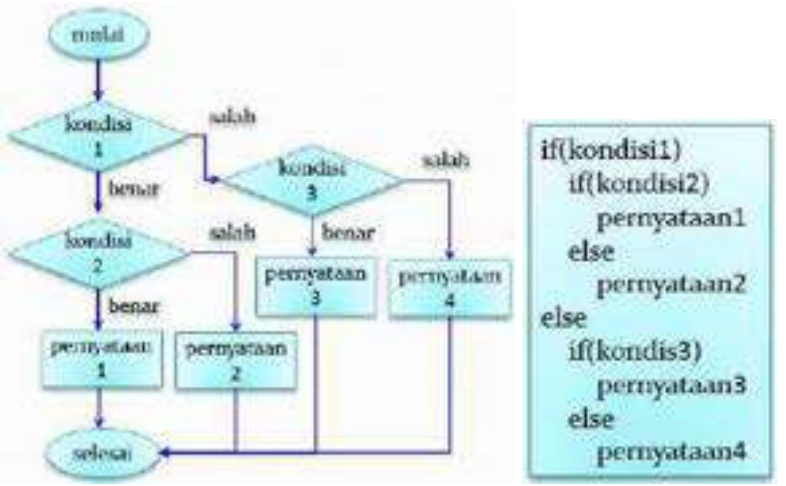
```
> y <- c(8,3,5,7,6,6,8,9,2,3,9,4,10,4,11)
> y
 [1]  8  3  5  7  6  6  8  9  2  3  9  4 10  4 11
> mu = mean(y)
> mu
[1] 6.333333
> binary = ifelse(y < mu,1,2)
> binary
 [1] 2 1 1 2 1 1 2 2 1 1 2 1 2 1 2
```

Dengan nilai **mu** adalah nilai rata-rata dari vektor **y**, pengguna dapat memanfaatkan *syntax* **ifelse(y < mu,1,2)** yang mengembalikan nilai **1** jika kondisi terpenuhi, atau mengembalikan nilai **2** jika kondisi tidak terpenuhi. *Syntax* ini biasanya digunakan untuk memberikan indeks yang berguna untuk membagi suatu *dataset* menjadi dua kondisi.

3. Pernyataan *Nested If*.

Pernyataan *nested if* adalah operasi penyeleksian kondisi yang memiliki suatu kondisi yang di dalamnya bersarang beberapa kondisi yang memiliki pilihan dari beberapa penyataan. Jika kondisi pertama benar terpenuhi, maka kondisi kedua akan diberikan dengan pilihan

pernyataan; Sedangkan kondisi ketika akan diberikan dengan pilihan pernyataan jika kondisi pertama tidak terpenuhi. Pernyataan ini digambarkan dalam diagram jalur berikut.



Gambar 5.3 Diagram Jalur Untuk Pernyataan *nested if*.

Dari Gambar 5.3, proses dimulai dengan satu kondisi. Jika kondisi pertama benar terpenuhi, maka kondisi kedua akan ditanyakan; Namun jika kondisi pertama tidak terpenuhi, maka kondisi ketiga akan ditanyakan. Jika kondisi kedua benar terpenuhi, maka penyataan pertama akan dilakukan kemudian proses selesai; Namun jika kondisi kedua tidak terpenuhi, maka penyataan kedua akan dilakukan kemudian proses selesai. Jika kondisi ketiga benar terpenuhi, maka penyataan ketiga akan dilakukan kemudian proses selesai; Namun jika kondisi ketiga tidak terpenuhi, maka penyataan keempat akan dilakukan kemudian proses selesai.

Untuk lebih memahami pernyataan *nested if*, diberikan contoh penerapan. Misalkan ada suatu perusahaan *Furniture* atau Mebel akan menentukan komisi kepada *Salesman*-nya dengan kriteria atau kondisi tertentu:

a. Jika *Salesman* berhasil menjual barang hingga Rp. 200.000 , maka uang jasa dan uang komisi akan diberikan

masing-masing sebesar Rp. 20.000 dan 2.5% dari pendapatan hari ini.

b. Jika *Salesman* berhasil menjual barang lebih dari Rp. 200.000 dan maksimal Rp. 500.000 , maka uang jasa dan uang komisi akan diberikan masing-masing sebesar Rp. 30.000 dan 5% dari pendapatan hari ini.
c. Jika *Salesman* berhasil menjual barang di atas Rp. 500.000, maka uang jasa sebesar dan uang komisi akan diberikan masing-masing sebesar Rp. 40.000 dan 10% dari pendapatan hari ini.

