KOMPUTER 1

Unversitas Terbuka



# Daftar Isi

[Daftar Isi 2](#_Toc78710638)

[1. Modul 1 : Pengenalan dan Instalasi Perangkat Lunak dan Lingkungan Pemrograman R 3](#_Toc78710639)

[Pendahuluan 3](#_Toc78710640)

[Perangkat lunak & Lingkungan Pemrograman R 3](#_Toc78710641)

[Menjalankan R 5](#_Toc78710642)

[Fasilitas Help 8](#_Toc78710643)

[Bantuan Online Help 9](#_Toc78710644)

[Demo pada R *Functions* 10](#_Toc78710645)

[*Libraries* dalam bahasa R 10](#_Toc78710646)

[Instalasi Packages 11](#_Toc78710647)

[Command Line Versus Script 12](#_Toc78710648)

[Data Editor 12](#_Toc78710649)

[Perubahan Tampilan Layar Dari R 15](#_Toc78710650)

[Fungsi Bantuan Lain 15](#_Toc78710651)

[Test Formatif 16](#_Toc78710652)

[2. Module 2 : Mekanisme Kerja, Pembuatan Objek, dan Informasi Bantuan dalam Sistem R 17](#_Toc78710653)

[K. Belajar 1 Mekanisme Kerja Sistem R 17](#_Toc78710654)

[K. Belajar 2 Pembuatan, Penayangan, serta Penghapusan Objek dalam R dan Informasi Bantuan Online. 21](#_Toc78710655)

[3. Modul 3 : Data dalam Sistem R 29](#_Toc78710656)

[K. Belajar 1 Jenis-Jenis Objek dan Operasi Dasar Aritmatika dalam R 30](#_Toc78710657)

[K. Belajar 2 Membaca dan Menulis *File* Data dalam R 35](#_Toc78710658)

[Cara Mengakses Nilai dari Data Frame 36](#_Toc78710659)

[Argumen atau Opsi read.table() 36](#_Toc78710660)

[Penggunaan Memoi 52](#_Toc78710661)

[4. Lampiran 53](#_Toc78710662)

# Modul 1 : Pengenalan dan Instalasi Perangkat Lunak dan Lingkungan Pemrograman R

### Pendahuluan

Modul 1 ini akan memperkenalkan anda sekalian pada perangkat lunak dan lingkungan pemrograman **R**. Lingkungan pemrograman **R** ini relatif baru dan belum banyak dikenal oleh kalangan di luar perguruan tinggi. Modul 1 ini akan menjelaskan langkah-langkah penginstalasian perangkat lunak dan lingkungan pemrograman **R**. Bagian akhir modul 1 ini akan dibahas dasar-dasar pengoprasian lunak dan pemrograman **R**.

Dalam modul 1, dipaparkan pembahasan mengenai pengenalan dan instalasi prangkat lunak dan lingkungan dalam Pemrograman **R**. Setelah mempelajar modul ini, diharapkan anda dapat;

1. memahami perangkat lunak bahasa R.
2. memahami lingkungan pemrograman bahasa R.

### Perangkat lunak & Lingkungan Pemrograman R

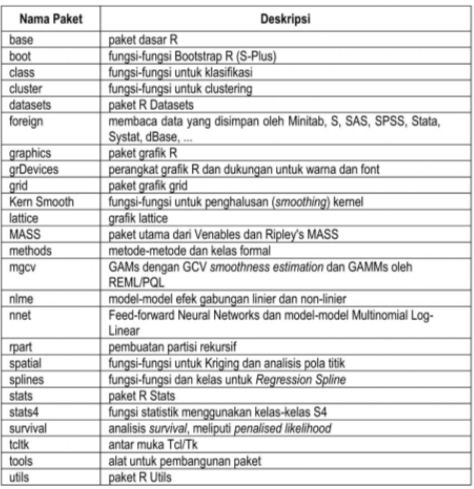
R adalah suatu perangkat lunak untuk membantu melakukan analisis stastik, perhitungan matriks, dan pembuatan grafik. R dikembangkan oleh Ross Ihaka & Robert Gentleman. R adalah seuatu perangkat lunak yang termasuk dalam kategori *Open Source*.

R digunakan untuk melakukan perhitungan dan manipulasi data secara statistik serta untuk menampilkan data dalam bentuk grafis. R memiliki beberapa fasilitas, yakni;

1. Pengelilaan dan peyimpanan data.
2. Kumpulan operasi untuk perhitungan bentuk array khususnya matriks
3. Kumpulan fasilitas pengolahan data yang cukup besar dan komprehensif.
4. Sarana peembuatan grafis dan penampulan untuk penyajian pada layar monitor ataupun pada kertas, juga dapat disimpan sebagai file dalam berbagai bentuk format.

R Dikembangkan oleh sebuah tim dan menerima kontribusi dari berbagai pihak dalam pengembangannya sehingga dari waktu ke waktu faslitas yang ada dalam R selalu bertambah banyak dan meningkat kualitasnya. Daftar lengkap paket tersebut dapat dilihat pada <https://cran.r-project.org/web/packages/available_packages_by_name.html> . Berikut adalah beberapa paket penglolaan, analisis, dan penampilan data yang terdapat pada R.

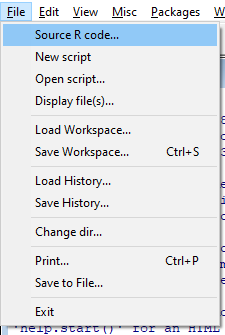
Tabel 1 Beberapa Paket dalam R



### Menjalankan R

Pada baris dibawah ini , tedapat delapan ikon alat (*icon tools*) dengan masing-masing memiliki fungsi yang telah ditentukan.

Gambar 1‑1 Menu File

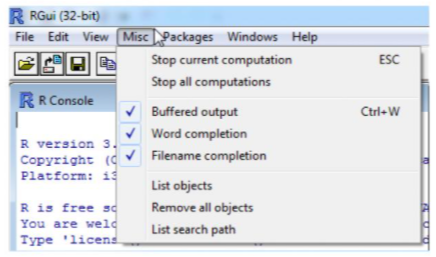


Fungsi-fungsi / sub menun dalam menu *file* sebagai berikut;

1. *Sour R code* berfungsi menjalankan program yang telah dibuat terlebih dahulu. Program disimpan dilam file dengan format R, Misalnya file pembayra.R
2. *New Script* berfungsi menulis program R baru atau *script*  yang baru.
3. *Open Script* berfungsi membuka program R yang sebelumnya telah ditulis dan disimpan.
4. *Display File*  berfungsi memperlihatkan daftar *file* yang ada didalam suatu *directory* atau *folder*.
5. *Load Workspace* berfungsi memasukan data dari tempat penyimpanan didalam *hard disk* kedalam memorit untuk diolah.
6. *Save Workspace* berfungsi meyimpan data yang ada didalam memori kedalam *hard disk* atau media penyimpanan lainnya.
7. *Change Dir* berfungsi mengganti direktori kerja. Sebaiknya dalam mengerjakan sebuah projek kita membuat sebuah folder yang terpisah dengan projek yang lainnya.
8. *Print* berfungsi mencetak apa saja yang tersimpan dalam *workspace/R console* (ruang kerja yang tampak dalam layar monitor) kedalam *printer*.
9. *Save to file* berfungsi menyimpan segala sesuatu yang ada didalam *workspace* atau *R Consile* kedalam suatu file teks.
10. *Exit* untuk mengakhiri atau menutup program R.

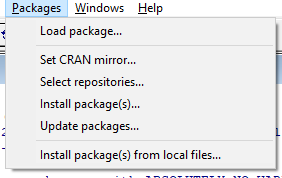
Untuk menu yang sejenis dengan aplkasi lain akan dilewatkan, dijelaskan dibawah ini fungsi dari menu ang sangat penting dan sering dipakai dalam menjalankan program R.

Gambar 1‑2. Menu Misc



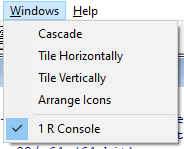
Pada menu *misc* (Gambar 1‑2. Menu Misc) terdapat fungsi *stop current computation* yang berfungsi untuk menghentikan perhitungan yang sedang belangsung, hal ini juga dapat digunakan dengan menekan tombol **ESC.** Fungsi lainnya adalah *List Objects* untuk menampilkan daftar objek dan *remove all objects* untuk menghapus semua objek.

Gambar 1‑3 Menu Package



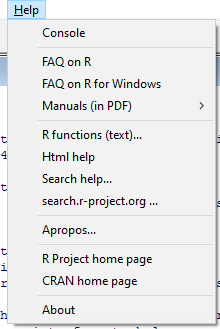
Menu *packages* berisi fungsi-fungsi untuk menambahakan paket statistik dan paket lainnya (*Load Package*), mengatur CRAN *mirror* (*Set CRAN Mirror*), memilih repositori tempat penyimpanan paket (*Select Repositories*), menginstal paket (Intall Package), dan memperbarui pakcage (*Upadte Packages*), dan kita dapat menginstall paket dari *file zip* yang disimpan dalam komputer kita (I*nstall package(s) from local fils*).

Gambar 1‑4 Menu Windows



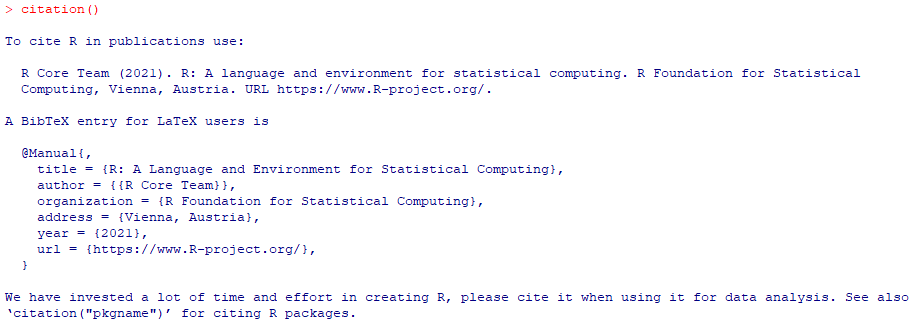
*Menu window* menyediakan pilihan-pilihan untuk tampilan jendela R *Console*. Yaitu, *Cascaed, Tile Horizontaly, Tile Vertically, dan Arrange Icons*.

Gambar 1‑5 Menu Help



Menu *help* menyediakan pertanyaan yang sering diajukan R (FAQ), panduan manual dalam bentuk PDF, bantuan pejelasan untuk fungsi-fungsi dalam R, tautan ke *website* R Project dan CRAN, serta mengenai versi R yang digunakan (*About*).

Gambar 1‑6 Cititaion Function



Walaupun Bahasa R adalah program open source yang dapat digunakan secara bebas, namun kode etis bagi seorang yang terpelajar diwajibkan untuk memberikan kutipan didalam tugas atau kerjaan. Informasi yang dikutip. Fungsi citation() (Gambar 1‑6 Cititaion Function) akan memberikan informasi yang dapat dimasukan pada kutipan didalam dokumen kerja.

### Fasilitas Help

Untuk mengetahui deskripsi, cara penggunaan dari sebuah fungsi yang sudah diketahui nama fungsinya dapat mengetik “?” setelah tanda promt diikuti dengan nama fungsi, > ? “read.tabel.

Gunakan fungsi help.search(“nama fungsi yang dicari”) untuk mengetahui fungsi yang diinginkan namun yang diketahui hanya subjek yang ingin dicari.

Fungsi lain yang berguna adalah dan . Fungsi digunakan untuk mencari data dari suatu nama tertentu.

