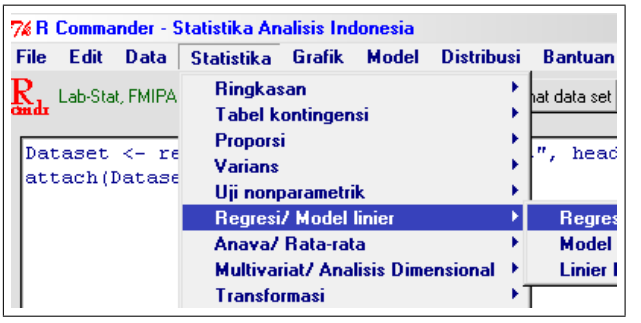
Buku Panduan
Program Statistika R



I Made Tirta

I Made Tirta UNEJ 2015

DAFTAR ISI

1 SCIVIEWS: UNTUK PROGRAMER DAN STATISTIKA KHUSUS 19

1.1 Menu SciViews 23

1.2 Mengenal Menu Dock SciView Lebih Jauh 27

1.2.1 Menu Galeri Grafik 32

1.2.2 Menu R Commander versi SCiViews 33

1.2.3 Jendela File Explorer 35

1.2.4 Jendela Objek Explorer 36

1.2.5 Jendela Kartu Referensi 40

1.2.6 Menu Web dan Dokumentasi 49

1.3 Referensi Lebih Lanjut 50

2 RCLI: MELANGKAH MELAMPAUI MENU 53

2.1 R dengan RCLI 54

2.2 Editor Skrip dengan Tinn-R 55

2.3 Beberapa Fungsi Penting dalam R 57

2.3.1 Fungsi Dasar Matematika 57

2.3.2 Operasi Vektor dan Matriks 59

2.3.3 Fungsi Dasar Statistika 62

2.3.4 Fungsi Pembangkit Data Peubah Acak 64
2.3.5 Fungsi untuk Menangani Grafik 66
2.3.6 Aneka Rupa Perintah R 70
2.4 Menulis Program pada R 75
2.4.1 Komponen-Komponen Program 75
2.4.2 Langkah- langkah Penting dalam Penulisan Program 77
2.4.3 Mendefinisikan Fungsi dalam R 81
2.4.4 Mengevaluasi Nilai Fungsi 85
2.4.5 Mengemas Keluaran Fungsi 88
2.4.6 Menghindarkan *Loop* 90
2.4.7 Menghitung Akar-Akar Persamaan dengan Metode Numerik 94
2.5 Mendefinisikan Ulang Fungsi Internal 102
2.6 Memanfaatkan Paket Program R 104
2.6.1 Mencari Informasi Paket Program R 104
2.6.2 Memanggil Pustaka dengan CLI 109
2.7 Mengemas Fungsi Menjadi Paket 111
2.8 CLI dan RCommander 113
2.9 Bacaan Lebih lanjut 114

Daftar Istilah **207**

A Daftar Paket Pustaka/*Library* R **165**

B Beberapa Demo Pustaka **213**

C Beberapa Fungsi Penting R **219**
C.1 Fungsi Terkait Grafik 220
abline 220
biplot 221
interaction.plot 223
par 226
rgl 237

C.2 Fungsi Terkait Simulasi dan Organisasi Data 238
Normal 239
mvrnorm 241
Random 242
C.3 Fungsi Terkait RGUI 247
tk2theme 247
C.4 Fungsi terkait Formula Model Statistika 252
formula 252
C.5 Fungsi dari Paket dalam Bahasa Indonesia 255
StatDemo 256
C.5.1 *Skrip Dokumen Paket* 258

DAFTAR GAMBAR

1.1 Menu Standar SciView 26
1.2 Menu Dock SciViews 41
1.3 Menu Galeri Grafik SciViews 42
1.4 Menu Bahasa Indonesia dari Dock R Commander 43
1.5 Menu Eksplorer File 44
1.6 Menu Eksplorer Objek 1 45
1.7 Menu Eksplorer Objek 2 46
1.8 Menu Kartu Referensi 47
1.9 Menu Dokumentasi 48

2.1 Menulis Skrip pada Jendela Command 55
2.2 Tampilan Tinn-R 56
2.3 Lay Out Multi Grafik 66
2.4 Lay Out Multi Grafik 68
2.5 Lay Out Kompleks 70
2.6 Lay Out Kompleks 71
2.7 Ilustrasi Simulasi Mean Populasi dan Sata-rata sampel . 82
2.8 Contoh Grafik Fungsi Parametrik Dimensi dua 82

2.9 Ilustrasi Maksimum/ Minimum dengan Newton Raphson 97
2.10 Contoh Grafik dari Pustaka mgcv 110
2.11 Skrip pada Jendela Skrip R Commander 113
2.12 Skrip pada R Commander 115

DAFTAR TABEL

2.1 Daftar Fungsi Matematika 58
2.2 Daftar Operasi Vektor dan Matriks 60
2.3 Fungsi Dasar Statistika 62
2.4 Fungsi Pembangkit Data 65
2.5 Fungsi R untuk Grafik 67
2.6 Aneka Rupa Fungsi R 74

BAB 2

RCLI: MELANGKAH MELAMPAUI MENU

Tujuan Umum

Bab ini terutama disediakan bagi pengguna R yang memerlukan kemampuan R melebihi yang telah disediakan dalam menu R Commander dengan tujuan agar pembaca dapat mendefinisikan atau menjalankan fungsi-fungsi R melalui RCLI.

Tujuan Khusus

Setelah membaca bab ini pembaca diharapkan dapat:

1. menulis skrip R;
2. mengaktifkan dan memanfaatkan pustaka R sesuai kebutuhan;
3. membuat skrip simulasi data dan analisis dengan efisien;
4. memahami, meniru, dan memodifikasi keluaran yang dihasilkan oleh R Commander melalui RCLI.

2.1 R dengan RCLI

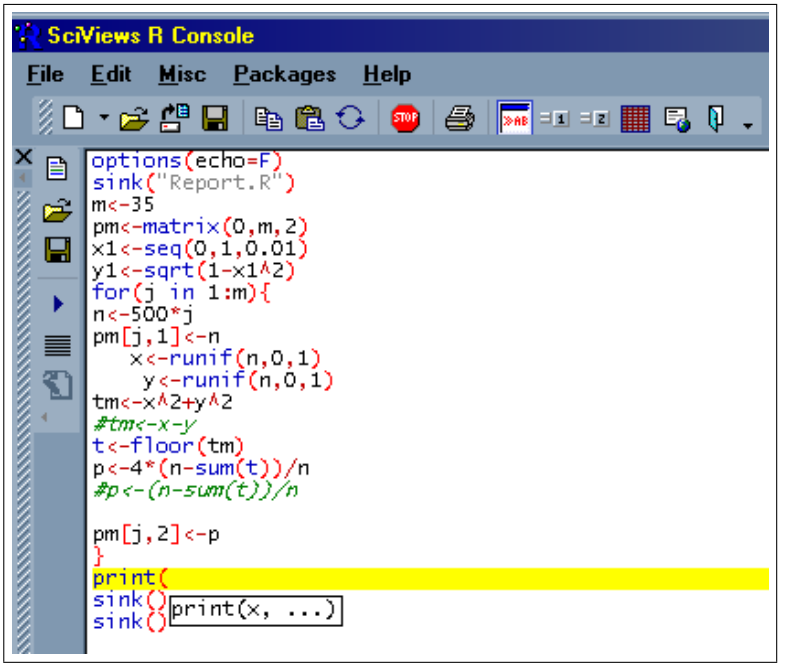
Dengan menggunakan RCLI diperoleh bentuk tampilan `R-Console` yang sama, baik dengan menggunakan sistem operasi Windows maupun Linux. Bentuk tampilan layar `R console` untuk R versi 2.0.0 adalah bagai berikut ini.

```
R version 2.6.1 (2007-11-26)
Copyright (C) 2007 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

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

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

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

Dengan adanya RGUI SciViews, seperti telah dibahas sebelumnya, maka penulisan dan eksekusi skrip dapat lebih mudah dilakukan melalui Jendela Command SciViews. Beberapa keuntungan menggunakan Jendela Command SciViews adalah:

1. telah dilengkapi panel penting seperti buka skrip, simpan skrip, dan eksekusi skrip;
2. dapat memeriksa kecocokan pasangan tanda kurung (), dan {};
3. bisa memberi arahan dalam memanggil fungsi yang telah didefinisikan;
4. dapat mengeksekusi skrip per baris atau secara keseluruhan;

5. tampilan memberikan warna berbeda (*script highlighting*) untuk notasi tertentu dan antara skrip dengan komentar.



Gambar 2.1: Skrip pada Jendela Command SciViews.

Semua kemampuan tersebut sangat membantu baik dalam mengedit maupun dalam mengeksekusi skrip (Lihat Gambar 2.1).

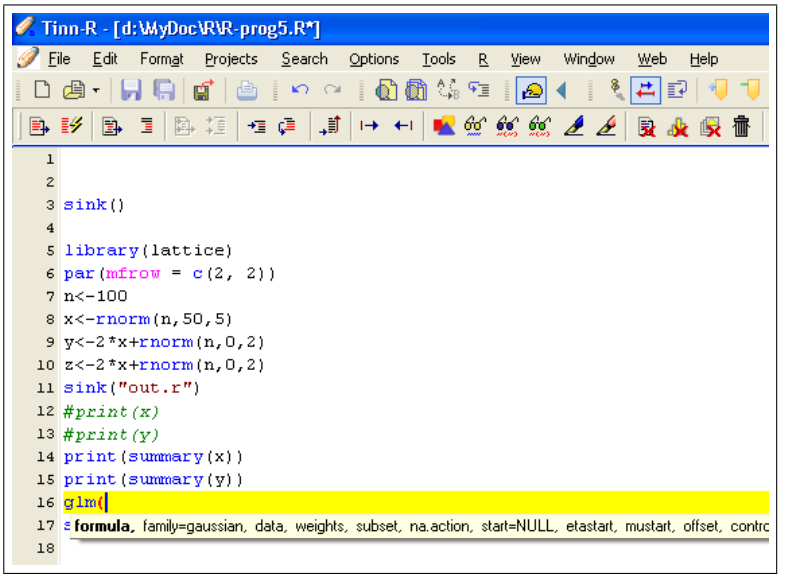
2.2 Editor Skrip dengan Tinn-R

Selain dengan SciView seperti telah diuraikan sebelumnya, tersedia juga editor dengan kemampuan sangat mirip dengan SciViews yang juga secara khusus dirancang untuk R, yaitu Tinn-R. Pada CD terlampir juga telah disediakan program Tinn-R versi 1.19. Untuk men-

goptimalkan komunikasi dengan Tinn-R (misalnya melacak paket-paket dan objek-objek yang aktif), kita harus mengaktifkan beberapa paket dengan perintah berikut melalui RConsule atau melalui Tinn-R.

```
library(svMisc)
library(svWidgets)
library(svIDE)
library(tcltk)
library(tcltk2)
```

Tampilan Tinn-R dapat dilihat pada Gambar 2.2



Gambar 2.2: Tampilan Tinn-R.

R termasuk pemrograman berorentasi objek. Semua hal yang dikerjakan dalam R dapat disimpan dalam bentuk objek dengan mendefinisikannya. Definisi dalam R menggunakan notasi <-. Per-

hatikan bahwa notasi garis bawah (*under score*) "_" yang biasa dipakai pada S-Plus bermakna lain pada R. Seperti halnya S-Plus, R sangat peka terhadap huruf besar/ kecil (*case sensitive*).

```
NamaObjek<-definisi
```

Contoh 2.1. Pada Contoh berikut objek X bernilai 9 dan objek x bernilai 8.