Untuk menyelesaikan kondisi ini, diberikan contoh penerapan sederhana dalam algoritma, *syntax*, dan output program berikut.

Algoritma Program untuk Fungsi **Contoh3** :

1) Menginputkan nilai variabel **pendapatan**
2) Jika nilai variabel **pendapatan ≥ 0 dan pendapatan ≤200000**, maka menentukan variabel **jasa = 20000** dan **komisi** = **0.025 * pendapatan**
3) Jika nilai variabel **pendapatan > 200000 dan pendapatan ≤ 500000**, maka menentukan variabel **jasa = 30000** dan **komisi = 0.05 * pendapatan**
4) Jika nilai variabel **pendapatan > 500000**, maka menentukan variabel **jasa = 40000** dan **komisi = 0.10 * pendapatan**
5) Melakukan operasi aritmatika **total** = **jasa** + **komisi**
6) Menampilkan output variabel **jasa**, **komisi**, dan **total**

Syntax program yang diketikkan di *R Editor*

```
Contoh3 = function()
{
 pendapatan = as.numeric(readline(" Pendapatan Hari ini Rp. "))
 if(pendapatan >= 0 & pendapatan <= 200000)
 {
   jasa = 20000
   komisi = 0.025 * pendapatan
 }
 else
 {
  if(pendapatan > 200000 & pendapatan <= 500000)
  {
    jasa = 30000
    komisi = 0.05 * pendapatan
  }
  else
  {
    jasa = 40000
    komisi = 0.10 * pendapatan
  }
 }
 total = komisi + jasa
 cat(" Uang Jasa Rp. ",jasa,"\n")
 cat(" Uang Komisi Rp. ",komisi,"\n")
 cat(" ================================\n")
 cat(" Hasil Total Rp. ",total,"\n")
}
```

Output program yang ditampilkan di *R Console*

```
> Contoh3()
 Pendapatan Hari ini Rp. 200000
 Uang Jasa Rp.  20000
 Uang Komisi Rp.  5000
 ================================
 Hasil Total Rp.  25000
> Contoh3()
 Pendapatan Hari ini Rp. 400000
 Uang Jasa Rp.  30000
 Uang Komisi Rp.  20000
 ================================
 Hasil Total Rp.  50000
> Contoh3()
 Pendapatan Hari ini Rp. 600000
 Uang Jasa Rp.  40000
 Uang Komisi Rp.  60000
 ================================
 Hasil Total Rp.  100000
```

Syntax **if(pendapatan >= 0 & pendapatan <= 200000)** akan menyeleksi nilai variabel **pendapatan** maksimal sama dengan **200000**. Jika kondisi benar terpenuhi, maka *Salesman* akan dapat uang jasa sebesar **20000** dan komisi sebesar **2.5%** dari pendapatan; Namun kondisi kedua diberlakukan jika kondisi tidak terpenuhi. Kondisi kedua dengan *syntax* **if(pendapatan > 200000 & pendapatan <= 500000)** akan menyeleksi nilai variabel **pendapatan** diatas **200000** dan maksimal sama dengan **500000**. Jika kondisi kedua benar terpenuhi, maka *Salesman* akan dapat uang jasa sebesar **30000** dan komisi sebesar **5%** dari pendapatan; Namun jika kondisi kedua tidak terpenuhi, maka *Salesman* akan dapat uang jasa sebesar **40000** dan komisi sebesar **10%** dari pendapatan.

4. Pernyataan *Nested If* Majemuk.

   Pernyataan *nested if* majemuk adalah operasi penyeleksian kondisi yang memiliki suatu kondisi yang di dalamnya bersarang jika kondisi tersebut tidak terpenuhi yang mana juga memiliki pilihan dari beberapa penyataan. Jika kondisi pertama benar terpenuhi, maka suatu pernyataan akan dilakukan; Sedangkan jika kondisi pertama tidak terpenuhi, maka kondisi kedua akan diberlakukan dengan pilihan pernyataan. Pernyataan ini digambarkan dalam diagram jalur berikut.