> find ("lowess")

[1] "package:stats"

Sementara itu, berguna untuk mencari kata atau string yang berkaitan dengan fungsi yang ingin kita cari tahu maksudnya (secara menyeluruh atau parsial). Perhatikan contoh berikut

> apropos ("lm")

[1] ".colMeans" ".lm.fit" "colMeans" "confint.lm" "contr.helmert" "dummy.coef.lm" "glm"

[8] "glm.control" "glm.fit" "KalmanForecast" "KalmanLike" "KalmanRun" "KalmanSmooth" "kappa.lm"

[15] "lm" "lm.fit" "lm.influence" "lm.wfit" "model.matrix.lm" "nlm" "nlminb"

[22] "predict.glm" "predict.lm" "residuals.glm" "residuals.lm" "summary.glm" "summary.lm"

### Bantuan Online Help

Informasi yang sangat banyak mengenai R anda dapat peroleh pada situr CARN <https://cran.r-project.org/> Dalam situs tersebut, Anda akan menjumpai beberapa manual penggunaan R seperti berikut.

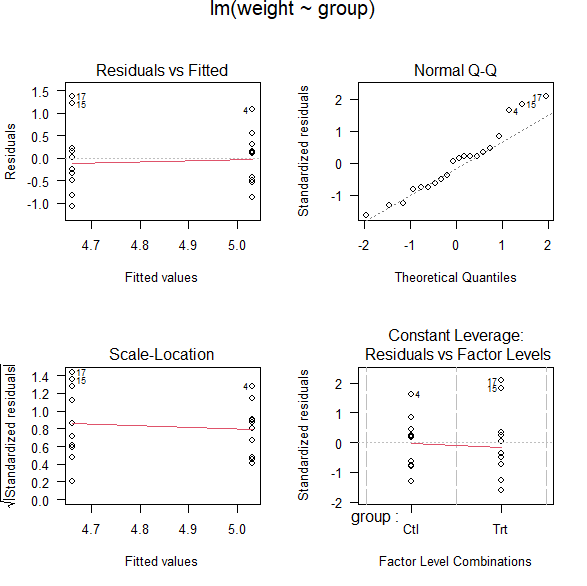
1. *An Indroduction on R*
2. *R data import/export*
3. *R installation and Administration*

Untuk mencari contoh penulisan suatu fungsi dan contoh *output*-nya, kita gunakan perintah sebagai contoh fungsi ,

> example (lm)

# Output

Gambar 1‑7 Hasil perintah > example (lm)



Hasil perintah pada (Gambar 1‑7) adalah proses visualisasi dari beberapa fungsi yang digunakanm fungsi yang digunakan dan ditulis dapat dilihat pada (Lampiran 4‑1 Kode hasil fungs >example (lm)).

### Demo pada R *Functions*

Perintah ini untuk melihat jangkauan pekerjaan yang dapat dilakukan oleh bahasa R. Berikut ini adalah beberapa demo yang bisa anda coba.

> demo (persp)

> demo (graphics)

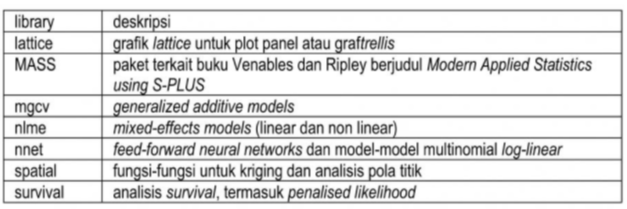
> demo (Hershey)

> demo (plotmath)

### *Libraries* dalam bahasa R

Secara sederhana, ketikan salah satu fungsi di dalam tanda kurung. Apa bila kita ingin menggunakan misalnya

Tabel 2 Beberapa Library dalam R



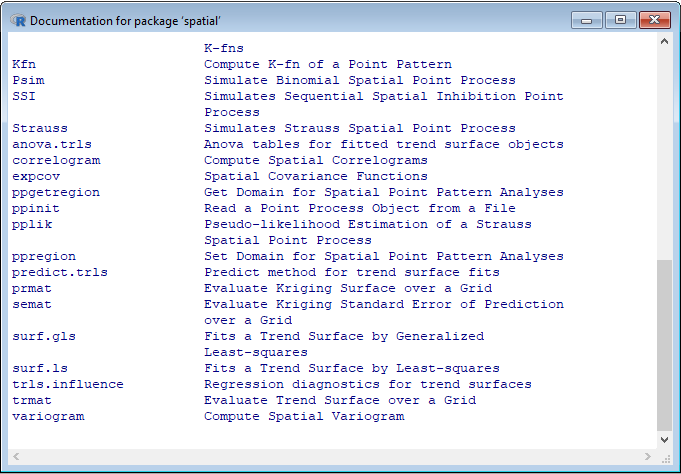
#### Isi dari Library

Sesuatu yang mudah menggunakan fungsi dalam mencari isi *library packages*. Sebagai contoh, perintah berikut untuk mencari informasi dari *spatial library*.

> library (help=spatial)

Maka itu tampil pada jendela pada (Gambar 1‑8 Hasil dari library(help=spatial))

Gambar 1‑8 Hasil dari library(help=spatial)



### Instalasi Packages

*Pakcages* dasar tidak berisi beberapa *library* yang dirujuk dalam buku ini, tetapi untuk mendownload *packages* ini sangatlah mudah. Gunakan fungsi install.packages untuk *download library* yang anda inginkan. Untuk mempercepat donwload, pilihlah *mirror* yang paling dekat dengan anda (Indonesia). Selanjutnya proses akan berjalan secara otomatis.

## Paramter 1 : nama packages yang ingin disinttal

## Parameter 2 : Mirror, atau sumber donwload package, pilih yang terdekat agar download lebih cepat.

install.packages('RMySQL', repos='https://repo.bppt.go.id/cran/)

Downloaded pakcages are stored in C:\Users\farras\AppData\Local\Temp\RtmpeMvcGy\downloaded\_packages

*Packages* yang digunakan dalam buku ini sebagai berikut

> install.packages ("akima", repos='https://repo.bppt.go.id/cran/)

> install.packages ("akima", repos='https://repo.bppt.go.id/cran/)

> install.packages ("akima", repos='https://repo.bppt.go.id/cran/)

> install.packages ("akima", repos='https://repo.bppt.go.id/cran/)

> install.packages ("akima", repos='https://repo.bppt.go.id/cran/)

> install.packages ("akima", repos='https://repo.bppt.go.id/cran/)

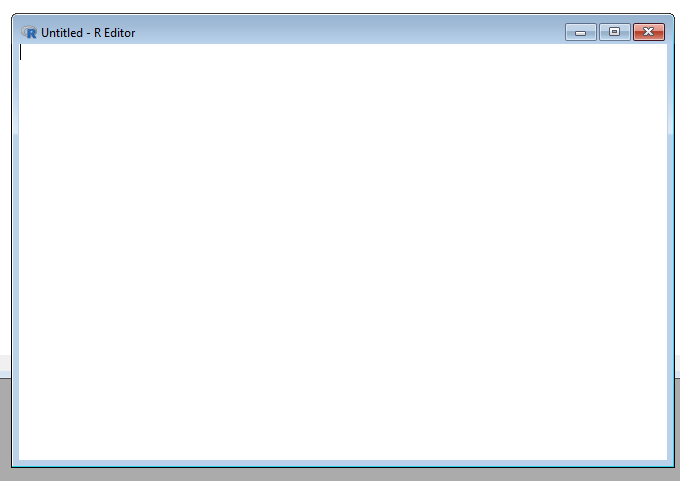
> install.packages ("akima", repos='https://repo.bppt.go.id/cran/)

> install.packages ("akima", repos='https://repo.bppt.go.id/cran/)

### Command Line Versus Script

Text editor didalam R adalah RGui, Untuk keperluan ini kita tinggal klik menu *file*. Pilihlah menu *new script*. Kemudian akan muncul jendela *untitled-R editor* (Gambar 1‑9 UntitledR Editor).

Gambar 1‑9 UntitledR Editor

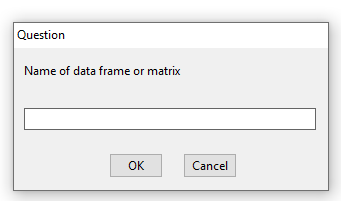


Dengan fasilitas ini, anda dapat menulis atau mengedit perintah. Untuk menjalan perintah yang ada didalam R Edtior tersebut dapat menekan **CTRL-R.** Baris perintah yang sudah diketik selanjutnya secara otomatis akan ditransfer ke *window* R *console* beserta hasil eksekusinya. Dengan **CTRL-S** anda dapat meyimpan peintah yang ditulis (*Script*) dalam jendela *untitled-R Editor*.

### Data Editor

*Data Editor* didalam R dapat diakses melalui *menu edit*, kemudian pilih *data editor*. Selanjutnya anda perlu memasukan nama *data frame* atau matriks (*data frame* disini adalah *data frame* yang aktif dalam R). *Data Frame* digunakan untuk menyimpan tabel-tabel data. *Data Frame* berisi vektor-vektor dengan panjang yang sama.

Gambar 1‑10 Jendela isian data frame atau matriks

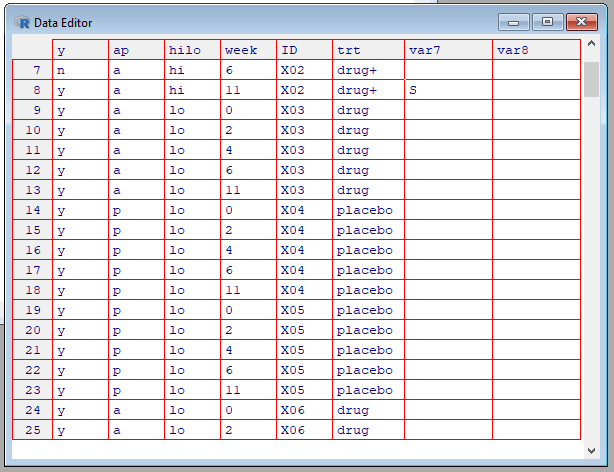


Setelah anda memasukan nama *dataframe* dalam kota *question* (Gambar 1‑10 Jendela isian data frame atau matriks), akan muncul jendela editor. Sebagai contoh, anda masukan salah satu nama *dataframe* yang terdapat dalam R *session* (*Bacteria* misalnya) sehingga akan muncul tampilan gambar (Gambar 1‑11 Jedela Data Editor untuk Data Frame Bacteria), *Data frame bacteria* terdapat dalam *package* MASS. Agar anda dapat menggunakan *data frame* tersebut, terlebih dahulu ketikan perintah berikut.

> library(MASS)

> attach(bacteria)

Gambar 1‑11 Jedela Data Editor untuk Data Frame Bacteria



Pembahasan mengenai *data frame* akan dibahas dalam pembahasan selanjutnya dalam BMP ini.

Cara lain untuk menampilkan dan mengedit *data frame*, yaitu menggunakan fungsi *fix*. Misalnnya anda ingin mengedit *data frame bacteria*. Setelah perintah diatas, dapa Anda ketikan perintah berikut ini.

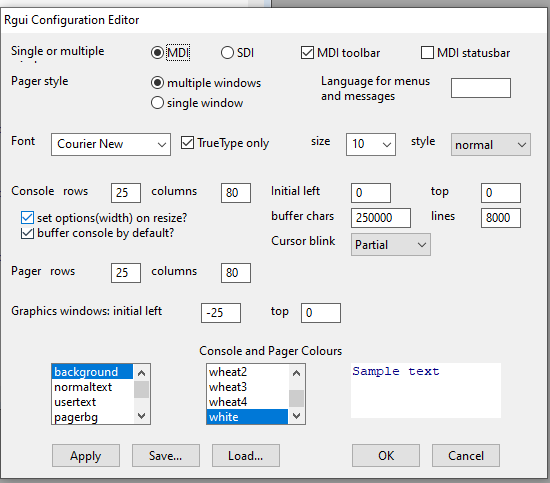
> fix(bacteria)

Bentuk tampilan dari RGui (*Data Editor*) tampak seperti tampilan *spreadsheet* Excel. Anda dapat mengubah judul dari kolum ataupun isi dari sel.