```
>x<-2^3
>X<-3^2
>x
[1] 8
>X
[1] 9
```

Objek R dapat berupa konstanta, vektor/ matriks atau fungsi. Agar objek yang didefinisikan dapat disimpan secara permanen, maka pada akhir penutupan program R harus dijawab **Y (yes)** ketika diminta konfirmasi *Save workspace image ?*

2.3 Beberapa Fungsi Penting dalam R

R telah dilengkapi dengan banyak fungsi yang dapat dimanfaatkan untuk penulisan skrip program. Selain fungsi-fungsi yang terdapat dalam paket standar, banyak fungsi yang didefinisikan dalam berbagai pustaka R.

2.3.1 Fungsi Dasar Matematika

Beberapa fungsi dasar telah didefinisikan secara internal di dalam R. Fungsi-fungsi tersebut dapat dilihat pada Tabel 2.1. Terhadap matriks atau vektor, operasi tersebut bekerja perunsur. Berikut beberapa contoh penerapan dari fungsi-fungsi tersebut.

Tabel 2.1: Daftar Beberapa Fungsi Matematika Penting dalam R

No	Nama Fungsi	Notasi Matematika	Fungsi R
1	harga mutlak	$\|\|$	`abs()`
2	goniometri	$\sin, \cos, \tan$	`sin(), cos(), tan()`
3	invers goniometri	$\sin^{-1}, \cos^{-1}, \tan^{-1}$	`asin(), acos(), atan()`
4	hiperbolikus	$\sinh, \cosh, \tanh$	`sinh(), cosh(), tanh()`
5	exponensial dan log	$\exp, \ln$	`exp(), log()`
6	logaritma 10	$\ln^{10} \log$	`log10()`
7	gamma	$\Gamma()$	`gamma()`
8	bilangan bulat terkecil yang tidak kurang dari bilangan tertentu	$\ulcorner x \urcorner$	`ceiling()`
9	bilangan bulat terbesar yang tidak lebih dari	$\llcorner x \lrcorner$	`floor()`
10	bagian bulat suatu bilangan		`tranc()`
11	bilangan bulat terdekat		`round()`
12	pembulatan ke desimal tertentu		`signif`

2.3.2 Operasi Vektor dan Matriks

Untuk matriks atau vektor yang berdimensi sama, operasi hitung biasa dapat dilakukan dan itu akan dikerjakan berdasarkan unsur-unsur yang bersesuaian seperti pada contoh di atas. Khusus untuk opersi vektor dan matriks, R memiliki operasi dasar seperti yang ditunjukkan dalam Tabel 2.2 pada halaman 60.

Contoh 2.2. Misalkan kita ingin membentuk barisan dengan nilai awal 0, nilai akhir 10 dan banyaknya unsur 10, maka perintah dan hasilnya adalah sebagai berikut.

```
>x<-seq(0,10,length=8)
x [1]
0.000000 1.428571  2.857143 4.285714 5.714286  7.142857  8.571429
[8] 10.000000

> rep(c("A","B","C"),5)
 [1] "A" "B" "C" "A" "B" "C" "A" "B" "C" "A" "B" "C" "A" "B" "C"

> rep(1:3,each=5)
 [1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3
> rep(1:3,5)
 [1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
> rep(1:3,each=2,5)
 [1] 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3
```

Beberapa cara pembulatan pecahan desimal dapat dilakukan dengan berbagai perintah berikut.

```
 round(x)
[1]  0  1  3  4  6  7  9 10
>floor(x)
[1]  0  1  2  4  5  7  8 10
> ceiling(x)
[1]  0  2  3  5  6  8  9 10
>signif(x,3)
```

Tabel 2.2: Daftar Operasi Vektor dan Matriks dalam R. Operasi lanjut yang lebih spesifik dapat dilihat pada pustaka `matrix`

No	Nama Fungsi/ Operasi	Notasi Matematika	Fungsi R
1	pembentukan matriks	$\mathbf{x}$	`matrix(data, nbaris, nkolom);`
2	pembentukan barisan		`seq(awal, akhir, kenaikan);` `seq(awal, akhir, length=n)`
3	barisan berpola		`rep((el.), replik)` atau `rep((elemen),each=n)`
4	transpose matriks	$\mathbf{x}^T$	`t(x)`
5	determinan matriks	$det(\mathbf{x})$	`det(x)`
6	matriks diagonal	$\mathbf{D}$	`diag(data)`
7	diagonal matriks		`diag(matriks)`
8	teras (*trace*) matriks		`tr(matriks)`
9	perkalian matriks	$\mathbf{xy}$	`x %*% y`
10	inverse matriks	$\mathbf{x}^{-1}$	`solve(x)`
11	jumlah berturutan elemen- elemen yang sudah dikenakan fungsi f	$\sum f(x)$	`sum(f(x))`
12	hasil kali berturutan elemen-elemen yang sudah dikenakan fungsi f	$\prod f(x)$	`prod(f(x))`
13	menunjuk elemen ke i vektor $\mathbf{v}$ atau elemen ke i, j matriks $\mathbf{x}$	v_i, x_{ij}	`v[i], x[i,j]`

```
[1]  0.00  1.43  2.86  4.29  5.71  7.14  8.57 10.00
> signif(x,4)
[1]  0.000  1.429  2.857  4.286  5.714  7.143  8.571 10.000
```

Contoh 2.3. Misalnya, kita memiliki dua vektor, yaitu $\mathbf{X} = \begin{pmatrix} 4 \\ 5 \\ 3 \\ 6 \end{pmatrix}$ dan $\mathbf{Y} = \begin{pmatrix} 2 \\ 4 \\ 3 \\ 6 \end{pmatrix}$, maka hasil berbagai operasi hitung biasa di antara kedua vektor ini adalah

```
> x<-matrix(c(4,5,3,6),4,1)
> y<-matrix(c(2,4,3,6),4,1)
> x*y
     [,1]
[1,]    8
[2,]   20
[3,]    9
[4,]   36
> x/y
     [,1]
[1,] 2.00
[2,] 1.25
[3,] 1.00
[4,] 1.00

> sum(log(x))
[1] 5.886104
> prod(log(x))
[1] 4.39191
```

Hasil beberapa operasi vektor atau matriks diperoleh seperti berikut.

```
> x%*%t(y)
     [,1] [,2] [,3] [,4]
```

```
[1,]    8   16   12   24
[2,]   10   20   15   30
[3,]    6   12    9   18
[4,]   12   24   18   36

> t(x)%*%y
     [,1]
[1,]   73

>solve(t(x)%*%y)
           [,1]
[1,] 0.01369863

> x[2]
[1] 5
```

2.3.3 Fungsi Dasar Statistika

Selain fungsi dasar dalam matematika, R juga mempunyai sekumpulan fungsi dasar yang biasa dipergunakan dalam bidang statistika. Variabel dalam fungsi statistika ini berupa vektor data. Fungsi-fungsi ini dirangkum pada Tabel 2.3.

Tabel 2.3: Fungsi Dasar Statistika pada R

No	Nama Fungsi	Notasi Statistika	Fungsi perintah dalam R
1	minimum. maksimum	min, max	`min(), max()`
2	range	range	`range()`
3	mean, median,	$\bar{\mathbf{x}}$, median	`mean(), median()`
4	variance	S^2	`var()`
5	correlation	ρ_{xy}	`cor(x,y)`
6	ringkasan data		`summary()`
7	contoh/ sampling data		`sample()`

`range` memberikan informasi minimum dan maksimum secara serempak.endtable

Contoh 2.4. Diketahui dua vektor $\mathbf{X}, \mathbf{Y}$ seperti pada contoh sebelumnya. Jika masing-masing vektor diperlakukan sebagai data, maka hasil terhadap beberapa fungsi statistika tadi adalah:

```
> min(x)
[1] 3

> max(y)
[1] 6
> mean(x)
[1] 4.5

> var(y)
         [,1]
[1,] 2.916667

> cor(x,y)
          [,1]
[1,] 0.8315218

> range(x)
[1] 3 6

> range(y)
[1] 2 6

> sample(0:1,30,replace=T) # simulasi Tos Uang  logam
 [1] 0 1 1 1 1 0 0 1 1 1 1 0 0 1 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 1

> sample(c("A","G"),15,replace=T)
 [1] "G" "G" "A" "A" "G" "G" "A" "A" "G" "G" "G" "A" "A" "G" "G"

> sample(1:6,30,replace=T) # Simulasi Tos Dadu
 [1] 6 3 2 6 4 1 1 4 2 3 6 5 3 4 3 4 2 3 1 4 1 5 3 5 1 5 1 6 4 1
```

2.3.4 Fungsi Pembangkit Data Peubah Acak

Disamping fungsi dasar yang telah dibicarakan, R juga memiliki fungsi-fungsi statistika lain yang banyak dipergunakan dalam simulasi data. Fungsi- fungsi ini merupakan fungsi untuk membangkitkan data dari peubah acak dengan berbagai distribusi yang banyak dijumpai, seperti normal, poisson, dan gamma dengan jumlah/ ukuran sampel n. Pada dasarnya ada empat jenis fungsi terkait dengan distribusi peubah acak yaitu sebagai berikut.

rdistribusi untuk membangkitkan data acak/ random dari suatu distribusi dengan Parameter tertentu.

ddistribusi untuk mencari nilai fungsi kepadatan $f(x)$ pada suatu nilai x tertentu.

pdistribusi untuk mencari luas daerah (nilai peluang) suatu distribusi yang dibatasi oleh nilai x tertentu

qdistribusi untuk mencarti nilai x yang membatasi luas daerah (nilai peluang) tertentu dari suatu distribusi

Dalam istilah di atas, `distribusi` merupakan nama-nama distribusi yang tersedia pada R di antaranya beberapa yang penting yang banyak dipakai adalah `norm` (normal), `gamma` (Gamma),`t`(t), `F`(F), `chisq`(χ^2), `pois` (Poisson), `binom`(Binomial). Sebagian daftar fungsi- fungsi ini dapat dilihat pada Tabel 2.4.

Contoh 2.5. Misalkan kita ingin mensimulasi data dari distribusi normal dengan parameter populasi $\mu = 50$ dan $\sigma = 5$. Kita dapat menghitung mean (rata-rata) maupun variansi sampel dari data yang dibangkitkan untuk melihat kedekatannya dengan μ dan σ^2.