Gambar 5.4
Diagram Jalur Untuk Pernyataan *nested if* Majemuk

Dari Gambar 5.4, proses dimulai dengan satu kondisi. Jika kondisi pertama benar terpenuhi, maka pernyataan pertama akan dilakukan kemudian proses langsung selesai; Namun jika kondisi pertama tidak terpenuhi, maka kondisi kedua akan dipertanyakan. Jika kondisi kedua benar terpenuhi, maka pernyataan kedua akan dilakukan kemudian proses langsung selesai; Namun jika kondisi kedua tidak terpenuhi, maka pernyataan ketiga akan dilakukan kemudian proses langsung selesai.

Untuk lebih memahami pernyataan *nested if* majemuk, diberikan contoh penerapan. Misalkan ada suatu perusahaan *Furniture* atau Mebel akan menentukan komisi kepada *Salesman*-nya dengan kriteria atau kondisi tertentu:

a. Jika *Salesman* berhasil menjual barang hingga Rp. 200.000 , maka uang jasa dan uang komisi akan diberikan masing-masing sebesar Rp. 20.000 dan 2.5% dari pendapatan hari ini.

b. Jika *Salesman* berhasil menjual barang lebih dari Rp. 200.000 dan maksimal Rp. 500.000 , maka uang jasa dan uang komisi akan diberikan masing-masing sebesar Rp. 30.000 dan 5% dari pendapatan hari ini.
c. Jika *Salesman* berhasil menjual barang dari Rp. 500.000 dan maksimal Rp. 1.000.000 , maka uang jasa dan uang komisi akan diberikan masing-masing sebesar Rp. 40.000 dan 10% dari pendapatan hari ini.
d. Jika *Salesman* berhasil menjual barang diatas Rp. 1.000.000 , maka uang jasa dan uang komisi akan diberikan masing-masing sebesar Rp. 50.000 dan 15% dari pendapatan hari ini.

Untuk menyelesaikan kondisi ini, diberikan contoh penerapan sederhana dalam algoritma, *syntax*, dan output program berikut.

Algoritma Program untuk Fungsi **Contoh4** :

1) menginputkan nilai variabel **pendapatan**
2) jika nilai variabel **pendapatan ≥ 0 dan pendapatan ≤ 200000**, maka menentukan variabel **jasa = 20000** dan **komisi = 0.025 * pendapatan**
3) jika nilai variabel **pendapatan > 200000 dan pendapatan ≤ 500000**, maka menentukan variabel **jasa = 30000** dan **komisi = 0.05 * pendapatan**
4) jika nilai variabel **pendapatan > 500000 dan pendapatan ≤ 1000000**, maka menentukan variabel **jasa = 40000** dan **komisi = 0.10 * pendapatan**
5) jika nilai variabel **pendapatan > 1000000**, maka menentukan variabel **jasa = 50000** dan **komisi = 0.15 * pendapatan**
6) melakukan operasi aritmatika **total = jasa + komisi**
7) menampilkan output variabel **jasa**, **komisi**, dan **total**

Syntax program yang diketikkan di *R Editor*

```
Contoh4 = function()
{
 pendapatan = as.numeric(readline(" Pendapatan Hari ini Rp. "))
 if(pendapatan >= 0 & pendapatan <= 200000)
 {
   jasa = 20000
   komisi = 0.025 * pendapatan
 }
 else if(pendapatan > 200000 & pendapatan <= 500000)
 {
   jasa = 30000
   komisi = 0.05 * pendapatan
 }
 else if(pendapatan > 500000 & pendapatan <= 1000000)
 {
   jasa = 40000
   komisi = 0.10 * pendapatan
 }
 else
 {
   jasa = 50000
   komisi = 0.20 * pendapatan
 }
 total = komisi+jasa
 cat(" Uang Jasa   Rp. ",jasa,"\n")
 cat(" Uang Komisi Rp. ",komisi,"\n")
 cat(" ================================\n")
 cat(" Hasil Total Rp. ",total,"\n")
}
```