### Perubahan Tampilan Layar Dari R

Pengaturan *default* yang telah ada didama *window*  R saat ini seperti telah standar dan memenuhi apa yang diinginkan. Namun, apabila ingin merubah tampilan *window* ini, Anda dapat masuk ke R *gui configuration editor* terdapat dalam menu *edit/GUI preferences* (Gambar 1‑12 Jendela GUI Configuration Editor)

Gambar 1‑12 Jendela GUI Configuration Editor



### Fungsi Bantuan Lain

Untuk melihat melihat variabel apa yang telah Anda buat dalam *session* yang tengah berlangsung. Ketikan perintah berikut.

> objects()

Untuk meilhat *library* dan *dataframe* yang telah terlampir, ketikan perintah berikut

> search()

Sementeara itu, untuk menghapus variabel yang telah anda buat, digunakan perintah *rm*. Misalnya, dalam *session* sebelumya, anda telah membuat variabel dengan nilai , untuk menghapus vairbale , digunakan fungsi berikut.

# Mengisi variabel x

> x= x \*123

# Menampilkan variabel x

> print (x)

[1] 2952

# Menghapus variabel x

> rm (x)

> print (x)

Error in print(x) : object 'x' not found

Perintah untuk menghapus *data frame* yang telah dilampirkan dapat menggunakan perintah fungsi *detach* seperti berikut

> detach (bacteria)

### Test Formatif

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Jawaban | Hasil | No | Jawaban | Hasil |
| 1 | B |  | 6 | A |  |
| 2 | B |  | 7 | C |  |
| 3 | C |  | 8 | C |  |
| 4 | A |  | 9 | ~~B~~ (D) |  |
| 5 | ~~B~~ (D) |  | 10 | A |  |

# Module 2 : Mekanisme Kerja, Pembuatan Objek, dan Informasi Bantuan dalam Sistem R

## Mekanisme Kerja Sistem R

Sebagaimana yang dijelaskan didepan bahwa R adalah perangkat lunak dan lingkungan pemrogram. R adalah bahasa yang langsung diterapkan oleh sistemnya, hal ini berbeda dengan bahasa pemrogram lainya, seperti C, Fortran, dan Pascal yang harus dikompilasi terlebuh dahulu untuk membangun suatu program yang lengkap. Dalam R, setiap perintah yang dituliskan langsung dijalankan oleh program R dan dapat langsung dilihat hasilnya.

Setiap fungsi didalam R, selalu ditulis dengan sepasang tanda kurung dibelakangnya walaupun fungsi tersebut tidak membutuhkan argumen yang harus ditulis diantara tanda kurung. Contoh :

> ls()

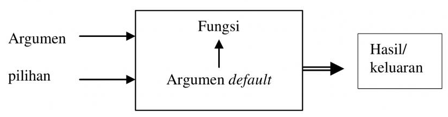
[1] "bacteria" "variane"

Penulisan fungsi dengan menggunakan tanda kurung dibelakang tujuannya untuk membedakan fungsi dari objek-objek lain di dalam R.

Ketika R sedang berfungsi, semua variabel, data, dan fungsi disimpan dalam bentuk objek di dalam *Random Access Memory* (RAM) komputer. Dalam beberapa literatur R, tempat pelaksanaan oeprasi disebu memori aktif. Anda dapat melakukan sesuatu pada objek-objek tersebut dengan operator, misalnya aritmatika, logika, perbandingan, serta fungsi.

Penggunaan fungsi sangat intuitif sebagaimana yang dijelaskan dalam bagan berikut (Gambar 2‑1 Bagan Penggunaan Fungsi R)

Gambar 2‑1 Bagan Penggunaan Fungsi R

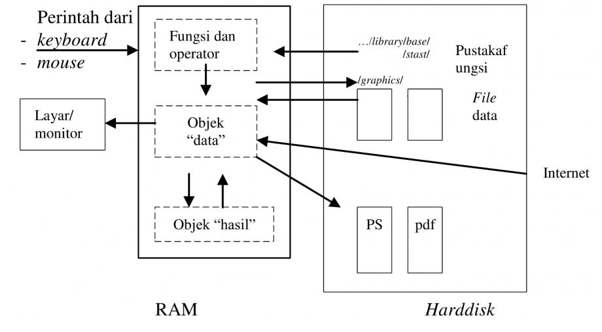


Nillai argumen *default* adalah nilai dari argumen yang telah ditentukan sebelumnya. Argumen dapat berupa objek (data, rumus atau ekspresi lainnya). Beberapa diantaranya mungkin telah ditentukan sebagai nilai *default*-nya. Anda dapat juga merubah nilai *default* tersebut dengan menggunakan pilihan atau *option*. Fungsi-fungsi dalam R ada yang membutuhkan argumen, termasuk argumen *default*, tetapi ada juga yang tidak membutuhkan argumen sama sekali.

Sebagaimana yang telah dijelaskan sebelumnya, bahwa semua objek yang akan dioperasikan oleh R akan disimpan didalam RAM. RAM adalah jenis memori yang apabila mesin dimatikan isi dari RAM tersebut akan hilang.

Apakah setiap kali mengelola data dengan R harus memasukan kembali ke RAM dengena mengetikanya ? Jawabanya tidak selalu demikian karena perangkat lunak R telah melengkapi dirinya dengan sarana pembacaan (*Reading*) dan penulisan (*writing*) data yang diperlukan dala operasi dengan R. Sarana pembacaan dipakai untuk menyalin data yang disimpan terlebih dahulu dalam media lain, seperti *hard disk ,* atau *flash disk*. Bahkan masukan data untuk R dapat dilakukan melalui internet. Sementara itu, faslitas penulisan digunakan untuk menyimpan hasil operasi R kedalam media lain. Gambar berikut menjelaskan mekanisme pemidahan data dari *hard disk* ke RAM dan sebaliknya.

Gambar 2‑2 Mekanisme Pemindahan Data dari Hardisk ke RAm dan Sebaliknya



Dari Gambar 2‑2 terlihat bahwa hasil analisi dari sistem R dapat disimpan dalam format *Post Script* (PS), *Portable Data Format* (PDF), dan format lainnya seperti jpg, png, bmp, emf, pictex, serta xfig.

Fungsi-fungsi yang tersedia didalam sistem R disimpan dalam direktori R-4.1.0/library. R-4.1.0 adalah direktori ketika kita menginstal R , dalam hal ini, R yang terinstal adalah versi 4.1.0. Direktori ini merupakan tempat penyimpanan paket-paket fungsi secara terstruktur. Paket yang bernama *base* adalah sistem inti R dan berisi fungsi-fungsi utama sistem R, khususnya untuk pembacaan dan perhitungan serta analisis data. Setiap paket disimpan dalam direktori dan diberi nama yang merupakan gabungan dari R dan nama paket itu sendiri, misalnya, R\R-4.1.0\library\base\R\base.

Beberapa contoh perintah yang relevan disini diantara lain adalah

# Mendapatkan informasi tempat kerja yang sedang berlangsung

> getwd()

[1] "C:/Users/farras/Documents"

# setwd() untuk mengganti tempat kerja yang sedang berlasung.

# argumen pertama : nama direktori tujuan

> setwd ("D:/10. KULIAH/02. SEMESTER 2/01. KOMPUTER 1/Belajar R")

# dir() untuk menayangkan isi dari direktori kerja

> dir()

Contoh salah satu perintah yang paling sederhana adalah menuliskan nama objek untuk menampilkan isi dari suatu objek. Misalnya, suatu objek myVariable yang mempunyani nilai 15 seperti dibawah ini.

> myVariable <- 15

> myVariable

[1] 15

Angka 1 dalam siku kurung menunjukan bahwa penayangan nilai myVariable dimulai dari nilai yang pertama. Perintah tersebut adalah pemakaian perintah print(myVariable) secara implisit. Itu artinya perintah print(myVariable) akan menghasilkan nilai yang sama dengan menuliskan myVariable saja.

Namun demikian, dalam beberap situasi, misalnya dalam fungsi atau *loop*, perintah print() harus dituliskan secara eksplisit dan lengkap. Sebagai contoh, perintah berikut ini mencetak angka 1 sampai dengan 5 menggunakan pernyataan for untuk melakukan perulangann (*loop*).

> for (i in 1:5) { print (i)}

[1] 1

[1] 2

[1] 3

[1] 4

[1] 5

Penamaan suatu *object* bersifat *case sensitive*. Itu artinya huruf kecil dan kapital dianggap berbeda sehingga X dan x merupakan objek yang berbeda. Aturan penamaan objek dalam R adalah.

1. Diawali dengan huruf, baiku huruf kapital maupun huruf kecil.
2. Diikuti oleh huruf, angka, titik, dan garis bawah.

## Pembuatan, Penayangan, serta Penghapusan Objek dalam R dan Informasi Bantuan Online.

Suatu objek dapat dibuat dengan operator yang digambarkan dengan anak panah. Anak panah ini bisa mengarah ke kanan dan ke kiri. Variabel yang nilainya ditetapkan diletakan di depan anak panah.

> # invalid

> m ->5

Error in 5 <- m : invalid (do\_set) left-hand side to assignment

> # valid

> m <-5

> 24 -> n

Nilai dari variabel m juga dapat merupakan hasil perhitungan dari beberapa angkat seperti berikut.

> m <- ((2+3)/5)+9

> m

[1] 10

Nilai suatu variabel juga dapat diperoleh dari nilai variabel lainnya. Sebagai contoh, perintah berikut mengisi variabel n dengan nilai dari variabel m, sehingga nilai m sama dengan nilain n.

> m -> n

> n

[1] 10

Disamping itu, nilai suatu variabel juga dapat merupakan hasil perhitungan dari satu atau lebih variabel menggunakan operator aritmatika dasar +, -, \*, /, dan ^ . Sebagai contoh, nilai variabel y berikut adalah hasil perhitungan persamaan , yaitu nilai terlebih dahulu diisi nilai 15.

> 15 -> x

> y <- x^2 + 5\*x - 12

> y

[1] 288

Nilai suatu variabel juga dapat ditentukan menggunakan fungsi yang tersedia dalam R. Sebagai contoh, variabel y diberi nilai , nilai

> x <- 144

> y = sqrt(x)+(sqrt(x)/2)+10

> y

[1] 28

Dalam perintah diatas dinyatakan sebagai sqrt(x), yaitu sqrt() merupakan fungsi aritmatika yang tersedia dalam R. Fungsi aritmatika lainya yang tersedia didalam R, diantaranya, *log*, *exp*, *sin*, *cos*, dan *tan*.

R beroperasi tidak hanya pada variabel yang bernilai konstanta, tetapi juga pada struktur data yang bernama vektor numerik.

Vektor numerik adalah entitas tunggal yang terdiri atas kumpulan terurut bilangan-bilangan.

Sebagai contoh, untuk membuat vektor yang diberi nama yang berisi lima bilangan 10, 6, 2, 41,dan 2, digunakan perintah R berikut.

> x <- c(10,6,2,41,2)

> x

[1] 10 6 2 41 2

Pernyataan penugasan x <- (10,6,2,41,2) menggunakan fungsi c() untuk merangkai nilai-nilai vektor.

Kita dapat membuat vektor-vektor numerik lainnya dari vektor numerik yang telah ada dengan menggunakan pernyataan aritmatika yang melibatkan vektor yang telah ada, konstanta, dan operator-operator arimatika. Sebagai contoh, vektor y diperoleh dair vektor x dengan mengalikan setiap elemen dalam vektor x dengan 2.

> y <- x\*2

> y

[1] 20 12 4 82 4

Contoh lain pernyataan aritmatika menggunakan vektor sebagai barikut.

> n <- x^2+2\*x+1

> n

[1] 121 49 9 1764 9

Kita dapat menerapkan fungsi-fungsi yang tersedia didalam R terhadap vektor numerik.