```
>mean(rnorm(100,50,5))
[1] 50.19985

>var(rnorm(100,50,5))
[1] 26.99507
```

Tabel 2.4: Fungsi Pembangkit Data pada R. Selain fungsi membangkitkan data acak juga terdapat fungsi-fungsi menghitung peluang distribusi.

No	Nama Distribusi	Parameter	Perintah dalam R

1	Normal $N(\mu, \sigma^2)$	mean=μ, varians=σ^2	`rnorm(n, mean, sigma);`
			`dnorm(x, mean, sigma);`
			`pnorm(x, mean, sigma);`
			`qnorm(p, mean, sigma)`

2	Gamma $G(\alpha, \beta)$	$\mu = \alpha/\beta$
		$\sigma^2 = \alpha/\beta^2$
			`rgamma(n, alpha,beta);`
			`dgamma(x, alpha, beta);`
			`pgamma(x, alpha,beta);`
			`qgamma(p, alpha, beta)`

3	Poison(λ)	$\mu = \sigma^2 = \lambda$	`rpois(n,lamda);`
			`dpois(x, lambda);`
			`ppois(x, lambda);`
			`qpois(p, lambda)`

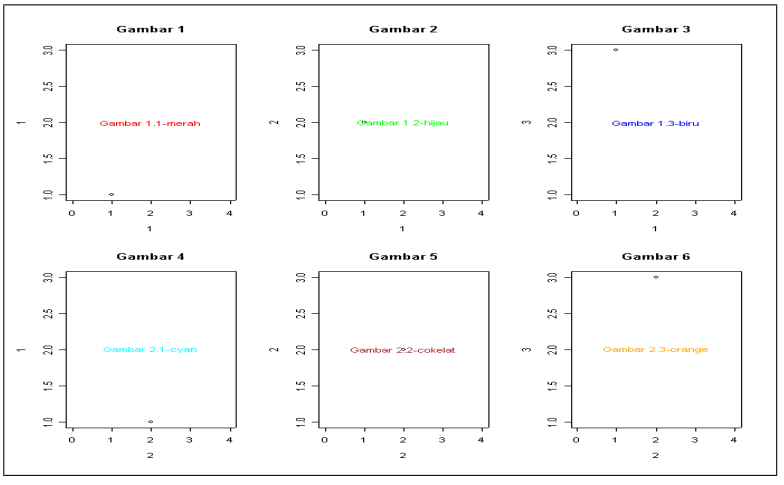
4	Binomial (s, π)	$\mu = s\pi$
		$\sigma^2 = s\pi(1-\pi)$
			`rbinom(n, s, pi);`
			`dbinom(x, s, pi);`
			`pbinom(x, s, pi);`
			`qbinom(p, s, pi)`

5	Chi-kuadrat (χ^2_ν)		`rchisq(n,nu);`
			`dchisq(x, nu);`
			`pchisq(x, nu);`
			`qchisq(p,nu)`

6	t_ν		`rt(n,nu)`
			`dt(x,nu)`
			`pt(x,nu)`
			`qt(p,nu)`

2.3.5 Fungsi untuk Menangani Grafik

Untuk menangani grafik, R memiliki beberapa fungsi seperti ditunjukkan pada Tabel2.5. Dokumentasi yang lebih lengkap dapat diperoleh dengan menggunakan perintah help(...). Di antara fungsi ini adalah untuk membuat *lay out* lembaran grafik yang dibagi menjadi matriks sublembaran kecil $(a \times b)$. Masing-masing sublembaran dapat memiliki judul dan absis sendiri (lihat Gambar 2.3).



Gambar 2.3: *Lay Out* Multi Grafik $m \times n$

Contoh 2.6. Misalkan kita inginkan 1 lembar tampilan grafik dibagi menjadi 6 subgrafik yang tersusun atas 2×3 (2 baris dan 3 kolom). Tentu saja sumbu grafik ini diatur sehingga yang biasa diberi label sumbu adalah semua sumbu bawah, sumbu paling kiri, sumbu atas, dan sumbu paling kanan. Dengan demikian, perintahnya adalah seperti berikut dan hasil grafiknya dapat dilihat pada Gambar 2.3.

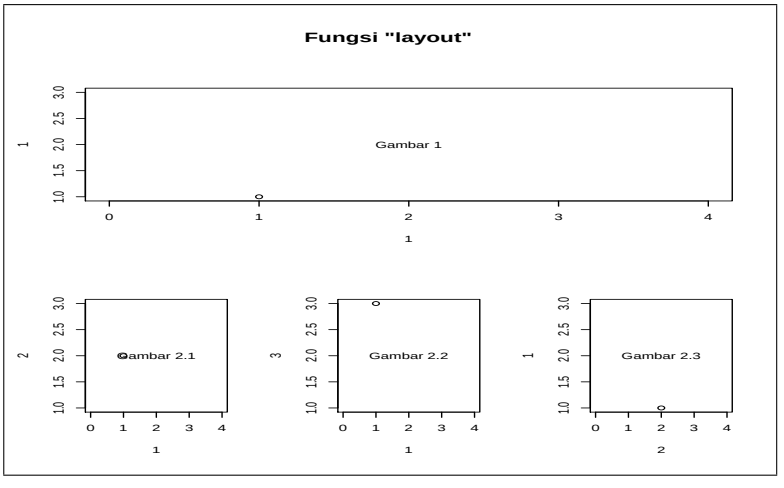
```
par(mfrow=c(2,3))
plot(1,1,xlim=c(0,4),ylim=c(1,3),)
```

Tabel 2.5: Beberapa Fungsi Dasar R untuk Grafik. Tersedia fungsi baik untuk dua maupun tiga dimensi.

No	Tujuan	Perintah R	Keterangan
1	membuat *lay out* multigrafik (banyak layar)	`par(mfrow=c(b, k))`	b=banyak baris k=banyak kolom
2	membuat diagram (grafik pencaran= p, dan garis =l)	`plot(x,y, type='l/p/b', xlab="", ylab="", lty=0, ylim=c(,))`	l=line(grafik garis) p=point(grafik titik) b=keduanya
3	menambah garis pada grafik yang sudah ada	`lines(x, y, lty=, pch= 'numerik')`	angka numerik menunjukkan jenis garis
4	menambah titik pada grafik yang sudah ada	`points(x, y, pch='numerik/simbol')`	numerik/simbol menunjukkan jenis titik
5	memunculkan sumbu	`axis(1,outer=T,las=1)`	Nomor aksis 1=bawah 2=kiri, 3=atas, 4=kanan.
6	menambah text	`text(x,y,"teks")`	
7	memberi Legend	`legend(xpos, ypos, "label", lty=1:m, marks=)`	lty=jenis garis, marks= jenis label titik, m=banyaknya label
8	membuat grafik normalitas data	`qqnorm(data)` dan `qqline(data)`	dikerjakan berurutan untuk memperoleh grafik titik dan garis
9	membuat grafik tiga dimensi	`persp(x,y,z)` `contour(x,y,z)`	Data dalam bentuk matriks
10	membuat*lay out* kompleks	`layout(matriks)`	Unsur matriks diisi posisi grafik
11	membagi layar	`split.screen(c(baris, kolom), no.layar)`	no.layar diurut berdasarkan urutan pembagian

```
text(2,2,"Gambar 1.1")
plot(1,2,xlim=c(0,4),ylim=c(1,3))
text(2,2,"Gambar 1.2")
plot(1,3,xlim=c(0,4),ylim=c(1,3))
text(2,2,"Gambar 1.3")
plot(2,1,xlim=c(0,4),ylim=c(1,3))
text(2,2,"Gambar 2.1")
plot(2,2,xlim=c(0,4),ylim=c(1,3))
text(2,2,"Gambar 2.2")
plot(2,3,xlim=c(0,4),ylim=c(1,3))
text(2,2,"Gambar 2.3")
```

Lay out yang lebih kompleks dapat dilakukan dengan menggunakan fungsi `layout()` dan `split.screen()`. Pada program berikut *lay out* layar dibagi menjadi 4 wilayah 1 bagian atas dan 3 bagian bawah. Hasilnya dapat dilihat pada Gambar 2.4



Gambar 2.4: Out Multi Grafik Kompleks

```
layout(matrix(c(1, 1, 1
                2, 3, 4), nr = 2, byrow = TRUE))
```

```
plot(1,1,xlim=c(0,4),ylim=c(1,3),)
text(2,2,"Gambar 1")
plot(1,2,xlim=c(0,4),ylim=c(1,3))
text(2,2,"Gambar 2.1")
plot(1,3,xlim=c(0,4),ylim=c(1,3))
text(2,2,"Gambar 2.2")
plot(2,1,xlim=c(0,4),ylim=c(1,3))
text(2,2,"Gambar 2.3")
mtext("Fungsi \"layout\" ", side = 3, outer = TRUE,
      font = 2)
```

Pada program berikut layar dibagi menjadi 5 wilayah layar tetapi dengan 7 nomor dengan cara berikut.

1. `split.screen(2,1)` membagi layar utama menjadi 2 baris 1 kolom dengan `screen(1)` adalah layar bagian atas dan `screen(2)` adalah layar bagian bawah;
2. `split.screen(c(2,1),1)` membagi layar `screen(1)` menjadi 2 bagian dengan nama `screen(3), screen(4)`;
3. `split.screen(c(1,2),1)` membagi layar `screen(1)` menjadi 2 bagian dengan nama `screen(3), screen(4)`;
4. `split.screen(c(1,3),2)` membagi layar `screen(2)` menjadi 3 bagian dengan nama `screen(5), screen(6), screen(6)`;

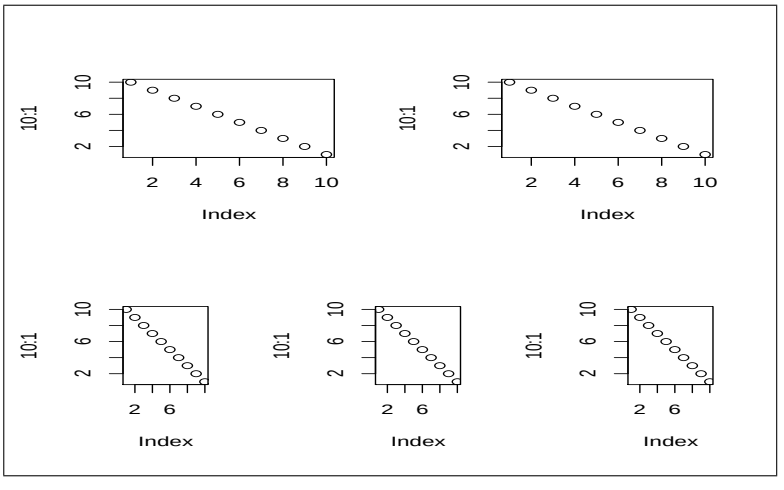
Selanjutnya pembuatan grafik disesuaikan dengan nomor layar yang diinginkan, seperti pada program berikut.

```
split.screen(c(2,1))
split.screen(c(1,2), screen = 1)
split.screen(c(1,3), screen = 2)
screen(3)
plot(10:1)
screen(4)
plot(10:1)
screen(5)
plot(10:1)
screen(6)
plot(10:1)
```

```
screen(7)
plot(10:1)
```

Hasilnya program di atas, dapat dilihat pada Gambar 2.5. Sementara Gambar **??** diperoleh dari program berikut ini.

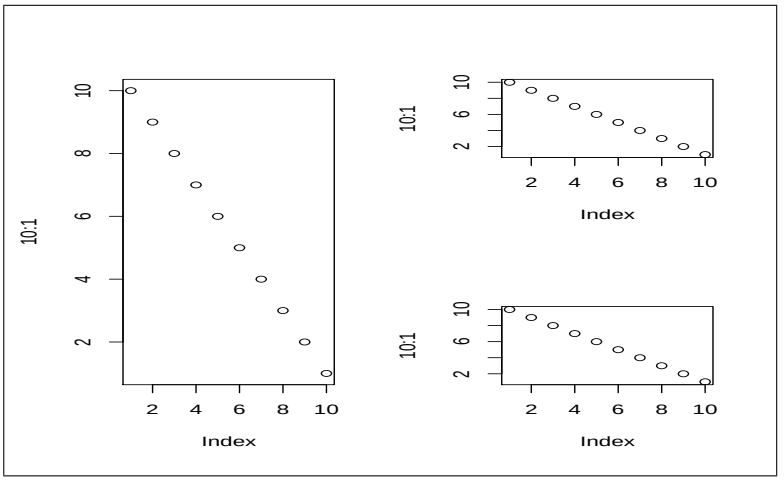
```
split.screen(c(1,2))
split.screen(c(2,1), screen = 2)
screen(1)
plot(10:1)
screen(3)
plot(10:1)
screen(4)
plot(10:1)
```



Gambar 2.5: *Lay Out* Multi Grafik Kompleks Lain

2.3.6 Aneka Rupa Perintah R

Di samping fungsi-fungsi yang berhubungan dengan penanganan file seperti membaca file skrip, menyimpan file keluaran, mencetak ko-



Gambar 2.6: *Lay Out* Multi Grafik Kompleks Lain

mentar dan variabel, ada juga fungsi lain di antaranya adalah seperti yang ada pada Tabel 2.6.

Contoh 2.7. Misalkan kita ingin mencetak keluaran yang sekaligus memuat komentar atau nama beserta nilainya, seperti "Jika $x = 2$ dan $y = 3$ maka hasil kali x dengan y adalah 6. Skrip untuk program ini adalah sebagai berikut.