Output program yang ditampilkan di *R Console*

```
> Contoh4()
 Pendapatan Hari ini Rp. 100000
 Uang Jasa    Rp.  20000
 Uang Komisi Rp.  2500
 ================================
 Hasil Total Rp.  22500
> Contoh4()
 Pendapatan Hari ini Rp. 300000
 Uang Jasa    Rp.  30000
 Uang Komisi Rp.  15000
 ================================
 Hasil Total Rp.  45000
> Contoh4()
 Pendapatan Hari ini Rp. 600000
 Uang Jasa    Rp.  40000
 Uang Komisi Rp.  60000
 ================================
 Hasil Total Rp.  100000
> Contoh4()
 Pendapatan Hari ini Rp. 1500000
 Uang Jasa    Rp.  50000
 Uang Komisi Rp.  300000
 ================================
 Hasil Total Rp.  350000
```

Syntax **if(pendapatan >= 0 & pendapatan <= 200000)** akan menyeleksi nilai variabel **pendapatan** maksimal sama dengan **200000**. Jika kondisi benar terpenuhi, maka *Salesman* akan dapat uang jasa sebesar **20000** dan komisi sebesar **2.5%** dari pendapatan; Namun kondisi kedua akan ditanyakan jika kondisi tidak terpenuhi. Kondisi kedua dengan *syntax* **else if(pendapatan > 200000 & pendapatan <= 500000)** akan menyeleksi nilai variabel **pendapatan** diatas **200000** dan maksimal sama dengan **500000**. Jika kondisi kedua benar terpenuhi, maka *Salesman* akan dapat uang jasa sebesar **30000** dan komisi sebesar **5%** dari pendapatan; Namun jika kondisi kedua tidak terpenuhi, maka *Salesman* akan ditanyakan kondisi ketiga. Kondisi ketiga dengan *syntax* **else if(pendapatan > 500000 & pendapatan <= 1000000)** akan menyeleksi nilai variabel **pendapatan** diatas **500000** dan maksimal sama dengan **1000000**. Jika kondisi ketiga benar terpenuhi, maka *Salesman* akan dapat uang jasa sebesar **40000** dan komisi sebesar **10%** dari pendapatan; Namun jika kondisi ketiga tidak terpenuhi, maka *Salesman* akan dapat uang jasa sebesar **50000** dan komisi sebesar **20%** dari pendapatan.

D. Rangkuman

Operasi penyeleksian kondisi mewakili pengandaian seorang *programmer*. Operasi penyeleksian kondisi dalam *software* R diberikan seperti: penyataan *if*, penyataan *if-else*, penyataan *nested if*, dan penyataan *nested if* majemuk. Pernyataan *if* merupakan operasi penyeleksian kondisi yang paling sederhana dengan satu kondisi dan satu penyataan, sedangkan pernyataan *if-else* adalah operasi penyeleksian kondisi yang memiliki satu kondisi dengan dua pilihan penyataan. Pengandaian bersarang diberikan pada penyataan *nested if* dan pernyataan *nested if* majemuk. Pernyataan *nested if* adalah operasi penyeleksian kondisi yang memiliki suatu

kondisi yang di dalamnya bersarang beberapa kondisi yang lain yang mana juga memiliki pilihan dari beberapa penyataan, sedangkan pernyataan *nested if* majemuk adalah operasi penyeleksian kondisi yang memiliki suatu kondisi yang di dalamnya bersarang jika kondisi tersebut tidak terpenuhi yang mana juga memiliki pilihan dari beberapa penyataan.