> x <- c(1,4,6,2,8,7,9,1,10,3)

> length(x)

[1] 10

> min(x)

[1] 1

> max(x)

[1] 10

> sum(x)

[1] 51

> mean (x)

[1] 5.1

> sort (x)

[1] 1 1 2 3 4 6 7 8 9 10

Dengan menggunakan fungsi *sum(), mean(),* dan *length(),* kita juga dapat menghitung nilai *vairance* dari vektor *x* sebagai berikut. Namun R sendiri memiliki fungsi *var()* untuk menghitung nilai *variance*.

> (sum((x-mean(x))^2))/(length(x)-1)

[1] 11.21111

> var (x)

[1] 11.21111

Kita juga dapat mengetahui nama-nama variabel dan vektor yang telah kita buat menggunakan fungsi ls(), ls(pat = “...”), atau ls(pat=”^...”). ls merupakan kependekan dari *list simply*. Apabila anda mempunyai banya objek, sebaiknya anda menamai objek seinformatif meungkin sehingga anda bisa menggunakan opsi *patern* atau pola yang dapat disingkat dengan *pat* untuk menayangkan nama-nama objek yang mengandung karakter atau huruf tertentu saja.

> ls (pat = "va")

[1] "variane"

> ls (pat = "ia")

[1] "bacteria" "myVariabel" "myVariable" "variane"

# tanda ^ untuk mencari objek yang diawali dengan huruf m.

> ls (pat = "^m")

[1] "m" "myVariabel" "myVariable"

Perintah ls.str() dapat dipakai untuk menayangkan infomasi objek secara lebih perinci(misalnya tipe dan nilai),

> ls.str()

bacteria : function ()

i : int 10

m : num 10

myVariabel : num 15

myVariable : num 15

n : num [1:5] 121 49 9 1764 9

variane : num 24

x : num [1:10] 1 4 6 2 8 7 9 1 10 3

y : num [1:5] 20 12 4 82 4

# dengan pola / *pattern*

> ls.str(pat = "^m")

m : num 10

myVariabel : num 15

myVariable : num 15

Untuk menghapus objek dalam memori, gunakanlah fungsi atau perintah rm(), rm merupakan kependekan dari *remove*.

|  |  |
| --- | --- |
| Pernyataan | Makna |
| rm(x) | Menghapus variabel |
| rm(x,y) | Menghapus variabel |
| rm (list=ls()) | Menghapus semua objek |
| rm (list = ls (pat = “^m”)) | Menghapus semua objek yang namanya diawali dengan huruf “m” |

> ls()

[1] "bacteria" "i" "m" "myVariabel" "myVariable"

[6] "n" "variane" "x" "y"

> rm (list = ls (pat = "^m"))

> ls()

[1] "bacteria" "i" "n" "variane" "x" "y"

# Objek myVariabel, myVariable, dan m dihapus.

Bantuan *online* dalam R akan sangant membantu dalam menggunakan R karena bantuan tersebut memberikan informasi lengkap.

> ?ls

starting httpd help server ... done

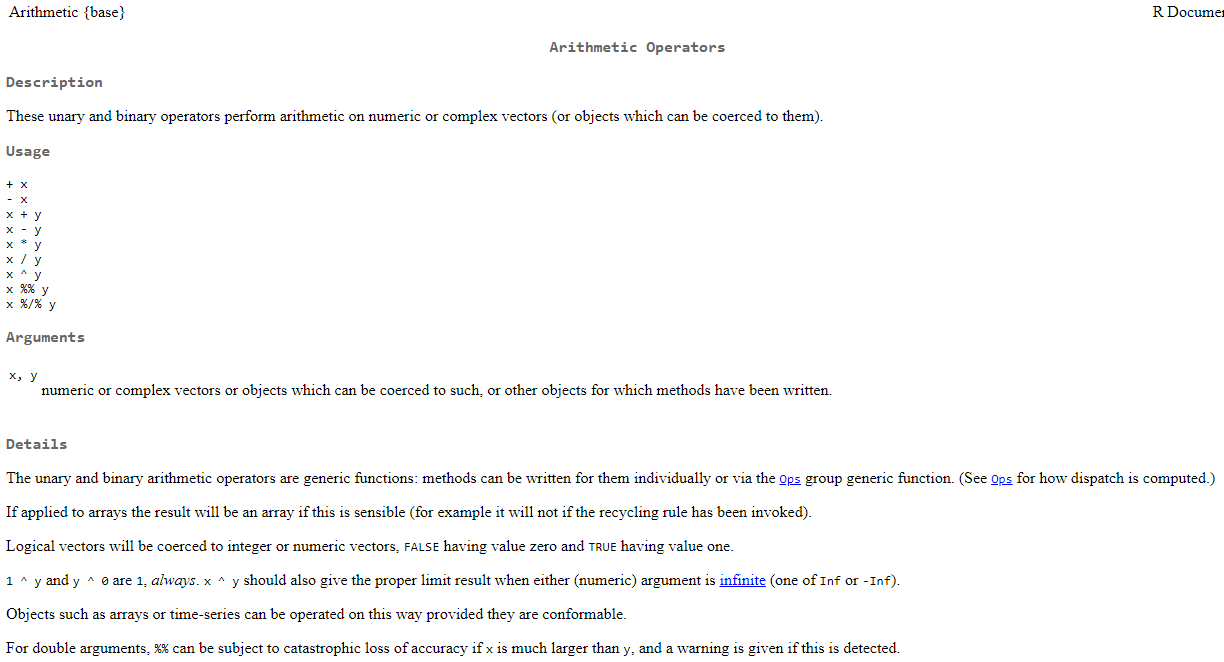
> help (ls)

> help ("ls")

perintah diatas akan menayangkan informasi bantuan yang berkenaan dengan fungsi ls().

> help ("\*")

Gambar 2‑3 Tampilan yang Dihasilkan dari Perintah help("\*")



Perintah *help* hanya akan menampilkan sistem R yang telah dimasukan dalam R atau memori aktif karena nilai yang telah ditetapkan sebelumnya untuk argumen fungsi *help*, yaitu try.all.packages, adalah *false*. Untuk melihat semua bantuan, argumen tersebut harus bernilai benar. Untuk melihat nilai argumen try.all.packages sebelum dan sesudah diubah perhatikan contoh berkut.

> help ("bs")

No documentation for ‘bs’ in specified packages and libraries:

you could try ‘??bs’

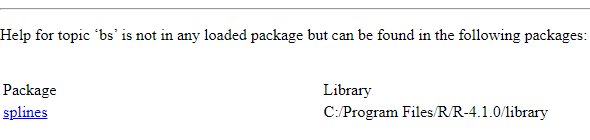
Pesan yang ditampilkan diatas menyatakan bahwa tidak ada dokumentasi dari “bs” dalam kumpulan paket dan pustaka yang telah dikumpulkan. Selain itu disarakan untuk menggunakan perintah ??bs.

Cara menggunakan try.all.packages sebagai berikut.

> help ("bs", try.all.packages = TRUE)

Perintah diatas akan menghasilkan pesan berikut yang tampil pada halaman *web* pada *browser*.

Gambar 2‑4 Hasil dari help dengna argumen try.all.pacages=true



Dalam kasus ini, informasi bantuan tidak ditayangkan secara langsung. Namun anda mendapatkan informasi bahwa informasi bantuan tersebut dapat ditemukan. Berdasarkan informasi diatas (Gambar 2‑4), anda dapat menggunakan perintah berikut.

> help(package="splines")

Gambar 2‑5 Tampilan yang DIhasilkan dari perintah help(package=”splines")



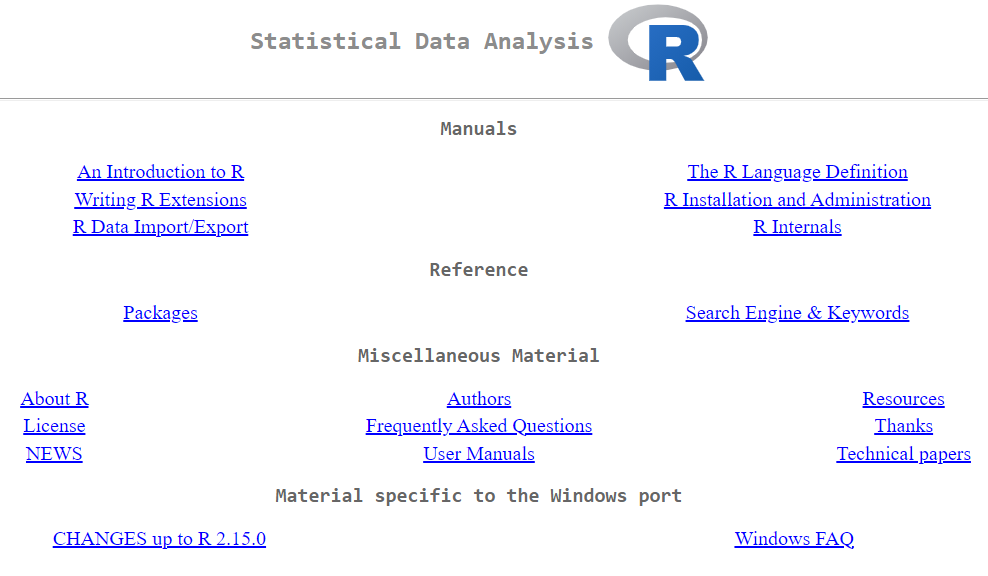
Anda juga dapat menampilkan informasi bantuan dalam format HTML yang dipakai untuk halaman web di Internet dengan perintah berikut. dan akan muncul seperti (Gambar 2‑6)

> help.start ()

If nothing happens, you should open

‘http://127.0.0.1:27911/doc/html/index.html’ yourself

Gambar 2‑6 Hasil dari help.start()



Pencarian informasi dengan memakai kata kunci dimungkinkan pada perintah *help* dengan memakai format html. peintah yang digukan adalah help.search().

> help.search("tree")

Pencarian informasi bantuan tersebut akan dilakukan pada semua paket yang terinstal. Oleh karena itu, Apabila anda baru saja melakkan instalasi paket baru, Anda perlu menggunakan opsi *rebuild* dan mengubah nilai dari opsi menjadi *true*.

> help.search("tree",rebuild=TRUE)

Perintah atau fungsi apropos() dipakai untuk menemukan informasi bantuan mengenai fungsi yang terdapat dalam argumen pemanggilan fungsi tersebut. Dalam hal ini, pencarian hanya dilakukan pada memori yang sedang aktif (RAM). Berikut adalah contoh untuk menampilkan informasi bantuan yang mengandung “help”.

> apropos ("help")

[1] "help" "help.request" "help.search" "help.start"

# Modul 3 : Data dalam Sistem R

Pembahasan pada modul ini meliputi jenis objek beserta karakteristiknya, pembacaan dan penyimpanan data dari *file* komputer. serta pembuatan data dan manipulasi data dengan berbagai operasi dan fungsi.

Capaian :

* Mengetahui Jenis Objek dan karateristiknya dalam R
* Mampu membaca data dari *file* yang sudah tersedia di media penyimpanan.
* Menulis objek dalam bentuk *file* di media penyimpanan.
* Membangun data dengan berbagai fungsi dasar dalam R.
* Membuat dan memanipulasi data dengan berbagai operasi aritmatika dan fungsi sederhana.

## Jenis-Jenis Objek dan Operasi Dasar Aritmatika dalam R

Setiap objek yang dipakai selalu memiliki nama. Nilai serta atribut yang menentukan jenis data dinyatakan oleh seuatu objek. Dalam pengolahan data suatu variabel yang bernilai 1, 2 dan 3, belum tentu variabel memiliki satu jenis data saja. Variabel tersebut dapat berjenis bilangan riil atau bilangan bulat, seperti jumlah siswa, atau bisa juga variabel kategorikal, gabungan antara beberapa jenis data seperti kode unit pelaksanaan belajar jarak jauh (UPBJJ) yang dipakai di lingkungan Universitas Terbuka.