```
x<-2
y<-3
cat("Hasil kali x dengan y adalah",x*y)
```

Contoh 2.8. Untuk menjalankan contoh yang ada pada fungsi *linear model* `lm()` maka perintahnya adalah

```
example(lm)
```

dan sebagian hasil luaran yang terjadi adalah:

```
> example(lm)

lm> ctl <- c(4.17, 5.58, 5.18, 6.11, 4.5, 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> anova(lm.D9 <- lm(weight ~ group))
Analysis of Variance Table

Response: weight
          Df Sum Sq Mean Sq F value Pr(>F)
group      1 0.6882  0.6882  1.4191  0.249
Residuals 18 8.7293  0.4850

lm> summary(lm.D90 <- lm(weight ~ group - 1))

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
```

Tabel 2.6: Aneka Rupa Fungsi R. Fungsi-fungsi ini bermanfaat untuk menangani file, skrip dan luaran

No	Tujuan	Perintah R	Keterangan
1	membaca tanpa membuka file	`source("namafile")`	
2	mengarahkan penulisan hasil ke file	`sink("nama file")`	
3	mencetak hasil (variabel)	`print(variabel)`	
4	mencetak teks	`cat("teks")`	
5	mencetak garis baru	`cat("\n")`	
6	mendefinisikan variabel, konstanta atau fungsi (berfungsi sebagai "=" dalam matematika)	`nama <- definisi`	
7	membaca tabel	`read.table("namatabel")`	
8	mengambil bagian dari kesatuan(objek)	`objek\$bagian`	
9	menulis komentar yang tidak dieksekusi R	`# komentar`	
10	mengatur tugas mendasar komputer	`options()`	
11	memeriksa file atau menjalankan file demo	`demo(NamaFile)`	pustaka harus diaktifkan
12	menjalankan file contoh pustaka	`example(NamaFile)`	pustaka harus diaktifkan

2.4 Menulis Program pada R

Pada bagian ini akan dibahas secara lebih detail cara- cara mendefinisikan fungsi, maupun membuat skrip program.

2.4.1 Komponen-Komponen Program

Program adalah sekumpulan perintah yang menjadi suatu kesatuan yang harus dikerjakan oleh komputer. Program biasanya ditulis untuk mengerjakan pekerjaan sejenis untuk keperluan jangka panjang dan bisa dimanfaatkan oleh orang lain yang tidak terlibat dalam pembuatan program tersebut. Oleh karena itu penulisan program ini harus memenuhi beberapa persyaratan. Naskah dari suatu pemrograman biasa disebut *listing* atau *script* dari program tersebut. Komponen- komponen pemrograman yang besar sebaiknya memuat hal-hal berikut.

1. **Seting Umum.** Bagian ini mengatur hal yang sangat mendasar dari komputer, seperti alokasi memori, jumlah digit dari luaran, nama file dan sebagainya. Dalam R, perintah ini biasanya dikerjakan dengan perintah `options`. Untuk jelasnya dapat dilihat dari dokumentasi perintah `options` ini.

   ```
   options(argumen1, argument2,...,.....);
   ```

   Beberapa pilihan yang dapat diatur adalah:

   - `echo=` T atau F, jika T berarti setiap ekspresi yang diekseksusi atau dievaluasi akan ditulis dulu (*echoed*) sebelum dievaluasi.
   - `digits= angka` yang menunjukkan banyaknya angka penting yang akan dicetak di *print out.*
   - `object.size= 1e+08` atau lebih yang diperlukan untuk program yang memerlukan memori besar (misalnya dalam simulasi).

2. **Definisi fungsi.** Apabila dalam program itu diperlukan fungsi yang didefinisikan sendiri, sebaiknya fungsi ini dikelompokkan dan didefinisikan pada bagian awal;

```
fungsi1<-function(){}
fungsi2<-function(){}
```

3. **Data dan Inisiasi.** Bagian berikut skrip biasanya berisi pengambilan data, baik yang dilakukan secara simulasi maupun dengan menggunakan data riil yang dimpor dari berbagai program yang tersedia. Data riil harus diaktifkan atau diset sebelum dianalisis.

   Untuk data simulasi sebelum variabel dan konstanta bisa dipergunakan harus ditetapkan nilai- nilai awalnya. Nilai awal (inisiasi) dari variabel-variabel ini juga sebaiknya diberikan pada bagian tersendiri secara mengelompok untuk memudahkan pemeriksaan atau perubahan.

4. **Program inti.** Bagian ini memuat inti dari pemrograman (looping, perhitungan- perhitungan matematika/ statistika dan lain -lainnya).

5. **Penutup.** Bagian penutup bisa berisi rangkuman dari hasil-hasil yang diperoleh dalam pemrograman tersebut. Biasanya bagian ini berisi perintah menyimpan file, mentabulasi hasil, atau membuat grafik.

Contoh 2.9. Contoh pemrograman yang lengkap beserta komponen-komponennya.

```
# namafile
options(echo=F,digits=4)
#fungsi
f1<-function(p1,p2){
```

```
     ekspresi1
     ekspresi2
     hasil akhir
    }
#inisiasi
p1<-5
p2<-15 #Program utama
#loop luar
for(i in 1:n){
        while(kon1){
        #loop dalam
            tugas1
    }
}
#Penutup Print(hasil.akhir)
write(t(x),file="x.byrows",ncol=ncol(x))
```

2.4.2 Langkah- langkah Penting dalam Penulisan Program

Program yang baik adalah program yang memenuhi beberapa persyaratan di antaranya adalah seperti berikut ini.

1. Mengerjakan dengan benar apa yang mestinya dikerjakan. Untuk program yang dikonsumsi sendiri atau bagi pemula, syarat yang paling utama adalah program yang dibuat harus mengerjakan dan memberi hasil yang benar.

2. Alur logika dan matematikanya benar dan mudah diikuti. Untuk itu, sebelum menulis skrip suatu pemrograman dan yakinkan bahwa baik persamaan-persamaan maupun fungsi-fungsi matematika yang akan digunakan sudah benar/valid. Selain itu langkah yang lebih rinci berupa algoritma dari apa yang akan dikerjakan oleh komputer sudah siap sebelum memulai menulis skrip dan yakinkan bahwa algoritma ini sudah benar. Algoritma ini bisa juga dibuat secara lebih eksplisit berupa diagram alir (*flow chart.*)

3. Mudah direvisi. Apa yang akan dikerjakan komputer seharusnya dapat dilacak dengan mudah sehingga kalau ada kesalahan juga mudah direvisi. Penelusuran atau revisi diharapkan tetap bisa dilakukan dengan mudah, meskipun suatu skrip program baru dibuka lagi setelah beberapa minggu, bulan, atau tahun.

4. Efektif dan efisien dalam memanfaatkan memori dan hardisk. Sebenarnya masalah efisiensi penggunaan memori dan kecepatan tidak terlalu penting bagi pemula, karena kecepatan dan penggunaan memori ini secara umum dapat teratasi dengan semakin canggihnya piranti keras komputer sekarang sehingga masalah ini tidak akan terlalu menjadi hambatan.

5. Mudah dimanfatkan. Syarat terakhir penting terutama kalau program yang dibuat juga disediakan bagi orang lain yang mungkin tidak banyak memahami pemrograman.

Serangkaian perintah atau fungsi-fungsi matematika yang sering dipergunakan sebaiknya didefinisikan dalam bentuk fungsi R. Identifikasi fungsi-fungsi dan variabel yang akan diperlukan dan didefinisikan di bagian awal. Usahakan memanfaatkan sebanyak mungkin fungsi-fungsi internal yang sudah ada dalam paket (dalam hal ini R). Ini penting agar penggunaan memori efisien mengingat selain karena sifatnya yang internal fungsi-fungsi ini telah teruji kemampuannya. Buatlah program dalam ukuran sedang dan program yang besar sebaiknya dipecah-pecah menjadi beberapa modul/ subprogram yang hanya dipanggil kalau diperlukan. Berilah komentar atau keterangan pada setiap fungsi yang didefinisikan, demikian juga pada setiap langkah penting. Hal ini akan membantu banyak dalam pemahaman dan proses perbaikan/revisi terutama jika program yang dibuat dibuka lagi setelah kurun waktu yang agak lama.

Khusus untuk R, sedapat mungkin hindarkan atau kurangi penggunaan *loop* terutama *loop* `for(){}` yang tidak perlu, kecuali tujuannya memang untuk mendemonstrasikan kerja dari *loop* tersebut. Untuk R, lebih efisien digunakan perhitungan vektor dari pada *loop*.

Karenanya, khususnya bagi pemrogram yang sudah berpengalaman, sedapat mungkin gunakan perhitungan secara vektor. Jika *loop* digunakan, pada tahap awal, gunakan kriteria konvergensi yang agak kasar/besar, setelah program berfungsi dengan baik, kriteria konvergensi dapat diatur sesuai kebutuhan. Skrip *loop* selain dikelompokkan dengan menggunakan kurung kurawal, juga perlu di-*indent* sesuai dengan tingkatnya. Berikut ini diberikan contoh program yang belum memerlukan adanya fungsi.

Contoh 2.10. Kita ingin mengilustrasikan hubungan antara mean populasi dengan rata- rata sampel berdasarkan ukuran sampelnya.

Untuk tujuan itu kita harus membuat *loop* pembangkitan data random untuk berbagai ukuran sampel, misalnya dari 10 sampai 100 dengan kenaikan 10. Untuk itu kita akan menempuh beberapa langkah berikut ini.

1. Mendefinisikan matriks yang terdiri atas 2 kolom dan 10 baris, untuk menampung mean sampel dari 10 macam ukuran sampel. Kolom pertama bisa diisi ukuran sampel dan kolom kedua berisi besarnya mean masing- masing sampel. Sebagai inisiasi, kita bisa mendefinisikan seluruh elemen matriks sama dengan 0.

   ```
   m<-matrix(0,10,2)
   ```

2. Membuat *loop* yang membuat pembangkitan data diulang mulai dari ukuran sampel 10 sampai dengan 100. Untuk ini ada beberapa alternatif.

   (a) Dengan menggunakan indeks $i = 1...10$ ditentukan ukuran sampelnya adalah $i \times 10$

   ```
   for(i in 1:10){ n.sampel<-i*10 }
   ```

   (b) Dengan menggunakan batasan maksimum 100, sedangkan `n.sampel` mengalami kenaikan 10 mulai dari 10

```
n.sampel<-10
while(n.sample<100){
      n.sampel<-n.sampel+10 }
```

3. Selanjutnya pada *loop* tadi dapat disisipkan perintah membangkitkan data normal dengan mean 50 dan variansi 10, $X \sim N(50, 10)$. Hal ini bisa juga dilakukan dengan membangkitkan data $Z \sim N(0, 1)$ selanjutnya ditransformasi dengan $X = \sigma Z + \mu$. Pada bagian/tahap yang sama kita menghitung mean dari X serta menaruh hasilnya pada matriks **m** pada baris dan kolom yang bersesuaian

```
Z<-rnorm(n.sample)
X<-sqrt(10)*Z+50
m[i,1]<-n.sampel
m[i,2]<-mean(X)
```

4. Selanjutnya, setelah *loop* berakhir, kita bisa mencetak grafiknya. Grafik sederhana dapat dibuat dengan menggunakan perintah

   `plot(m[,1],m[,2],type='l').`

   `m[,k]` menunjukkan seluruh baris pada kolom k sedangkan `m[i,]` menunjukkan baris kedua untuk seluruh kolom. Perintah yang lebih lengkap dapat dilakukan dengan memberikan judul dan label sumbu koordinat, atau bahkan warna.

```
plot(m[,1],m[,2],type='l',xlab='Sumbu X',ylab='Sumbu Y',
      main='Judul Grafik')
```

   Secara keseluruhan kita mempunyai skrip berikut.