E. Soal Latihan

Diberikan latihan untuk membuat program atau fungsi yang menghitung nilai rata-rata pertandingan untuk seorang mahasiswa dengan kriteria tertentu:

1. Menginputkan nama mahasiswa, nilai pertandingan pertama, nilai pertandingan kedua, dan nilai pertandingan ketiga.
2. Menghitung nitai rata-rata pertandingan yaitu nilai jumlahan dari nilai pertandingan pertama, nilai pertandingan kedua, dan nilai pertandingan ketiga; yang mana kemudian dibagi dengan angka tiga.
3. Menentukan hadiah pertandingan dengan ketentuan berikut:
   a. Jika nilai rata-rata ≥ 90, maka mahasiswa akan memeperoleh hadiah uang sebesar Rp. 3.000.000.
   b. Jika diperoleh nilai rata-rata ≥ 80 dan rata-rata < 90, maka mahasiswa akan mendapat uang sebesar Rp. 1.500.000.
   c. Jika diperoleh nilai rata-rata ≥ 70 dan rata-rata < 80, maka mahasiswa akan mendapat uang sebesar Rp. 750.000.
   d. jika nilai rata-rata < 70, maka Mahasiswa akan memperoleh hadiah hiburan.

F. Sumber Referensi

Bell, D. (2020). *R Programming: A Step-by-Step Guide for Absolute Beginners* (2nd Ed). KDP Amazon Publishing.

Crawley, M. J. (2013). *The R Book* (2nd Ed). John Wiley & Sons.

*****

Denny Nurdiansyah

Penulis lahir di Surabaya, 26 Mei 1987 dengan riwayat pendidikan SD, SLTP, dan SMU di Kota Surabaya, Jawa Timur. Kemudian tahun 2005 melanjutkan pendidikan S-1 di Departemen Matematika, Universitas Airlangga, Surabaya. Setelah lulus, penulis mulai tahun 2010 bekerja di bidang Jasa Pengolahan Data Statistik hingga sekarang sebagai Konsultan Statistik. Tahun 2012 penulis melanjutkan pendidikan S-2 di Pasca Sarjana Program Studi Statistika, Institut Teknologi Sepuluh Nopember, Surabaya. Penulis mulai bekerja sebagai Dosen Statistika tahun 2017 hingga sekarang di Program Studi Statistika dan Sains Data, Universitas Nahdlatul Ulama Sunan Giri, Bojonegoro. Sekarang penulis juga sedang aktif sebagai *Chief Editor* di Jurnal Statistika dan Komputasi (STATKOM). Penulis pernah mengampu matakuliah di bidang Statistika dan Teknik Informatika, meliputi: Pemrograman Komputer, Statistika Komputasi, Metode Numerik, Teknik Simulasi, Sistem Pendukung Keputusan, *Machine Learning*, *Data Mining*, Basis Data, *Big Data Analytics*, Sistem Informasi Manajemen, Analisis Deret Waktu, Proses Stokastik, Ekonometrika, Metode Riset Sosial, dan *Statistical Consulting*. Penulis bisa dihubungi melalui email: *nur.denny@gmail.com.*

***

Agus Sulistiawan

Penulis lahir di Bojonegoro, 24 September 1991 dengan riwayat pendidikan SD, SLTP di Bojonegoro dan SMU di Tuban, Jawa Timur. Kemudian tahun 2010 melanjutkan pendidikan S-1 Pendidikan Teknik Mesin di Fakultas Teknik, Universitas Negeri Surabaya. Setelah lulus, penulis mulai tahun 2015 bekerja di Sekolah Menengah Kejuruan (SMK). Tahun 2015 penulis melanjutkan pendidikan S-2 di Pasca Sarjana Program Studi Magister Teknik Industri, Institut Teknologi Adhi Tama Surabaya serta masih menempuh program Profesi Insinyur di Institut teknologi Sepuluh Nopember Surabaya tahun 2023 hingga saat ini. Penulis mulai bekerja sebagai Dosen Fakultas Sains dan Teknologi tahun 2017 hingga sekarang di Program Studi Teknik Mesin, Universitas Nahdlatul Ulama Sunan Giri, Bojonegoro. Sekarang penulis juga sedang aktif sebagai *Editorial Team* di Publisher Arta Puri Kencana. Penulis pernah mengampu matakuliah di Teknik Mesin dan Teknik Informatika, meliputi: Matematika Diskrit, Algoritma Pemrograman, Matematika Rekayasa, Instrumentasi & Pengukuran, Statistika & Perencanaan Eksperimen, dan Manajemen Operasional. Selain Itu penulis juga sebagai Pebisnis di bidang Mebeler dan Furniture. Penulis bisa dihubungi melalui email: *agus.dmc354@gmail.com*.

*****