Operasi penglohan data tergantung dari tipe data yang tersedia. Jika salah dalam pengolahan data maka hasilnya akan sia-sia.

Dengan memahami jenis datanya, tentu konsekuensi operasi dalam pengolahan data atau analisis statistiknya menjadi berbeda. Anda tidak bisa menjumlahkan angka-angka kode UPBJJ untuk mendapatkan nilai rata-rata kode UPBJJ diantara 100 mahasiswa yang ada dalam penelitian. Walaupun secara manual dapat saja mendapatkan nilai rata-rata tersebut dengan mejumlahkan kode UPBJJ dan dibagi dengan jumlah mahasiswa; tindakan tersebut tidak bermakna sama sekali. Hal ini disebabkan angka-angka tersebut hanyalah kode yang mewakili nama-nama UPBJJ.

Semua objek dalam R memilki dua atribut instrinsik, diantaranya adalah;

1. Mode

Jenis atau tipe suatu objek. Mode dari suatu objek dapat berupa Tabel 3 Jenis Mode (Tipe-Tipe).

Untuk melihat jenis dari suatu objek dapat menggunakan fungsi mode atau class. Gunakan fungsi in.{tipe objek}(value) akan mengembalikan nilai TRUE jika benar dan sebaliknya.

Tabel 3 Jenis Mode (Tipe-Tipe)

| Jenis | Dalam R mode (penulisan) | Contoh |
| --- | --- | --- |
| Numerik | **num** | > mode (27)  [1] "numeric"  > is.numeric (27.0)  [1] TRUE  # 27 tetap dianggap sebagai numerik oleh R. Salah satu kekurangan dari R.  > is.integer (27)  [1] FALSE |
| Karakter | **char** | > mode ("Hai")  [1] "character"  # “27” akan menjadi karakter, bukan numerik  > mode ("27")  [1] "character"  > is.character ("27")  [1] TRUE |
| Complex | **com** | > mode (4i)  [1] "complex"  > is.complex (4i)  [1] TRUE |
| Logic | **log** | > mode (true)  Error in mode(true) : object 'true' not found  # Penulisan tipe objek logikal harus huruf kapital  > mode (TRUE)  [1] "logical"  > is.logical (TRUE)  [1] TRUE |

1. Panjang

Panjang dari suatu objek dapat diketahui dengan menggunakan fungsi length() dengan parameter sebuah nilai atau variabel

> length (24)

[1] 1

> length (24)

[1] 1

> x <- c (2,5,2,1,5)

> length (x)

[1] 5

Bilangan yang sangat besar dan sangat kecil dapat dinyatakan secara eksponensial. Misalnya, 17.000.000 dapat dinyatakan dengan 17e6

> n = 17000000

> m = 17e6

> n

[1] 1.7e+07

> m

[1] 1.7e+07

> n == m

[1] TRUE

Sistem R mempunyai cara yang standar atau mirip dengan sistem yang lain dalam menangani data yang hilang (N/A, *No Availabel*), bilangan tak hingga (Inf ), dan sesuatu yang seharusnya bilangan tapi bukan bilangan (NaN, *Not a Number*).

> x <- 5

> y = x/0

> y

[1] Inf

> y-y

[1] NaN

Nilai dari variabel dengan mode karakter dimasukan dalam tanda petik *double* atau *single*. Dimungkinkan pula untuk memasukan karakter dengan diikut dengan tanda \. Kombinasi tanda \ dengan “ akan diperlukan secara khusus oleh beberapa fungsi, misalnya cat yang berkaitan dengan penayanangan di layar atau write.table untuk menulis hasil kedalam *disk.*

> x <- "jum'at"

> x

[1] "jum'at"

> y <- 'jum'at'

Error: unexpected symbol in "y <- 'jum'at"

> y <- 'jum\'at'

> y

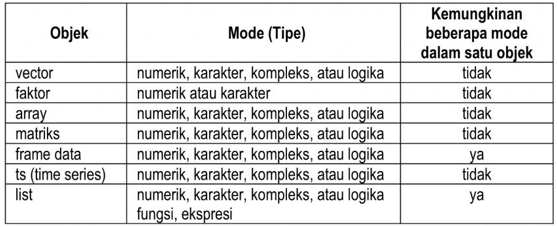
[1] "jum'at"

> cat ("hari ini hari ",y)

hari ini hari jum'at

Tabel 4 Tipe-tipe Objek untuk Penyimpanan Data berikut ini memberi gambaran berbagai jenis objek untuk meyimpan data.

Tabel 4 Tipe-tipe Objek untuk Penyimpanan Data



* Vektor adalah variabel dalam makna yang telah disepakati secara umum.
* Faktor adalah variabel kategorikal.
* Array adalah tabel berdimensi . ,
* Matriks adalah *array* berdimensi dua.
* Frame data adalah suatu tabel yang terdiri atas satu atau beberapa vektor yang sama panjangnya, tetapi dimungkinkan mempunyai mode yang berbeda.
* ts (*time series*) adalah data runtun waktu yang mengandung informasi tambahan, misalnya waktu, tanggal, dan frekuensi.
* List adalah objek yang dapat mengandung sembarang objek, termasuk list itu sendiri.

Dalam suatu vektor, mode dan panjanganya sudah cukup untuk menjelaskan datanya. Namun untuk objek lain, informasi lain masih diperlukan dan informasi ini diberikan oleh antribut nonintrisik, diantaranya atirbut dim yang mengacu pada dimensi dari objek. Sebagai contoh matriks dengan dua baris dan dua kolom.

## Membaca dan Menulis *File* Data dalam R

R menggunakan direktori tempat kerja atau *working directory*. Untuk mengetahui direktori tempat kerja ini digunakan perintah getwd(). Untuk mengubah atau pindah direktori digunakan perintah setwd(), sementara untuk mengetahui isi dari direktori, gunakan perintah dir().

R mampu membaca data dalam berbagai format, misalnya ASCII, Excel, SAS, dan SPSS. R dapat juga membaca data yang disimpan dalam basis data yang diakses dengan SQL (*Structured Query Language*). Namun *package* untuk membaca data dalam format tersebut tidak terdapat dalam *base pakcage*, kita harus mengnistall *library* tambahan kedalam sistem R.

Tabel 5 Cara Membaca File Dan Penjelasannya[[1]](#footnote-1)

|  |  |  |
| --- | --- | --- |
| Fungsi | Penggunaan | Penjelasan |
| getwd() | > getwd()  [1] "D:/direktori/lokasi/" | Menampilkan alamat dari *working directory* |
| setwd(Path) | > setwd("direkotri//direktori lain//tujan")  atau  > setwd("direkotri\direktori lain\tujan") | Berpindah working directory |
| dir() | > dir()  [1] "Data" "RScript"  # Nested, untuk melihat isi dari subdirektori  > dir(dir()[1])[1]  [1] "data\_saya.dat" | Untuk melihat isi dari direktori |
| read.table() | >myData<- read.table("data/data\_saya.dat") | Untuk membaca isi data\_saya.dat dan menyimpannya dalam *frame data* yang diberi nama myData. |
|  | > myData | Untuk menampilkan isi *framedata* |

### Cara Mengakses Nilai dari Data Frame

Untuk mengakses nilai dari *data frame* tersebut dapat dilakukan dengan cara sebagai berikut: myData$V1, myData$V2, dan myData$V3. Bisa juga dengan cara myData[“V1”], myData[“V2”], dan myData[“V3”]. Selain itu juga bisa dengan myData[,1], myData[,2], myData[,3]. Hasil lengkapnya dapat dilihat di Lampiran 4‑2 Berbagai Cara untuk Mengakses Nilai dari Data Frame

### Argumen atau Opsi read.table()

#### Deskripsi

Membaca *file* dengan format tabel dan menjadikannya sebuah *data frame*, *with cases coressponding to lines and variabel to fields in the file.*

#### Penggunaan

read.table(file, header = FALSE, sep = "", quote = "\"'",

dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),

row.names, col.names, as.is = !stringsAsFactors,

na.strings = "NA", colClasses = NA, nrows = -1,

skip = 0, check.names = TRUE, fill = !blank.lines.skip,

strip.white = FALSE, blank.lines.skip = TRUE,

comment.char = "#",

allowEscapes = FALSE, flush = FALSE,

stringsAsFactors = FALSE,

fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)

#### Argumen - File[[2]](#footnote-2)

*Bernilai String* : Alamat dan nama dari *file* yang ingin dibaca. Setiap *row* (baris) dari tabel ditampilkan sebagai satu garis di dalam *file*. Jika argumen ini tida berisian alamat absolut, maka nama *file* bisa berupa alamat relatif dari *working director* saat ini, getwd().

*file* juga dapat diambil dari URL.

# Alamat relatif

> dataMahasiswaRlf <- read.table ("Data/nilai\_mahasiswa.dat")

# Alamat absolut

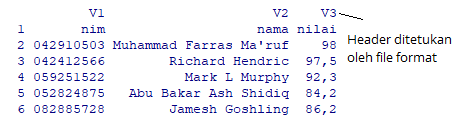
> dataMahasiswaAbs <- read.table ("D:/{alamat}/Data/nilai\_mahasiswa.dat")

#### Argumen - header[[3]](#footnote-3)

*Bernilai Logic* : TRUE atau FALSE. Nilai tersebut mengidentifikasi apakah *file* tersebut memilki nama variabel pada baris pertamanya. Jika tidak ada (header=FALSE) maka, nilai *header* ditentukan oleh *file* format. Seperti V1, V2. atau V3. *Default value* dari argumen ini adalah FALSE, jika TRUE maka baris pertama dieksekusi sebagai nilai *header*.

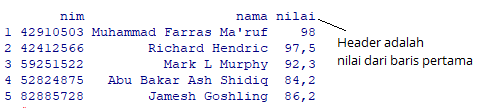
> dataMahasiswaNoHeader <- read.table ("Data/nilai\_mahasiswa.dat", header = FALSE)

Gambar 3‑1 Data Fram dengan Argumen header = FALSE



> dataMahasiswaWHeader <- read.table ("Data/nilai\_mahasiswa.dat", header = TRUE)

Gambar 3‑2 Data Frame dengan argumen header = TRUE



#### Argumen - sep[[4]](#footnote-4)

*Bernilai Character :* Pemisah kolom yang digunakan dalam *file* data. Nilai dari setiap baris pada sebuah *file* dipisahkan oleh *Character* ini. Jika jika sep = “” (Nilai bawaan) maka pemisahnya adalah ‘*white space*‘.

Jika pemisahnya adalah tab maka bisa gunaka nilai sep=”\t”. Jika koma bisa kita gunakan sep=”,”.

> path <- "Data/nilai\_mahasiswa\_separator.dat"

> dataMahasiswaSep <- read.table (path, header = TRUE,sep="|")

> dataMahasiswaSep

nim nama nilai

1 42910503 Muhammad Farras Ma'ruf 98

2 42412566 Richard Hendric 97,5

3 59251522 Mark L Murphy 92,3

4 52824875 Abu Bakar Ash Shidiq 84,2

5 82885728 Jamesh Goshling 86,2

#### Argumen – quote[[5]](#footnote-5)

*Bernilai Character* : Kumpulan dari *quoting character* sebagai karakter *string* tunggal.

Penggunaan quote biasa digunakan seperti dibawah ini, dimana sep adalah argumen seperator.

quote = if(identical(sep, "\n")) "" else "'\""

Agar lebih mudah dipahami, argument quote digunakan untuk memberitahu R jika ada sebuah nilai yang diawali dan diakhiri dengan *quoting character* seperti (‘ , /”, \*, &) maka anggap string di dalamnya sebagai sebuah *character string tunggal*.