```
n<-10
m<-matrix(0,n,2)
for(i in 1:n){
```

```
        n.sampel<-i*10
        Z<-rnorm(n.sample)
        X<-sqrt(10)*Z+50
        m[i,1]<-n.sampel
        m[i,2]<-mean(X)
        }
  plot(m[,1],m[,2],type='l',xlab='Sumbu X',ylab='Sumbu
        Y',main='Judul Grafik')
```

Jika petunjuk di atas diikuti dengan benar maka kita akan mendapat grafik seperti pada Gambar 2.7 pada halaman 82.

Contoh 2.11. Misalkan kita ingin membuat grafik dari persamaan matematika berbentuk persamaan parametriks seperti berikut.

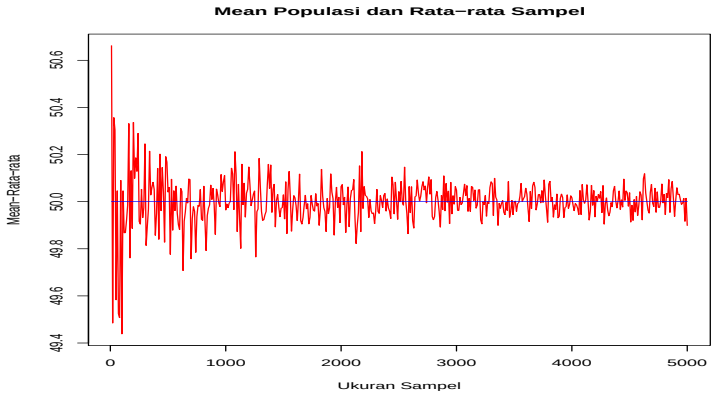
$$(x, y) = \begin{cases} x = \phi_1(t) \\ y = \phi_2(t) \end{cases}$$

Salah satu program yang bisa dibuat adalah seperti berikut dengan hasil seperti pada Gambar 2.8.

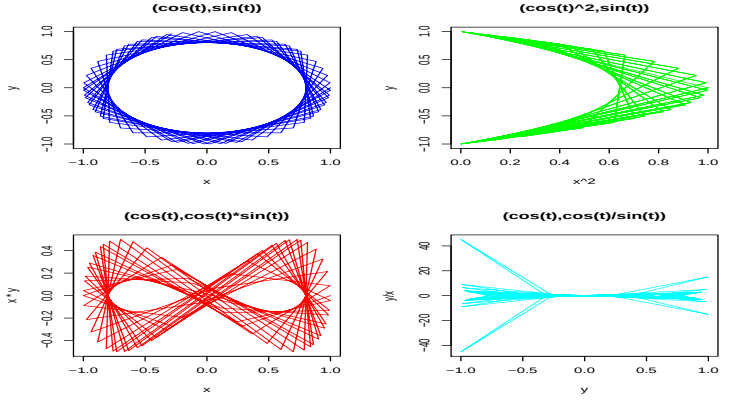
```
par(mfrow=c(2,2))
t<-seq(0,360,5)
y<-sin(t)
x<-cos(t)
plot(x,y,type='l',col='blue',main='(cos(t),sin(t))')
plot(x^2,y,type='l',col='green',main='(cos(t)^2,sin(t))')
plot(x,x*y,type='l',col='red',main='(cos(t),cos(t)*sin(t))')
plot(y,y/x,type='l',col='cyan',main='(cos(t),cos(t)/sin(t))')
```

2.4.3 Mendefinisikan Fungsi dalam R

Perintah panjang (terutama terkait dengan rumus-rumus matematika) dan sering dievaluasi dengan nilai berbeda dapat ditulis sebagai



Gambar 2.7: Ilustrasi Simulasi Mean Populasi dan Rata-rata Sampel



Gambar 2.8: Contoh Grafik Fungsi Parametrik Dimensi Dua

fungsi. Berikut akan dibahas secara lebih rinci cara-cara mendefinisikan fungsi sesuai petunjuk yang telah diberikan pada bagian sebelumnya. Sebagaimana telah dibicarakan sebelumnya bahwa fungsi dalam R memiliki Parameter dengan struktur berikut:

```
nama.fungsi<-function(parameter1, parameter2){
                    ekspresi1,
                    ....
                    fungsi1
                    ....
                    hasil akhir (returned value)
                  }
```

Selain variabel atau konstanta yang didefinisikan secara internal (seperti pi $=\pi = 3.1415$ ...), maka semua variabel atau konstanta yang dipergunakan dalam definisi harus diperlakukan sebagai parameter. Dalam suatu fungsi kita dibenarkan memanggil fungsi yang lain. Di antara sekian banyak perhitungan yang dilakukan dalam suatu fungsi, maka harus ditegaskan hasil yang akan ditampilkan sebagai hasil akhir eksekusi fungsi. Hasil ini disebut sebagai *returned value* yang dilakukan dengan memanggil kembali hasil yang telah dihitung. Pada dasarnya fungsi R adalah translasi dari fungsi matematika ke dalam bahasa R. Hal ini akan terlihat jelas dari beberapa contoh yang diberikan.

```
fungsi.f<-function(x){x*sin(x)}
fungsi.g<-function(x,y){
       log(x)+fungsi.f(x)+exp(y)
```

Fungsi g ini, selain mempunyai parameter yang lebih banyak dari f juga memanggil fungsi f dan ini hal yang bisa dilakukan dalam mendefinisikan fungsi-fungsi R. Agar fungsi g berfungsi dengan baik maka fungsi f harus sudah didefinisikan terlebih dahulu sebelum mendefinisikan fungsi g.

Menghitung nilai fungsi dan akar-akar persamaan kuadrat

Kita dapat mendefinisikan fungsi untuk menghitung nilai fungsi persamaan tersebut untuk berbagai nilai konstanta dan variabel. Hal ini bermanfaat dalam membuat grafik dari persamaan tersebut.

Contoh 2.12. Fungsi untuk menghitung nilai fungsi kuadrat

```
f<-function(a,b,c,x){
    a*x^2+b^x+c
    }
```

Fungsi di atas dapat dievaluasi untuk nilai-nilai konstanta baik a, b, c maupun variabel x yang berbeda-beda. Selain menghitung nilai fungsi, kita juga dapat membuat program untuk menghitung akar-akar persamaan kuadrat.

Contoh 2.13. Misalkan kita ingin membuat program/fungsi R dari rumus abc untuk menghitung akar-akar persamaan kuadrat $f(x) = ax^2 + bx + c = 0$ untuk berbagai nilai a, b, c.

Berikut ini adalah langkah-langkah yang bisa ditempuh.

1. Yakinkan bahwa fungsi yang akan diprogramkan secara matematis sudah valid. Untuk fungsi yang merupakan rumus abc bentuk matematikanya adalah

$$x_{12} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Dalam format fungsi R, koefisien a, b, dan c diperlakukan sebagai parameter fungsi. Akar-akar x_1 dan x_2 didefinisikan sesuai dengan rumus abc. Ini adalah bagian algoritma yang harus dimiliki sebelum kita menulis skrip pemrograman. Untuk rumus abc, kita tahu hasil secara matematis untuk persamaan kuadrat fungsi matematikanya ditunjukkan oleh persamaan di atas.

2. Langkah selanjutnya adalah menerjemahkan rumus atau komponen-komponennya ke dalam bahasa R. Mengingat ada dua nilai yang dihasilkan yaitu x_1 dan x_2, maka variabel x_1 dan x_2 harus dikombinasikan melalui perintah `cbind(x1,x2)` jika dikelompokkan menjadi vektor baris, atau `rbind(x1,x2)` jika dikelompokkan menjadi vektor kolom. Vektor ini sekaligus menjadi hasil yang ditampilkan (*returned value*).

```
fungsi.abc<-function(a,b,c){
           x1<-(-b+sqrt(b^2-4*a*c))/2*a
           x2<-(-b-sqrt(b^2-4*a*c))/2*a
           cbind(x1,x2)
        }
```

2.4.4 Mengevaluasi Nilai Fungsi

Setelah fungsi terbentuk kita bisa melakukan evaluasi. Kita bisa mengevaluasi fungsi tersebut untuk suatu nilai a, b, c tertentu. Berikut adalah beberapa hasil yang diperoleh dari hasil evaluasi fungsi yang dibuat.

```
> fungsi.abc(1,-5,6)
     x1 x2
[1,]  3  2
> fungsi.abc(1,0,-4)
     x1 x2
[1,]  2 -2
```

Jika pembentukan vektor mwenggunakan perintah `rbind(x1,x2)` maka kita akan memperoleh hasil seperi berikut

```
> fungsi.abc(1,-5,6)
   [,1]
x1    3
```

```
x2    2
> fungsi.abc(1,0,-4)
   [,1]
x1    2
x2   -2
```

Selanjutnya apabila akar-akar yang terjadi adalah imajiner, maka komentar R yang muncul adalah

```
> fungsi.abc(1,0,4)
   [,1]
x1   NA
x2   NA
```

NA berarti tidak ada hasil yang tersedia.

Jika dalam pemanggilan fungsi hanya diberi nilai parameter, R akan mendistribusikan nilai parameter sesuai dengan urutan parameternya. Jika dalam pemanggilan identitas parameternya telah ditentukan, urutan penempatan dapat berbeda. Pemanggilan berikut menghasilkan hasil yang sama

```
fungsi.abc(1,-2,-15)
fungsi.abc(a=1,c=-15,b=-2)
fungsi.abc(c=-15,b=-2,a=1)
```

```
> fungsi.abc(1,-2,-15)
     x1 x2
[1,]  5 -3
> fungsi.abc(a=1,c=-15,b=-2)
     x1 x2
[1,]  5 -3
> fungsi.abc(c=-15,b=-2,a=1)
     x1 x2
[1,]  5 -3
```

Untuk menjadikan program ini lebih komunikatif, maka kita perlu memberikan beberapa pesan yang lebih dipahami, kalau diskriminan dari rumus abc tersebut kurang dari 0. Untuk itu, kita perlu menggunakan perintah- perintah kontrol seperti `if, if else`. Fungsi di atas dapat dikembangkan menjadi seperti berikut ini.

```
# contoh fungsi untuk menghitung akar- akar persamaan
# kuadrat dengan rumus abc
# dalam fungsi ini a tidak boleh sama dengan 0.
fungsi.abc<-function(a,b,c){
        if(a==0){stop("\na harus <>0")}
        D<-b^2-4*a*c # diskriminan
         if(D>=0){
         x1<-(-b+sqrt(b^2-4*a*c))/2*a
         x2<-(-b-sqrt(b^2-4*a*c))/2*a
         rbind(x1,x2) # membuat vektor akar- akar
            }
         else{cat("\n Akar- akar imaginer")}
         # pesan kalau D<0
}
```

Setiap kali kita melakukan perubahan atau revisi pada fungsi, maka fungsi itu harus di eksekusi dulu supaya perbaikannya menjadi efektif tercatat dalam R. Jika tidak, maka R tetap akan memanggil fungsi yang belum diperbaiki. Beberapa hasil yang diperoleh dari eksekusi fungsi yang telah dimodifikasi untuk berbagai nilai parameter a, b, c adalah sebagai berikut.

```
> fungsi.abc(1,0,-4)
   [,1]
x1    2
x2   -2

> fungsi.abc(1,5,-6)
   [,1]
x1    1
x2   -6
```

```
> fungsi.abc(1,5,6)
   [,1]
x1   -2
x2   -3

> fungsi.abc(1,0,4)
  Akar- akar imaginer
>
> fungsi.abc(0,0,4)
   Error in fungsi.abc(0, 0, 4):
   a harus <>0
```

Prinsip pemnulisan fungsi di atas dapat dikembangkan untuk penulisan fungsi yang lebih kompleks, misalnya untuk analisis data, parameter fungsi dapat berupa: nama data, formula, distribusi data, danlain-lainnya).

2.4.5 Mengemas Keluaran Fungsi

Untuk fungsi yang lebih kompleks, misalnya dalam analisis data, banyak hal yang dikerjakan dalam fungsi. Hal-hal yang dikerjakan dalam fungsi dapat dikemas menjadi satu kesatuan keluaran. Misalnya dalam hal persamaan kuadrat, selain perhitungan akar-akar, ada perhitungan diskriminan. Semua perhitungan ini dapat dikemas menjadi suatu daftar atau `list()`. Program fungsi.abc di atas selanjutnya dapat dimodifikasi dengan menambahkan beberapa baris program berikut.