Contoh, kita memliki data yang aneh seperti dibawah ini ;

nim nama &nama panggilan& nilai

042910503 'Muhammad Farras Maruf' 'faras maruf' 98

042412566 'Richard Hendric' 'richard' 97,5

059251522 \*Mark L Murphy\* 'Mark Murp' 92,3

052824875 'Abu Bakar Ash Shidiq' 'ABS' 84,2

082885728 “Jamesh Goshling” 'Jams' 86,2

Jika kita menjalankan perintah

> dataMahasiswaQuote <- read.table (path, header = TRUE)

Error in scan(file = file, what = what, sep = sep, quote = quote, dec = dec, :

line 1 did not have 6 elements

Maka akan muncul pesan *error* yang mengatakan bahwa baris pertama tidak memliki 6 elemen. Hal tersebut terjadi karena sebuah seluruh baris dari *file* tersebut tidak memiliki jumlah elemen yang konsisten. Dimana dijelaskan sebagai berikut;

* *Header* dari *file* tersebut (baris pertama) hanya memliki 5 elemen (nim, nama, &nama, panggialn&, dan nilai)
* Baris ke-2 hanya memilki 4 elemen (042910503, Muhammad Farras Maruf, dan 98)
* Baris ke-4 memliki 6 elemen (049251522, \*Mark, L, Murphy\*, Mark Murp, dan 92,3)
* dan baris ke-6 hanya memiliki 4 elemen (082885728, Jamesh Goshling, Jams, dan 86,2)

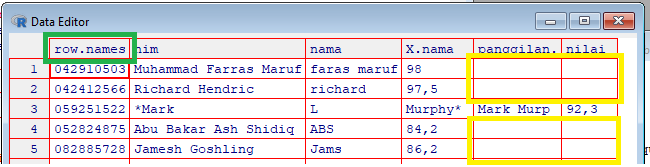
Nilai ‘Muhammad Farras Ma’ruf’, ‘richard’, “James Ghoslin” dan sejenisnya dibaca menjadi satu kesatuan karakter string karena karakter “ (Petik Ganda) dan ‘ (Petik) termasuk dalam quoting character. Sedangkan karakter & (Ampersand) dan \*(Asterisk) tidak, sehingga harus dimasukan dalam argume quotes

Untuk mengetahui lebih dalam mari kita coba menambahkan argumen *fill* dengan nilai true (InsyaAllah argumen *fill* dibahas setelah ini). Intinya argumen *fill* akan mengisi nilai *blank* (kosong) jika ada kasus baris-baris tidak memliki jumlah elemen yang sama.

> dataMahasiswaQuote <- read.table (path, header = TRUE,fill=TRUE)

> fix(dataMahasiswaQuote)

Gambar 3‑3 Hasil dari fix(dataMahasiswaQuote)



Pada Gambar 3‑3 Hasil dari fix(dataMahasiswaQuote) dapat dilihat pada border kuning bahwa nilai dari 2 elemen pada baris ke-1,2 dan ke-4,5 berisi *Blank* untuk menyesuaiakan dengan jumlah elemen dari baris ke-3. Dan pada border hijau berisi nilai *row,names* yang mana nilai tersebut sama untuk menyesuaikan dengan baris ke-3. Untuk membentulkan dan membuat *data frame* sesuai dengan yang kita inginkan dengan kondisi *file* seperti yang dijelaskan diawal. Dapat menggunakan argumen *quote* untuk memasukan daftar karakter yang membuat R paham bahwa karakter tersebut termasuk *quoting character*.

> dataMahasiswaQuote <- read.table (path, header = TRUE,quote="\"'\*&")

> dataMahasiswaQuote

nim nama nama.panggilan nilai

1 42910503 Muhammad Farras Maruf faras maruf 98

2 42412566 Richard Hendric richard 97,5

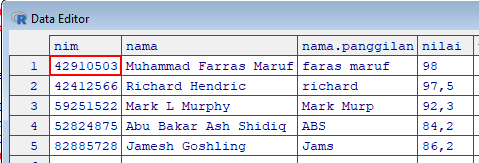
3 59251522 Mark L Murphy Mark Murp 92,3

4 52824875 Abu Bakar Ash Shidiq ABS 84,2

5 82885728 Jamesh Goshling Jams 86,2

> fix(dataMahasiswaQuote)

Gambar 3‑4 Hasil dari fix(dataMahasiswaQuote) dengan argumen quote



Argumen quote="\"'\*&" memberitahu R bahwa nilai yang berada antara karakter “, ‘, \*, dan & adalah sebuah satu kesatuan *character* *string*.

#### Argumen – dec[[6]](#footnote-6)

*Bernilai* *character :* Karakter yang digunakan dalam *file* sebagai tanda desimal.

Sistem komputer saya menggunakan dot (.) sebagai tanda desimal, sehingga 2.4 dibaca dua koma empat (Bisa jadi sistem di komputer anda menggunakan koma sebagai tanda desimal).

Contoh saya memliki *file* yang berisi nilai sebagai berikut ;

nim nama nilai

042910503 "Muhammad Farras Maruf" 98

042412566 "Richard Hendric" 97,5

059251522 "Mark L Murphy" 92,3

052824875 "Abu Bakar Ash Shidiq" 84,2

082885728 "Jamesh Goshling" 86,2

Pada kolom nilai, nilai dari mahasiswa tidak bulat, atau pecahan, sehingga menggunakan tanda desimal. Dalam file tersebut, tanda desimal menggunakan koma (,). Sekarang kita coba untuk membaca file tersebut dan di uji apa tipe data dari nilai tersebut.

> x <- read.table (path, header=TRUE)

> x$nilai

[1] "98" "97,5" "92,3" "84,2" "86,2"

> typeof(x$nilai)

[1] "character"

Ternyata tipe dari kolom nilai adalah character. Hal ini karena pada sistem komputer saya, tanda desimal yang dikenal adalah dot (.) sehingga penggunaan koma pada nilai tersebut dipahami oleh R sebagai sebuah *character*. Untuk menghasilkan sesuai yang diinginkan, kita dapat menggunakan argumen dec=”,” untuk memberitahu R bahwa koma adalah tanda desimal.

> x <- read.table (path, header=TRUE,dec=",")

> x$nilai

[1] 98.0 97.5 92.3 84.2 86.2

> typeof(x$nilai)

[1] "double"

#### Argumen – row.names[[7]](#footnote-7)

*Bernila*i *character atau numerik (dari jumlah row)* : Argumen ini berfungsi untuk menjadikan salah satu vektor didalam *data frame* menjadi nama baris (*row names*).Argumen dari row.names dapat diisi dengan 3 objek data,

1. Isi dari argumen ini dapat berupa vector yang dijadikan nama dari setiap baris

# Isi dari argumen row.names dapat berupA vector

myRowNames <- c("Satu","Dua","Tiga","Empat","Lima")

x <- read.table (path, header=TRUE, row.names=myRowNames)

x

nim nama nilai

Satu 42910503 Muhammad Farras Maruf 98.24289

Dua 42412566 Richard Hendric 97.51242

Tiga 59251522 Mark L Murphy 92.39826

Empat 52824875 Abu Bakar Ash Shidiq 84.20922

Lima 82885728 Jamesh Goshling 86.23024

Pada contoh diatas variabel myRowNames diisi dengan vektor, lalu variabeltersebut dijadikan isi dari parameter *row.names*, yang ditunjukan dengan row.names=myRowNames.

1. atau sebuah *character* *string* yang menunjukan nama dari kolom tabel yang mengandung nama dari setiap baris

> x <- read.table (path, header=TRUE, row.names="nim")

> x

nama nilai

42910503 Muhammad Farras Maruf 98.24289

42412566 Richard Hendric 97.51242

59251522 Mark L Murphy 92.39826

52824875 Abu Bakar Ash Shidiq 84.20922

82885728 Jamesh Goshling 86.23024

Contoh kali ini saya menjadikan kolom nim menjadi nama dari setiap baris. Untuk mengambil nilai dari baris, misalkan saya mau mengambil nam dan nilai dari nim 59251522, maka dapat menggunakan perintah namaVaribelDataFrame(“namaBaris”,”namaKolom”)

> namaSiswa <- x["59251522","nama"]

> nilaiSiswa <- x["59251522","nilai"]

> cat (namaSiswa," mendapatkan nilai ",nilaiSiswa,"\n")

Mark L Murphy mendapatkan nilai 92.39826

1. atau dapat berupa *integer* (angka) yang menunjukan kolom dari tabel yang dijadikan nama dari setiap baris.

> x <- read.table (path, header=TRUE, row.names=1)

> x

nama nilai

42910503 Muhammad Farras Maruf 98.24289

42412566 Richard Hendric 97.51242

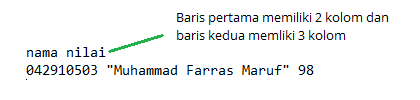
59251522 Mark L Murphy 92.39826

52824875 Abu Bakar Ash Shidiq 84.20922

82885728 Jamesh Goshling 86.23024

Lalu, Jika kita memliki header, dan kolom pertama memliki lebih sedikit baris dari pada row (baris) berikutnya, maka kolom pertama akan dijadikan sebagai nama dari baris walaupun kita tidak mengisi argumen header=TRUE.

Gambar 3‑5 Isi dari File nilai\_mahasiswa\_comma\_fewer\_than\_column.dat



anotherPath <-"Data/nilai\_mahasiswa\_comma\_fewer\_than\_column.dat"

> x <- read.table (anotherPath)

> x

nama nilai

042910503 Muhammad Farras Maruf 98

Untuk melihat perbedaan dengan *file* diatas dengan file yang setiap barisnya memliki kolom yang sama dan sama seperti contoh diatas tidak menggunakan argumen header=TRUE.

> pathWithoutHeader <-"Data/nilai\_mahasiswa\_comma\_fewr\_without\_header.dat"

> x <- read.table (pathWithoutHeader)

> x

V1 V2 V3

1 nim nama nilai

2 042910503 Muhammad Farras Maruf 98

Dapat dilihat pada percoban diatas bahwa kolom ***nim*** tidak dijadikan sebagai nama dari kolom melainkan R secara otomatis menjadikan *generic number* yang berurutan menjadi nama dari setiap baris. Serta karena kita tidak menggunakan perintah header=TRUE maka R menggunakan {V1,V2,V3} sebagai nama dari setiap kolom.

#### Argumen – col.names[[8]](#footnote-8)

*Bernilai vector* : Isi dari vector tersebut dijadikan opsi nama dari kolom pada sebuah *data frame*, Sebagai contoh saya meiliki kumpulan data yang tidak memliki nama dari kolom (*header*)

Gambar 3‑6 Isi dari File nilai\_mahasiswa\_col.names.dat



Lalu kita jalankan perintah berikut dengan tidak menggunakan argumen header=TRUE dan argumen *col.names.*

> x <- read.table(path)

> x

V1 V2 V3

1 42910503 Muhammad Farras Maruf 98.24289

2 42412566 Richard Hendric 97.51242

3 59251522 Mark L Murphy 92.39826

Sehingga nama kolom dari *data frame* tersebut adalah V yang dikuti dengan nomor dari kolom.

Sekarang kita buat sebuah *vektor* yang kita gunakan pada argumen *col.names*, sehingga isi dari *vektor* tersebut menjadi nama kolom pada *data frame* tersebut.

> myColumnName <- c ("NIM","Nama","Nilai")

> x <- read.table(path,col.names=myColumnName)

> x

NIM Nama Nilai

1 42910503 Muhammad Farras Maruf 98.24289

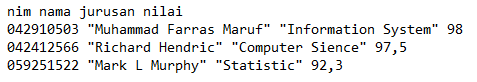
2 42412566 Richard Hendric 97.51242

3 59251522 Mark L Murphy 92.39826

#### Argumen – as.is[[9]](#footnote-9)

Mengendalikan konversi variabel karakter sebagai faktor (jika salah) atau tetap sebagai karakter (bila benar); *as.is* dapat berupa vektor yang mempunyai mode logikal, numerik atau karakter yang mengatur apakah variabel-variabel tetap dipertahankan sebagai karakter.

Gambar 3‑7 Data Mahasiswa



Argumen *as.is* dapat diisi dengan nilai logika saja (bukan vektor berisi logika), yang mana memerintahkan R untuk mengkonversi atau tidak mengkonversi (tergantung isi argumen *as.is*) seluruh kolom yang bertipe *character*. Mari kita coba untuk membaca data seperti pada Gambar 3‑7 Data Mahasiswa.

> x <- read.table(path,header=TRUE)

> typeof(x$nama); class(x$nama); mode(x$nama);

[1] "character"

[1] "character"

[1] "character"

> typeof(x$jurusan); class(x$jurusan); mode(x$jurusan);

[1] "character"

[1] "character"

[1] "character"

Pada contoh diatas kita membuat vairabel x tanpa menggunakan argumen as.is, lalu kita mencoba untuk menampilkan tipe data dari kolom nama (x$nama) dan jurusan (x$jurusan) yang mana kedua kolom tersebut bertipe *character*. Selanjutnya saya coba untuk memasukan nilai FALSE pada argumen *as.is* dan memeriksa tipe tipe dari kedua kolom tersebut, berdasarkan hasil output dapat dilihat kedua kolom tersebut tipe data bukan lagi *charater*, atau dengan kata lain data tersebut telah dikonversi.

> y <- read.table (path,header=TRUE, as.is = FALSE);

> typeof(y$nama); class(y$nama); mode(y$nama);

[1] "integer"

[1] "factor"

[1] "numeric"

> typeof(y$jurusan); class(y$jurusan); mode(y$jurusan);

[1] "integer"

[1] "factor"

[1] "numeric"

Parameter dari argumen *as.is* dapat diisi juga dengan vektor yang berisikan logika. urutan logikan pada vektor tersebut menunjuk pada nomor kolom pada data.frame. Misalkan, berdasarkan data yang kita miliki, kita ingin menkonversi kolom nama saja.

> myParam <- c(FALSE, TRUE, FALSE)

> y <- read.table (path,header=TRUE, as.is = myParam);

> typeof(y$nama); class(y$nama); mode(y$nama);

[1] "character"

[1] "character"

[1] "character"

> typeof(y$jurusan); class(y$jurusan); mode(y$jurusan);

[1] "integer"

[1] "factor"

[1] "numeric"

Parameter dari argumen *as.is* juga dapat diis dengan vekotr yang beriskan numeric. Dalam kasus kita tidak ingin mengkonversi kolom jurusan misalnya. Maka kita dapat memasukan nomor kolom tersebut kedalam vektor.

> myParam <- c(3)

> y <- read.table (path,header=TRUE, as.is = myParam);

> typeof(y$nama); class(y$nama); mode(y$nama);

[1] "integer"

[1] "factor"

[1] "numeric"

> typeof(y$jurusan); class(y$jurusan); mode(y$jurusan);

[1] "character"

[1] "character"

[1] "character"

Selain vektor yang berisikan logika dan numeric, argumen ini juga dapat diisi dengan vektor yang bertipe charater. Character yang dimasukan menujukan ke nama kolom dari data frame. Nama kolom yang dimasukan kedalam vektor tidak akan dikonversi oleh R. Contoh kali inn kita tidak ingin menkonversi kolom nama dan sebaliknya kita ingin menkonversi kolom jurusan, kita dapat menggunakan script berikut.

> myParam <- c("nama")

> y <- read.table (path,header=TRUE, as.is = myParam);

> typeof(y$nama); class(y$nama); mode(y$nama);

[1] "character"

[1] "character"

[1] "character"

> typeof(y$jurusan); class(y$jurusan); mode(y$jurusan);

[1] "integer"

p[1] "factor"

[1] "numeric"

Kita bisa memasukan lebih dari satu numeric atau character di dalam vektor tersebut dengan tujuan agar R tidak mengkonversi kolom-kolom yang dimaksud.

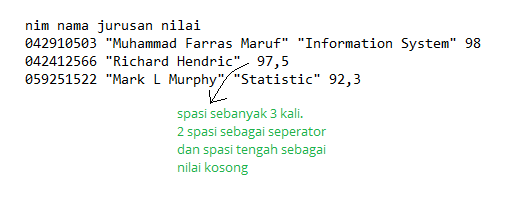
#### Argumen – na.strings[[10]](#footnote-10)

*Argumen ini bernilaikan vector dari string (lebih dari satu kata)* yang diinterpreptasi sebagai nilai Not Available (NA). Blank field (“ “ atau ””) biasyanya dianggap sebagai nilai yang hilang didalam tipe logika, integer, numerik dan kompleks.

Dalam beberapa kasus saya beberapa kali menemukan data dari hasil survey seperti google form yang tidak diisi, sehingga didalam tabel rekap nilai tersebut ditampilkan kosong (“ “,””). Didalam R nilai tersebut (“ “,””) tidak dianggap *NA* jika kita tidak mengisi argumen dari *na.values*.

Kita juga bisa memasukan sebuah nilai ke dalam vektor yang dijadikan argumen *na.strings* agar R menaggapnya sebagai nilai *missing* (NA).

Gambar 3‑8 Data Mahasiswa



nim nama jurusan nilai

1 42910503 Muhammad Farras Maruf Information System 98

2 42412566 Richard Hendric 97,5

3 59251522 Mark L Murphy Statistic 92,3

Contoh, saya memiliki data seperti Gambar 3‑8 Data Mahasiswa dengan hasil dari perintah fix(“data mahasiswa”). Isi dari kolom jurusan baris ke dua adalah kosong, namun bukan nilai *NA*. Kita bisa mengkonversi nilai yang kosong tersebut menjadi *NA* dengan mengisi argumen dari *na.strings*, seperti perintah dibawah ini.

> x <- read.table (path,header=TRUE, sep=" ", na.strings=naValues)

> x

nim nama jurusan nilai

1 42910503 Muhammad Farras Maruf Information System 98

2 42412566 Richard Hendric <NA> 97,5

3 59251522 Mark L Murphy Statistic 92,3

Contoh lain, saya memliki vector seperti ini naValues = c("","Mark L Murphy"). Maka apa yang terjadi dengan *data frame* ?

nim nama jurusan nilai

1 42910503 Muhammad Farras Maruf Information System 98

2 42412566 Richard Hendric <NA> 97,5

3 59251522 <NA> Statistic 92,3

Mark L Murphy akan dianggap sebagai *Not Available Value* karena argumen yang berisikan vektor tersebut didalamnya ada nilai “Mark L Murphy”.

#### Argumen – nrows[[11]](#footnote-11)

*Bernilai integer* : Nilai maksimum baris yang dibaca. Nilai negatif dan nilai yang tidak valid diabaikan.

Contoh pada kasus data mahasiswa. Data tersebut berisi 5 baris mahasiswa. Saya dapat menggunakan argumen nrows seandainya saya hanya ingin menampilkan 2 baris pertama dari data mahasiswa.

> x <- read.table (path,header=TRUE, nrows=2)

> x

nim nama nilai

1 42910503 Muhammad Farras Maruf 98

2 42412566 Richard Hendric 97,5

#### Argumen – skip[[12]](#footnote-12)

*Bernilai integer* : Banyak baris yang dilewatkan dari data *file* sebelum dibaca.

Sebelumnya, argumen *nrows* dapat membantu kita untuk membaca hanya beberapa baris dari baris awal. Argumen ini berkebalikannya, yaitu membaca file dari beberapa baris file yang dilewatkanya. Contoh kita ingin membaca 2 baris terakhir dari data mahasiswa.

> x <- read.table (path,header=TRUE, skip=3)

> x

X059251522 Mark.L.Murphy X92.3

1 52824875 Abu Bakar Ash Shidiq 84,2

2 82885728 Jamesh Goshling 86,2

Argumen tersebut kurang lebih dibaca sepeti ini, “R tolong lewatkan 3 baris pertama dari *data* *file*, dan tampilkan semua data setelahnya”.

#### Argumen – nrows & skip*[[13]](#footnote-13)*

Kita dapat menggunakan kedua argumen tersebut untuk beberapa kasus. Misalnya kita hanyak ingin mengambil baris ke tiga dari data mahasiswa. Maka kita dapat menggunakan argumen nrows dan skip bersamaan.

> x <- read.table (path,header=TRUE, skip=2, nrows=1)

> x

X042412566 Richard.Hendric X97.5

1 59251522 Mark L Murphy 92,3

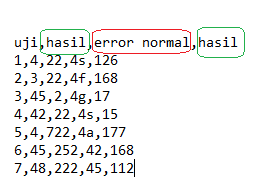
Script diatas dibaca “R tolong lewatkan dua baris pertama dan setelahnya baca hanya satu baris”.

#### Argumen – check.names[[14]](#footnote-14)

*Bernilai Logika* : Jika argumen bernilai TRUE maka nama variabel didalam data frame di periksa dan dipastikan tidak ada nama variabel yang tidak valid, serta juga memastikan nama variabel tidak redudant.

Contoh saya memiliki data seperti Gambar 3‑9 Gamba Data Hasil Uji

Gambar 3‑9 Gamba Data Hasil Uji



Data tersebut memliki nilai yang sama pada baris pertama dan digunakan sebagai header (nama variabel). Secara bawaan (tidak menggunakan argumen *check.names* / check.names = TRUE) R akan memeriksa dan membetulkan nama variabel tersebut jika ada diantara nama varibel yang tidak valid dan redudant.

> x <- read.table (path,header=TRUE,sep=",")

> x

uji hasil error.normal hasil.1

1 4 22 4s 126

2 3 22 4f 168

3 45 2 4g 17

4 42 22 4s 15

5 4 722 4a 177

6 45 252 42 168

7 48 222 45 112

nama variable “error normal” dengan spasi dikoreksi menjadi “error.normal” dan nama variabel hasil yang kedua diganti menjadi “hasil.1” agar tidak ada nama variabel yang sama.

Namun dalam kondisi kita tidak ingin R mengkoreksi nama variabel yang ada didalam data file, kita dapat menggunakan argumen check.names dan di berikan nilai FALSE.

> x <- read.table (path,header=TRUE,sep=",",check.names=FALSE)

> x

uji hasil error normal hasil

1 4 22 4s 126

2 3 22 4f 168

3 45 2 4g 17

4 42 22 4s 15

5 4 722 4a 177

6 45 252 42 168

7 48 222 45 112

#### Argumen – strip.white[[15]](#footnote-15)

*Argumen bernilai logika* : Hanya digunakan jika argumen *sep* disepesifikasikan, dan akan menghapus spasi diawal atau diakhir *field* yang mengandung karakter tanpa spesifikasi *quote*.

nim nama nilai

1 42910503 Muhammad Farras Maruf 98

2 42412566 Richard Hendric 97,5

> if(x[1,"nama"] == "Muhammad Farras Maruf")

+ {x[1,"nilai"]

+ }else{print("tidak ditemukan")}

[1] "tidak ditemukan"

Menggunakan kondisional *if statement* nilai dari “Muhammad Farras Maruf” tidak cocok dengan nilai di *field* ke satu kolom nama. Padahal jika anda bisa melihat pada hasil dari variabel x bahwa pada baris ke 1 pada kolom nama berisikan nilai “Muhammad Farras Maruf” dengan terlihat spasi dikiri dan dikanan. Untuk menghilangkan spasi ini anda dapat menggunakan argumen strip.white=TRUE.