```
 x12<-rbind(x1,x2) # membuat vektor akar- akar
         hasil<-list()
         hasil<-list(akar1=x12[1],akar2=x12[2],disk=D)
```

Pemanggilan fungsi dengan evaluasi nilai tertentu akan menghasilkan learan yang dilabel sebagai akar dan det.

```
$akar1
[1] 1

$akar2
```

```
[1] -3

$disk
[1] 16
```

Jika pemanggilan disimpan dalam sustu objek, misalkan x, maka subkomponen x dapat diperiksa melalui `names(x)` dan `print(x)`

```
> x<-fungsi.abc(1,2,-3)
> names(x)
[1] "akar1" "akar2" "disk"
> x$akar1
[1] 1
> x$akar2
[1] -3
> x$disk
[1] 16
```

Dalam analisis data, keluaran fungsi dapat berupa: nilai hitung statistik (nilai t, z, Fdan sejenisnya), kesalahan baku dari masing-masing statistik, nilai peluang $p - values$, dan informasi lain yang dianggapperlu. Semua keluaran dan hasil perhitungan yang dlakukan dapat dikemas dalam daftar keluaran (`list()`) denganmenggunakan identitas yang mudah untuk dipanggil.

Pada contoh berikut kita membuat fungsi yang menghitung statistik sampel (rata-rata, deviasi baku, maksimum, minimum, median dan sejenisnya). Semua keluaran ini dapat dikemas dalam `(list()` keluaran.

```
fs.stat.norm<-function(n,mu,sd){
              x<-rnorm(n,mu,sd)
              mn<-min(x)
              mx<-max(x)
              vr<-var(x)
              md<-median(x)
              rt<-mean(x)
              rks<-list()
              rks<-list("n"=n,"rata-rata"=rt, "min"=mn, "maks"=mx,
              "variansi"=vr, "median"=md)
              }
```

Eksekusi berikut menghasilkan

```
> y<-fs.stat.norm(1000,50,5)
> names(y)
[1] "n"          "rata-rata" "min"        "maks"       "variansi"
    "median"
>print(y)
$n
[1] 1000

$`rata-rata`
[1] 50.08186

$min
[1] 36.48708

$maks
[1] 65.65245

$variansi
[1] 26.41415

$median
[1] 50.21028
```

2.4.6 Menghindarkan *Loop*

R lebih efisien bekerja menggunakan vektor dibandingkan dengan menggunakan *loop* `if then` seperti di diuraikan sebelumnya. Burn (1998) mengilustrasikan bahwa dalam kondisi tertentu, penggunaan vektor dapat mempercepat eksekusi program secara dramatis. Ada dua kondisi utama yang perlu diperhatikan untuk mengganti *loop* dengan vektor, yaitu:(i) untuk kondisi dimana pada setiap putaran memiliki panjang elemen yang sama dapat dipergunakan matriks, (ii) untuk kondisi dimana pada setiap putaran memiliki panjang elemen yang tidak sama dapat dipergunakan fungsi `apply`

Penggunaan Matriks/Vektor

Untuk *loop* yang sederhana, dapat digantikan dengan menggunakan matriks dengan cara

1. mengemas indeks
   dalam satu vektor (atau kolom suatu matriks)
2. melakukan operasi scara keseluruhan pada vektor tersebut.

Berikut adalah adalah contoh *loop* yang diganti dengan penggunaan vektor/matriks. Misalkankita ingin membuat program untuk menghitung seperti ditunjukkan oleh tabel berikut

n	n^2	n^3	$\log n$
10			
20			
30			
40			
...			
1000			

Dengan menggunakan loop

```
for (i  in 1:100){
    n<-i*10
    t1<-n^2
    t2<-n^3
    t3<-log(n)
    }
```

Dengan menggunakan matriks/vektor

```
ind<-seq(1,100,1)
n<-ind*10
t1<-n^2
t2<-n^3
t3<-log(n)
```

Semakin banyak putaran *loop* semakin terasa beda kecepatan penggunaannya dengan matriks atau vektor.

Penggunaan Aplly

Misalkan kita ingin mengilustrasikan hubungan antara besarnya (ukuran) sampel dengan kedekatan rata-rata dengan mean populasi. Berarti pada setiap putaran *loop* kita harus membangkitkan data dengan ukuran sampel yang berbeda, lalu menghitung rata-ratanya. Persoalan ini sulit kalau dikerjakan dengan menggunakan matriks atau vektor seperti diatas. R menyediakan fungsi keluarga `apply`, yaitu `apply, tapply, lapply` untuk mengerjakan loop yang tidak bisa dikerjakan pemggunakan matriks biasa. Sintaks penggunaan `apply` adalah

```
apply(matriks,p,fungsi)
```

Dengan

matriks adalah matriks yang akandijadikan sebagai acuan indeks dalam mengerjakan *loop*;

p adalah posisi yang dijadikan sebagai acuan yaitu:1 jika berdasarkan baris dan 2 jika berdasarkan kolom;

fungsi adalah fungsi yang mengaturtugasyangharus dikerjakan pada setiap putaran

Misalkan kita ingin menghitung rata-rata sampel dari berbagai sampel yang berasal dari populasi berdistribusi normal, $N(50, 10)$, yang ukurannya semakin besar (10,20,30, ...,1000).

n	$\mu = 50$	$\sigma^2 = 25$	X
10			
20			
30			
40			
...			
1000			

Tugas yang harus dikerjakan komputer pada setiap putaran adalah:(i) membangkitkan data dan (ii) menghitung rata-ratanya. Tugas ini dapat didefinisikan dalam bentuk fungsi berikut:

```
myfun<-function(x){
```

```
y<-rnorm(x)*sqrt(10)+50
mean(y)
}
```

Program lengkap dengan presentasi grafik yang menunjukkan hubungan antara ukuran sampel dengan rata-sata sampel adalah

1. Dengan *loop*

```
n<-100
m<-matrix(0,n,2)
for(i in 1:n){
    n<-i*10
    y<-rnorm(n)*sqrt(10)+50
    m[i,1]<-n
    m[i,2]<-mean(y)
}
plot(m[,1],m[,2],type='l',xlab='N',ylab='Rata-rata',main='Judul Grafik')
```

2. Dengan*apply*

```
n<-100
m<-matrix(0,n,1)
m[,1]<-10*seq(1,n,1)
myfun<-function(x){
       y<-rnorm(x)*sqrt(10)+50
       mean(y)
       }
m2<-apply(m,1,myfun)

plot(m,m2,type='l',xlab='N',ylab='Rata-rata',main='Judul Grafik')
```

Tugas yang harus dikerjakan komputer pada setiap putaran dapat dibuat lebih kompleks misalnya menghitung ringkasan statistik sampel dan melaporkan banyaknya sampel

```
n<-10
m[,1]<-20*seq(1,n,1)

myfun<-function(x){
```

```
y<-rnorm(x)*sqrt(10)+50
c(x,summary(y))
}
```

Keluaran yang dihasilkan adalah sebagai berikut

```
         [,1]  [,2]  [,3]  [,4]   [,5]   [,6]   [,7]   [,8]
 n      20.00 40.00 60.00 80.00 100.00 120.00 140.00 160.00
Min.    46.40 41.46 41.37 43.89  40.46  41.78  43.20  40.75
1st Qu. 48.36 48.42 47.98 47.74  48.55  46.87  47.86  47.97
Median  49.20 50.10 50.58 49.47  50.02  49.21  50.03  50.01
Mean    49.95 50.39 50.16 49.64  50.14  49.57  50.09  49.95
3rd Qu. 51.78 52.41 52.41 51.47  52.46  52.17  52.12  52.16
Max.    54.42 57.16 56.70 56.04  55.98  58.86  58.12  59.60
>
```

2.4.7 Menghitung Akar-Akar Persamaan dengan Metode Numerik

Untuk persamaan selain persamaan linear dan kuadrat, biasanya penyelesaian ditempuh dengan menggunakan metode numerik. Salah satu metode numerik yang banyak dipergunakan dalan statistika adalah Metode Newton-Raphson. Dalam statistika kita sering berhubungan dengan fungsi-fungsi likelihood yang akan dicari maksimumnya. Mencari maksimum dari suatu fungsi pada dasarnya sama dengan menyelesaikan persamaan dari turunan pertamanya. Pada umumnya persamaan seperti ini, dalam statistika jarang mempunyai penyelesaian analitik, sehingga harus dicari dengan metode numerik. Penyelesaian numerik suatu persamaan dicari melalui proses iterasi yaitu proses mengerjakan sekelompok operasi hitung yang semakin lama menghasilkan nilai yang semakin dekat dengan hasil yang sebenarnya, kecuali jika persamaan itu tidak memiliki jawaban. Secara umum bentuk iterasi Newton-Raphson yang dipergunakan untuk menyelesaikan persamaan $f(x) = 0$ adalah

$$x_1 = x_0 - \frac{f(x)}{f'(x)} \tag{2.1}$$

dengan $f'(x) = df/dx$. Jika yang dicari adalah nilai x yang menyebabkan fungsi itu mencapai maksimum/ minimum, maka iterasi Newton-Raphson ini dimodifikasi menjadi

$$x_1 = x_0 - \frac{f'(x)}{f''(x)} \tag{2.2}$$

Hal ini sesuai dengan penjelasan sebelumnya bahwa mencari titik maksimum suatu fungsi sama halnya mencari penyelesaian dari fungsi turunan pertamanya.

Contoh 2.14. Misalkan kita ingin mencari titik maksimum atau minimum dari fungsi:

$$f(x) = \sin(x) + x^2 + 2x \text{ pada } -10 \leq x \leq 10. \tag{2.3}$$

Langkah-langkah untuk membuat program dalam mencari titik maksimum fungsi tersebut dengan menggunakan Metode Newton-Raphson adalah seperti berikut.

1. **Menentukan hasil-hasil matematika.** Dalam hal ini, fungsi yang diperlukan adalah turunan pertama dan kedua dari bagian persamaan yang dicari akar-akarnya. Dengan menggunakan berbagai tehnik dalam diferensial integral diperoleh hasil sebagai berikut. Fungsi turunan pertama dan kedua:

$$f'(x) = \cos(x) + 2x + 2 \tag{2.4}$$
$$f''(x) = -\sin(x) + 2 \tag{2.5}$$

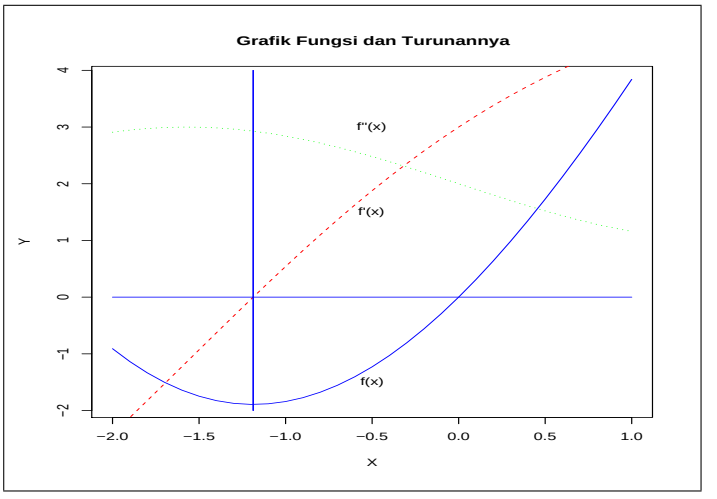
2. **Menulis skrip fungsi.** Skrip lengkap fungsi untuk menghitung titik minimum adalah:

```
d<-2
eps<-0.01
x0<-0
f.f1<-function(x){
```

```
    cos(x)+2*x+2}
f.f2<-function(x){
    -sin(x)+2}
it<-0
cat("\n Iterasi: ")
while(d>eps){
it<-it+1
x1<-x0-f.f1(x0)/f.f2(x0)
d<-abs(x1-x0)
x0<-x1
cat(" ",it)}

print(x0)

x<-seq(-2,1,0.1)
y<-sin(x)+x^2+2*x
f1<-cos(x)+2*x+2
f2<--sin(x)+2
plot(x,y,type='l',main='Grafik Fungsi dan Turunannya',xlab='X',
     ylab='Y')
lines(x,f1,lty=2)
lines(x,f2,lty=3)
lines(x,0*x)

ya<-seq(-2,4,0.1)
xa<-0*ya+x0
lines(xa,ya,lty=3)
```

Secara geometris dapat diilustrasikan/diperiksa kebenaran antara fungsi, turunannya dan nilai maksimum atau minimum dengan memeriksa grafiknya. Ilustrasi pada Gambar 2.9 halaman 97 menunjukkan hubungan sebagai berikut:

- saat $f(x)$ mencapai nilai minimum/maksimum, $f'(x) = 0$;
- nilai merupakan minimum jika $f''(x) > 0$, sebaliknya merupakan maksimum jika $f''(x) < 0$.



Gambar 2.9: Ilustrasi Maksimum/ Minimum dengan Newton Raphson.

Contoh 2.15. Buat program untuk mencari titik maksimum/ minimum dari persamaan dengan menggunakan metode Newton-Raphson.

$$f(x) = x \exp\left(-\frac{x^2}{10}\right)$$

Dengan mengikuti langkah-langkah pemrograman sebelumnya, kita akan peroleh hasil dari tiap-tiap tahap sebagai berikut.

1. Fungsi-fungsi turunan

$$f'(x) = \left(-\frac{2x^2}{10} + 1\right) \exp\left(-\frac{x^2}{10}\right) \tag{2.6}$$

$$f''(x) = \left(\frac{4x^3}{100} - \frac{6x}{10}\right) \exp\left(-\frac{x^2}{10}\right) \tag{2.7}$$

2. Dalam fungsi R fungsi turunan tersebut dapat ditulis sebagai berikut.

```
f.tur.1<-function(x){
         d1<-(-2*x^2/10+1)*exp(-x^2/10)
         d1 }

f.tur.2<-function(x){
         d2<-(4*x^3/100-6*x/10)*exp(-x^2/10)
         d2
}
```

3. **Menentukan nilai awal.** Nilai awal dari x dapat ditentukan, misalnya $x_0 = 5$ atau $x_0 = -5$. Sedangkan $\Delta x = 10$ (delta.x) dan ϵ(crit) sebagai kriterium konvergensi dapat dipilih sekecil mungkin sesuai keparluan, misalnya 0.001. Sebelum program berjalan sebagaimana mestinya nilai ϵ dibuat agak besar, misalnya 0.5, sehingga dalam pengujian program tidak memakan waktu terlalu lama.

4. **Bagian utama**. Bagian utama dari program ini berisi:

```
# looping
no.it<-0 cat("x awal adalah ",x0,"\n")
while(delta.x>crit){
     x1<-x0-f.tur.1(x0)/f.tur.2(x0)
     delta.x<-abs(x1-x0)
     x0<-x1
     no.it<-noit+1 }
```

`no.it` hanyalah konter/pencacah untuk mendeteksi jumlah iterasi yang diperlukan. Pada bagian penutup kita dapat memerintahkan komputer untuk mencetak hasil.

```
cat("Fungsi akan memperoleh nilai max/min
     pada titik x=",x1,"\n")
cat("Tingkat ketelitian adalah ",delta.x,
```

```
        "dan banyaknya iterasi
         adalah ",no.it,"\n")
```

Selanjutnya hasil yang diperoleh dari mengeksekusi program yang dibuat dengan berbagai titik awal adalah

```
x awal = -4 Fungsi memperoleh nilai max/min
pada titik
x=-2.2444951635172
Tingkat ketelitian adalah  0.000934621250368472
Banyaknya iterasi adalah  53

x awal = 3 Fungsi akan memperoleh nilai max/min
pada titik
x=2.24448623841328
Tingkat ketelitian adalah
0.00093363323004958
Banyaknya iterasi adalah  44
```

Hasil di atas menunjukkan bahwa fungsi yang kita evaluasi memiliki dua titik dimana dia mencapai maksimum/ minimum. Ilustrasi akan lebih jelas pada saat kita menampilkan grafik fungsi tersebut.

Untuk menghitung akar-akar persamaan fungsi multivariabel (fungsi dua variabel atau lebih) prinsip yang kita gunakan sama, hanya kita bekerja dalam operasi vektor/ matriks ($\mathbf{v}$), vektor/matriks turunan pertama $\mathbf{D}$ dan turunan kedua dari fungsi tersebut adalah matriks Hessiannya ($\mathbf{H}$) dan bentuk iterasi Newton-Raphsonnya menjadi

$$\mathbf{v_1} = \mathbf{v_0} - \mathbf{H^{-1}D} \tag{2.8}$$

atau lebih lengkapnya

$$\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_p \end{pmatrix}_1 = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_p \end{pmatrix}_0 - \begin{pmatrix} \frac{\partial^2 f}{\partial v_1^2} & \frac{\partial^2 f}{\partial v_1 \partial v_2} & \cdots & \frac{\partial^2 f}{\partial v_1 \partial v_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial v_p \partial v_1} & \frac{\partial^2 f}{\partial v_p \partial v_2} & \cdots & \frac{\partial^2 f}{\partial v_p^2} \end{pmatrix}_0 \begin{pmatrix} \frac{\partial f}{\partial v_1} \\ \frac{\partial f}{\partial v_2} \\ \vdots \\ \frac{\partial f}{\partial v_p} \end{pmatrix}_0 \tag{2.9}$$

Misalkan kita ingin mencari titik maksimum atau minimum dari fungsi

$$f(x, y) = -x^2 + 2xy + 2x - 4y - 3y^2,$$

maka langkah yang harus kita tempuh adalah mencari fungsi turunan pertama dan kedua terhadap x dan y dari fungsi di atas. Hasil matematikanya adalah sebagai berikut.

Hasil-hasil matematika

$$\frac{\partial f}{\partial x} = 2x + 2y + 2 \tag{2.10}$$

$$\frac{\partial f}{\partial y} = 2x - 4 - 6y \tag{2.11}$$

$$\frac{\partial^2 f}{\partial x^2} = 2 \tag{2.12}$$

$$\frac{\partial^2 f}{\partial y^2} = -6 \tag{2.13}$$

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x} = 2 \tag{2.14}$$

Fungsi-fungsi R Dalam bahasa R fungsi dapat ditulis sebagai berikut.

```
fmv<-function(x,y){
    -x^2+2xy+2x-4y-3y^2}
df.dx<-function(x,y){
       -2*x+2*y+2}
df.dy<-function(x,y){
       2*x-6*y-4}
```

Fungsi $f(x, y)$ didefinisikan dalam R bermanfaat jika dibuat ilustrasi grafiknya (3 dimensi) baik dalam bentuk perspektif maupun konturnya. Hal ini sangat bermanfaat untuk mengilustrasikan adanya maksimum/minimum secara grafis atau numerik. Untuk turunan ke dua karena hanya berupa konstanta,

maka matriksnya langsung dapat didefinisikan

$$\mathbf{H} = \begin{pmatrix} 2 & 2 \\ 2 & -6 \end{pmatrix}$$

Dalam bahasa R dapat didefinisikan dengan

```
H<-matrix(c(-2,2,2,-6),2,2)
```

Inisiasi. Nilai awal variabel yang perlu ditetapkan terlebih dahulu adalah nilai awal $\mathbf{v_0}$, matriks $\mathbf{D}$, kriteria konvergensi.

```
v0<-matrix(0,2,1)
D<-matrix(0,2,1)
crit<-0.001
delta<-10
```

Program inti. Bagian ini terdiri atas loop Newton-Raphson dalam dua variabel.

```
while(delta>crit){
     D[1,1]<-df.dx(x,y)
     D[2,1]<-df.dy(x,y)
     v1<-v0-solve(H)%*%D
     delta<-max(abs(v0-v1))
     v0<-v1
     it<-it+1
     cat("It ",it,"V=",v1,"delta=",delta,"\n") }
print(v1)
```

Hasil yang diperoleh adalah

```
It  1
v =  0.499999999999998 -0.5;
delta= 10.5
It  2
v =  0.5
-0.5;
delta = 1.7763568394003e-015
> print(v1)
     [,1]
[1,]  0.5
[2,] -0.5
```

Jadi, program yang dibuat konvergen dalam dua iterasi. Dari beberapa contoh tadi jelas bagi kita bahwa pada dasarnya fungsi R adalah fungsi matematika yang didefinisikan dengan menggunakan bahasa R . Ini menunjukkan bahwa alur logika pemrograman dengan R secara alamiah sejalan dengan alur logika matematika.

2.5 Mendefinisikan Ulang Fungsi Internal

Secara teoritis fungsi-fungsi internal R dapat ditiru untuk, misalnya, didefinisikan ulang menggunakan bahasa Indonesia. Jika suatu fungsi dipanggil namanya tanpa memberi parameter, maka yang keluar adalah definisi fungsinya. Misalnya pemanggilan fungsi plot akan menghasilkan tampilan berikut yang merupakan definisi fungsi plot.

```
> plot # tanpa tanda kurung ()
function (x, y, ...)
     {
    if (is.null(attr(x, "class")) && is.function(x)) {
        nms <- names(list(...))
        if (missing(y))
            y <- {
```

```
                if (!"from" %in% nms)
                  0
                else if (!"to" %in% nms)
                  1
                else if (!"xlim" %in% nms)
                  NULL
            }
        if ("ylab" %in% nms)
            plot.function(x, y, ...)
        else plot.function(x, y,
            ylab = paste(deparse(substitute(x)),
            "(x)"), ...)
    }
    else UseMethod("plot")
     }
    }
```

Untuk mendefinisikan ulang fungsi `plot()` dengan nama `gambar()`, maka kita cukup mendefinisikan fungsi `gambar()` dengan menggunakan skrip definisi tersebut, lalu menjalankan skrip yang ada. Selanjutnya, kita bisa memanggil fungsi `gambar` sebagaimana memanggil fungsi `plot()`. Perlu diketahui bahwa untuk dokumentasi (bantuan dan lain-lain) perlu didefinisikan tersendiri.