> x <- read.table (path,header=TRUE,sep="|",strip.white=TRUE)

> x

nim nama nilai

1 42910503 Muhammad Farras Maruf 98

2 42412566 Richard Hendric 97,5

> if(x[1,"nama"] == "Muhammad Farras Maruf")

+ {cat("Nilai ",x[1,"nama"], x[1,"nilai"],"\n")

+ }else{print("tidak ditemukan")}

Nilai Muhammad Farras Maruf 98

#### Argumen - comment.char[[16]](#footnote-16)

*Bernilai vektor yang berisikan karakter* : Semua karakter setelah karater yang berada dalam vektor tersebut diabaikan (tidak ditampilkan).

> x <- read.table (path,header=TRUE)

> x

nim nama nilai

1 042910503 Muhammad Farras Maruf 98

2 059251522 Mark L -Murphy 92,3

3 - Ini Komentar

4 $ Juga Komentar

5 @ Sama Komentar

Gunakan argumen comment.char=c(“-“,”$”,”@”) untuk mengabaikan karakter setelah karakter-karater dalam vektor pada argumen tersebut.

#### Argumen - colClasses[[17]](#footnote-17)

*Bernilai karakter atau vektor yang berisi kumpulan dari tipe data* : argumen ini berfungsi menetapkan class kepada kolom-kolom pada data frame. Nilai yang dapat dimaskan didalam argumen ini adalah, NA (nilai bawaan), “NULL” (Jika kolom dilewatkan), satu dari atomic vetor class (logical, integer, numeric, complex, character, dan raw) , atau “factor”, “Data” atau “POSIXct”.

Dibutuhkan sebuah method (dari paket metod) untuk mengkoversi dari “character” ke formal class tertentu.

> myC = c ("character","double","character")

> x <- read.table (path,header=TRUE,dec=",", colClasses =myC)

> x

nim nilai nama

1 042910503 98.0 Muhammad Farras Ma'ruf

2 042412566 97.5 Tania Dwi Haryanti

3 059251522 92.3 Muhammad Faris Ma'ruf

4 052824875 84.2 Al Rayyan

5 082885728 86.2 Vina Lativana

Pada *script* diatas saya menggunakan argumen *colClasses* yang berisi *vector*. *Vector* tersebut berisi “c*haracter”, “double”, dan “character”*. Argumen tersebut memerintahkan R untuk menetapan kolom pertama bertipe character, kolom ke dua bertipe double dan kolom ke tiga bertipe character. Hal tersebut bisa kita buktikan dengan script dibawah ini, untuk menguji tipe dari sebuah nilai.

> cat ("Is ",x$nim[1]," character :", is.numeric (x$nim[1]),"\n")

Is 042910503 character : FALSE

Nilai yang diuji ada pada baris ke satu kolom nim (042910503). Seharusnya tipe data nilai tersebut adalah *numeric*, namun karena pada argumen *colClasses* kita memerintahkan R untuk menetapkan kolom pertama menjadi *character* sehingga semua nilai pada kolom tersebut bertipe *character*.

### Penggunaan Memori

* Memori yang digunakan akan lebih sedikit jika *colClasses* digunakan dan diisi dengan satu atau lebih dari atomic vector classes (numeric, character, logical, integer, complex, raw).
* Menggunakan argumen *nrows* dapat membantu penggunaan memori.
* Penggunaan argumen *comment.char=””* (menon aktifkan) akan lebih cepat dari pada *read.table* (bawaan).
* *read.table* bukan alat yang cocok untuk data yang memilki kolom yang banyak. Fungsi ini dirancanguntuk membaca *dataframe* yang meliki kolom yang memliki tipe kelas yang berbeda-bada.

### Fungsi *scan* dan argumen *scan*()

Berdasarkan yang telah kita ketahui sebelumya, fungsi read.table tidak dirancang untuk membaca *dataframe* yang memliki kolom yang banyak. Untuk membaca *dataframe* yang memiliki kolom yang banyak sangat disarankan menggunakan fungsi *scan().* *InsyaAllah*, dibawah ini saya akan membahas bagaimana cara menggunakan fungsi tersebut dan opsi pengaturan (argumen) yang digunakan pada fungsi tersebut.

#### Deksripsi

Mebaca data kedalam *vector* atau *list* dari konsol atau *file*.

#### Penggunaan

scan(file = "", what = double(), nmax = -1, n = -1, sep = "",

quote = if(identical(sep, "\n")) "" else "'\"", dec = ".",

skip = 0, nlines = 0, na.strings = "NA",

flush = FALSE, fill = FALSE, strip.white = FALSE,

quiet = FALSE, blank.lines.skip = TRUE, multi.line = TRUE,

comment.char = "", allowEscapes = FALSE,

fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)

#### Argumen – File

*Bernilai String* : Argumen ini menunjuk pada nama *file* yang nantinya akan dibaca. Jika nilai dari atribut kosong (“”) maka nilai yang akan diambil berasal dari *keyboard*. (Dalam kasus file=””, nilai yang sedang dimasukan kedalam *vector* atau *list* akan dihentikan dengan baris kosong atau dari EOF signal, Ctrl-D di Unix dan Ctrl-z di Windows)

# Lampiran

Lampiran 4‑1 Kode hasil fungs >example (lm)

> example (lm)

lm> require(graphics)

lm> ## Annette Dobson (1990) "An Introduction to Generalized Linear Models".

lm> ## Page 9: Plant Weight Data.

lm> ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)

lm> trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)

lm> group <- gl(2, 10, 20, labels = c("Ctl","Trt"))

lm> weight <- c(ctl, trt)

lm> lm.D9 <- lm(weight ~ group)

lm> lm.D90 <- lm(weight ~ group - 1) # omitting intercept

lm> ## No test:

lm> anova(lm.D9)

Analysis of Variance Table

Response: weight

Df Sum Sq Mean Sq F value Pr(>F)

group 1 0.6882 0.68820 1.4191 0.249

Residuals 18 8.7292 0.48496

lm> summary(lm.D90)

Call:

lm(formula = weight ~ group - 1)

Residuals:

Min 1Q Median 3Q Max

-1.0710 -0.4938 0.0685 0.2462 1.3690

Coefficients:

Estimate Std. Error t value Pr(>|t|)

groupCtl 5.0320 0.2202 22.85 9.55e-15 \*\*\*

groupTrt 4.6610 0.2202 21.16 3.62e-14 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6964 on 18 degrees of freedom

Multiple R-squared: 0.9818, Adjusted R-squared: 0.9798

F-statistic: 485.1 on 2 and 18 DF, p-value: < 2.2e-16

lm> ## End(No test)

lm> opar <- par(mfrow = c(2,2), oma = c(0, 0, 1.1, 0))

lm> plot(lm.D9, las = 1) # Residuals, Fitted, ...

Waiting to confirm page change...

lm> par(opar)

lm> ## Don't show:

lm> ## model frame :

lm> stopifnot(identical(lm(weight ~ group, method = "model.frame"),

lm+ model.frame(lm.D9)))

lm> ## End(Don't show)

lm> ### less simple examples in "See Also" above

Lampiran 4‑2 Berbagai Cara untuk Mengakses Nilai dari Data Frame

> # Berbagai cara untuk mengakses nilai dari data frame

> myData$V1

[1] "Nama" "Muhammad Farras Ma'ruf" "Tania Dwi Haryanti"

[4] "Muhammad Faris Ma'ruf" "Another Person"

> typeof (myData$V1);mode (myData$V1);class (myData$V1)

[1] "character"

[1] "character"

[1] "character"

> myData$V1[2]

[1] "Muhammad Farras Ma'ruf"

> # atau -----

> myData[,1]

[1] "Nama" "Muhammad Farras Ma'ruf" "Tania Dwi Haryanti"

[4] "Muhammad Faris Ma'ruf" "Another Person"

> typeof (myData[,1]);mode (myData[,1]);class (myData[,1])

[1] "character"

[1] "character"

[1] "character"

> myData[,3][2]

[1] "98"

> #-----------

> # Mengakses kolom dari data frame

> myData[1]

V1

1 Nama

2 Muhammad Farras Ma'ruf

3 Tania Dwi Haryanti

4 Muhammad Faris Ma'ruf

5 Another Person

> typeof (myData[1]);mode (myData[1]);class (myData[1])

[1] "list"

[1] "list"

[1] "data.frame"

> myData[5,1]

[1] "Another Person"

1. Script : [Belajar R\Modul 3\K.B 2\RScript\1. Cara Membaca File dan Penjelasannya.R](Belajar%20R/Modul%203/K.B%202/RScript/1.%20Cara%20Membaca%20File%20dan%20Penjelasannya.R) [↑](#footnote-ref-1)
2. Script : [Belajar R\Modul 3\K.B 2\RScript\3. read.table - file.R](Belajar%20R/Modul%203/K.B%202/RScript/3.%20read.table%20-%20file.R) [↑](#footnote-ref-2)
3. Script : [Belajar R\Modul 3\K.B 2\RScript\4. read.table-header.R](Belajar%20R/Modul%203/K.B%202/RScript/4.%20read.table-header.R) [↑](#footnote-ref-3)
4. Script : [Belajar R\Modul 3\K.B 2\RScript\5. read.table-sep.R](Belajar%20R/Modul%203/K.B%202/RScript/5.%20read.table-sep.R) [↑](#footnote-ref-4)
5. Script : [Belajar R\Modul 3\K.B 2\RScript\6. read.table-quote.R](Belajar%20R/Modul%203/K.B%202/RScript/6.%20read.table-quote.R) [↑](#footnote-ref-5)
6. Script : [Belajar R\Modul 3\K.B 2\RScript\7. read.table-dec.R](Belajar%20R/Modul%203/K.B%202/RScript/7.%20read.table-dec.R) [↑](#footnote-ref-6)
7. Script : [Belajar R\Modul 3\K.B 2\RScript\9. read.table-row.numes.R](Belajar%20R/Modul%203/K.B%202/RScript/9.%20read.table-row.numes.R) [↑](#footnote-ref-7)
8. Script : [Belajar R\Modul 3\K.B 2\RScript\10. read.table-col.numes.R](Belajar%20R/Modul%203/K.B%202/RScript/10.%20read.table-col.numes.R) [↑](#footnote-ref-8)
9. Script : [Belajar R\Modul 3\K.B 2\RScript\11. read.table-as.is.R](Belajar%20R/Modul%203/K.B%202/RScript/11.%20read.table-as.is.R) [↑](#footnote-ref-9)
10. Script : [Belajar R\Modul 3\K.B 2\RScript\12. read.table-na.string.R](Belajar%20R/Modul%203/K.B%202/RScript/12.%20read.table-na.string.R) [↑](#footnote-ref-10)
11. Script : [Belajar R\Modul 3\K.B 2\RScript\13. read.table-nrows.R](Belajar%20R/Modul%203/K.B%202/RScript/13.%20read.table-nrows.R) [↑](#footnote-ref-11)
12. Script : [Belajar R\Modul 3\K.B 2\RScript\14. read.table-skip.R](Belajar%20R/Modul%203/K.B%202/RScript/14.%20read.table-skip.R) [↑](#footnote-ref-12)
13. Script : [Belajar R\Modul 3\K.B 2\RScript\15. read.table-nrows-skip.R](Belajar%20R/Modul%203/K.B%202/RScript/15.%20read.table-nrows-skip.R) [↑](#footnote-ref-13)
14. Script : [Belajar R\Modul 3\K.B 2\RScript\16. read.table-check.names.R](Belajar%20R/Modul%203/K.B%202/RScript/16.%20read.table-check.names.R) [↑](#footnote-ref-14)
15. [↑](#footnote-ref-15)
16. Script : [Belajar R\Modul 3\K.B 2\RScript\18. read.table-comment.char.R](Belajar%20R/Modul%203/K.B%202/RScript/18.%20read.table-comment.char.R) [↑](#footnote-ref-16)
17. Script : [Belajar R\Modul 3\K.B 2\RScript\19. read.table-colClasses.R](Belajar%20R/Modul%203/K.B%202/RScript/19.%20read.table-colClasses.R) [↑](#footnote-ref-17)