```
gambar<- function (x, y, ...){
    if (is.null(attr(x, "class")) && is.function(x)) {
        nms <- names(list(...))
        if (missing(y))
            y <- {
                if (!"from" %in% nms)0
                else if (!"to" %in% nms )1
                else if (!"xlim" %in% nms) NULL
            }
        if ("ylab" %in% nms)
            plot.function(x, y, ...)
        else plot.function(x, y,
        ylab = paste(deparse(substitute(x)),
```

```
        "(x)"), ...)
}
else UseMethod("plot")
 }}
```

2.6 Memanfaatkan Paket Program R

Fungsi-fungsi kompleks untuk keperluan tertentu (misalnya regresi, grafik, atau analisis data lainnya) telah banyak dibuat dan dikemas orang dalam bentuk paket-paket (*packages* program R. Hampir semua paket program R didistribusikan secara cuma-cuma dan disediakan di situs R dan dapat dimanfaatkan bersama dengan program utama R. Paket-paket yang telah diinstal bersama R akan ditempatkan dalam direktory `library`, oleh karena itu paket yang telah diinstal lebih populer disebut sebagai *library* atau pustaka.

2.6.1 Mencari Informasi Paket Program R

Hal pertama dan terpenting yang dapat dilakukan dalam menggunakan program R adalah mendapatkan informasi tentang fasilitas yang tersedia. Dari tampilan sebelumnya dapat diketahui bahwa jika kita ingin mengetahui beberapa kemampuan dan fasilitas R dapat dilakukan dengan menggunakan salah satu alternatif berikut.

`help()` untuk mengetahui dokumentasi bantuan secara umum. Selanjutnya kita dapat memilih topik yang tersedia dan lebih mengkhususkan pencarian kita pada topik tersebut dengan menggunakan `help(topik)`.

`help.start()`. Untuk mengaktifkan dokumentasi dalam format html yang lebih interaktif. Pengguna selanjutnya dapat memilih topik yang tersedia seperti layaknya mencari informasi di internet.

`help.search("kata kunci")`. Jika kita ingin mendapat informasi dari suatu topik tertentu dapat juga kita masukkan suatu kata kunci melalui `help.search()`.

Misalkan kita ingin mencari informasi tentang analisis regresi, maka kata kunci yang mungkin bisa dimasukkan di antaranya adalah `regression, linear model`. Hasil pelacakan menghasilkan

```
>help.search("regression")

Help files with alias or concept or title matching
'regression' using fuzzy matching:

MCMClogit(MCMCpack)     Markov chain Monte Carlo for
                        Logistic    Regression
...
gam(mgcv)               Generalized Additive Models using
                        penalized
                        regression splines and GCV

gam.fit(mgcv) Generalized Additive Models fitting using
                        penalized regression splines
                        and GCV
...
ksmooth(stats)          Kernel Regression Smoother
lm(stats)               Fitting Linear Models
```

Sementara itu, pelacakan dengan menggunakan `linear models` menghasilkan

```
>help.serach("linear models")
Help files with alias or concept or title matching
'linear models' using fuzzy matching:
stdres(MASS)            Extract Standardized Residuals from
                        a Linear   Model
studres(MASS)           Extract Studentized Residuals from
                        a Linear Model
gls(nlme)               Fit Linear Model Using Generalized
                        Least Squares
...
```

Nama dalam kurung menunjukkan pustaka tempat perintah tersebut berada. Untuk memperoleh informasi lebih lanjut, yang perlu dilakukan adalah:

1. mengaktifkan pustaka bersangkutan dengan menggunakan `library(NamaPustaka)` dan
2. mencari informasi dengan menggunakan `help(topik)`.

Misalkan kita ingin mendapat informasi `gam(mgcv)` maka yang harus kita lakukan adalah menjalankan perintah berikut.

```
>library(mgcv)
>help(gam)
```

Sebagian dari dokumentasi fungsi ini diberikan pada keluaran berikut:

```
gam             package:mgcv             R Documentation

Generalized Additive Models using penalized regression
splines and GCV

Description:

Fits the specified  generalized additive model (GAM)
to data. Smooth terms are represented using penalized
regression splines with smoothing parameters selected
by GCV/UBRE or by regression splines with fixed
degrees of freedom (mixtures of the two are permitted).
Multi-dimensional smooths are available using penalized
thin plate regression splines (isotropic) or tensor
product splines (when an isotropic smooth is
inappropriate).
For more on specifying models see 'gam.models'.
```

```
For more on model selection see 'gam.selection'.
For large datasets see warnings. ...

Usage:

gam(formula,family=gaussian(),data=list(),
    weights=NULL,   subset=NULL,
    na.action,offset=NULL,control=gam.control(),
    scale=0,knots=NULL,
    sp=NULL,min.sp=NULL,H=NULL,gamma=1,
    fit=TRUE,G=NULL,...)
...
References:

Key References on this implementation:

Wood, S.N. (2000)  Modelling and Smoothing Parameter
     Estimation with Multiple  Quadratic Penalties.
     J.R.Statist.Soc.B 62(2):413-428

Wood, S.N. (2003) Thin plate regression splines.
     J.R.Statist.Soc.B      65(1):95-114

Wood (2001) mgcv:GAMs and Generalized Ridge
     Regression for R. R News 1(2):20-25
...

Example
     library(mgcv)
     set.seed(0)
     n<-400
     sig<-2
     x0 <- runif(n, 0, 1)
```

```
x1 <- runif(n, 0, 1)
x2 <- runif(n, 0, 1)
x3 <- runif(n, 0, 1)
f <- 2 * sin(pi * x0)
f <- f + exp(2 * x1) - 3.75887
f <- f+0.2*x2^11*(10*(1-x2))^6+10*(10*x2)^3
     *(1-x2)^10-1.396
e <- rnorm(n, 0, sig)
y <- f + e
b<-gam(y~s(x0)+s(x1)+s(x2)+s(x3))
summary(b)
plot(b)
```

Dalam tipa-tiap pustaka, biasanya ada informasi referensi untuk mempelajari teori serta beberapa contoh penggunaan pustaka tersebut, seperti terlihat pada dokumentasi `gam(mgcv)` sebelumnya. Apabila sebagian contoh yang ada skripnya `gam()` ini dikopi ke RConsole maka diperoleh keluaran seperti berikut dan grafik *smoothing* untuk tiap-tiap peubah bebas yang ditunjukkan oleh Gambar 2.10.

```
Family: gaussian
Link function: identity

Formula:
y ~ s(x0) + s(x1) + s(x2) + s(x3)

Parametric coefficients:
            Estimate    std. err. t ratio  Pr(>|t|)
(Intercept) 2.7602      0.1049    26.31    < 2.22e-16

Approximate significance of smooth terms:
              edf       chi.sq     p-value
s(x0)       5.173        28.87     4.4272e-05
```

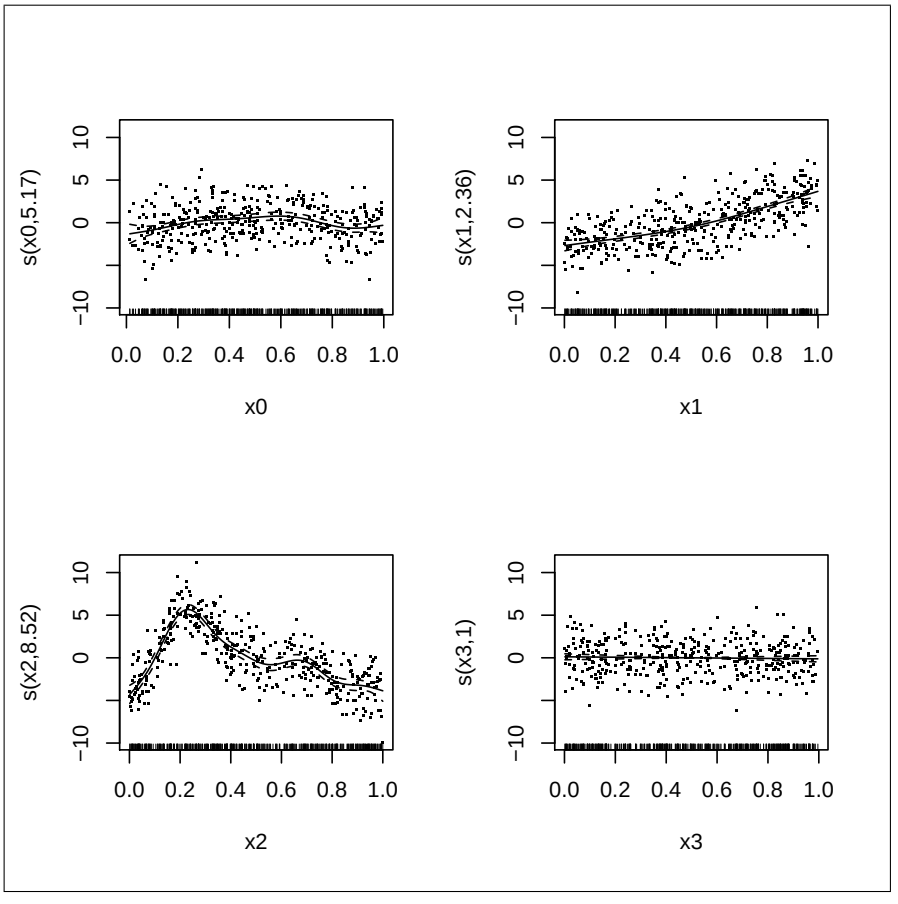
```
s(x1)      2.357       302.05     < 2.22e-16
s(x2)      8.517        753.1     < 2.22e-16
s(x3)          1      0.44123     0.50693

R-sq.(adj) =  0.726   Deviance explained = 73.7%
GCV score =  4.611   Scale est. = 4.4029    n = 400
```

2.6.2 Memanggil Pustaka dengan CLI

Apablia kita telah mengetahui cara pemanfaatan suatu pustaka melalui contoh yang ada, maka kita dapat menirunya untuk dipergunakan dalam analisis data riil yang kita miliki. Misalkan dari data `Orange` dengan variabel Tree, age dan circunference, kita dapat melakukan analsis gam yang paling sederhana seperti berikut.

```
> attach(Orange)
> library(mgcv)
> summary(gam(circumference~age,data=Orange))

Family: gaussian
Link function: identity

Formula:
circumference ~ age

Parametric coefficients:
             Estimate  std. err.    t ratio    Pr(>|t|)
(Intercept)      17.4      8.623      2.018    0.051793
        age   0.10677   0.008277       12.9    1.9306e-14

R-sq.(adj) =   0.83   Deviance explained = 83.5%
GCV score = 597.63   Scale est. = 563.48    n = 35
```

Dengan pilihan distribusi = Gamma dan fungsi link log, maka diperoleh hasil



Gambar 2.10: Contoh Grafik *Smoothing* dengan analisis GAM pada Pustaka mgcv

```
> summary(gam(circumference~age+Tree,family=Gamma(log),
  data=Orange))

Family: Gamma
Link function: log

Formula:
circumference ~ age + Tree

Parametric coefficients:
             Estimate  std. err.    t ratio    Pr(>|t|)
(Intercept)     3.5246    0.06863      51.35    < 2.22e-16
        age  0.0011746 6.588e-05      17.83    < 2.22e-16
     Tree.L    0.30679    0.07141      4.296    0.00017816
     Tree.Q   0.013257    0.07141     0.1856    0.85402
     Tree.C  -0.059361    0.07141    -0.8312    0.41263
     Tree^4  -0.064685    0.07141    -0.9058    0.37251


R-sq.(adj) =  0.835   Deviance explained = 90.5%
GCV score = 0.043084   Scale est. = 0.035698  n = 35
```

2.7 Mengemas Fungsi Menjadi Paket

Berbagai fungsi yang terkait dengan permasalahan tertentu dapat dikemas manjadi suatu paket. Paket yang dihasilkan selanjutnya dapat dipublikasikan dan dimanfaatkan orang lain. R sesungguhnya telah menyediakan panduan khhusus untuk pengembangan paket. Ada beberapa komponen penting yang harus disiapkan dalam menyusun paket yaitu:

1. **Fungsi R**. Program yang memuat definisi fungsi-fungsi yang ingin dijadikan paket. Cara mendefinisikan fungsi secara garis besar telah dibahas pada bagian sebelumnya.

2. **Panduan.** Bagian ini memuat panduan atau manual dari fungsi-fungsi penting yang akan dijadikan paket. Pedoman penulisan panduan atau manual paket dapat dilihat pada dokumentasi setiap versi R yang berjudul *Writing R Extension*. Panduan yang ada seara khusus membahas pembuatan pedoman dengan menggunakan pengolah kata LaTeX.

Sesungguhnya langkah langkah pengemasan paket juga dapat dilihat pada dokumen yang sama, namun bagi pemula hal tersebut relatif sulit diikuti. Schnute (2006) dan Schnute (2007) menguraikan langkah-langkah yang lebih rinci dan sederhana untuk membuat paket. Langkah-langkah yang perlu dibuat dapat dibedakan menjadi dua bagian besar yaitu:

1. Instalasi program-program yang diperlukan. Ada beberapa program yang diperlukan untuk membuat paket R yaitu Program R itu sendiri, Program rtool, Perl dan MinGW untuk membuat struktur paket, Program LaTeX(misalnya MikTex 2.5 dengan Editornya WinEdt) untuk memroses dokumentasi dan panduan, Program HTML Help untuk membuat dokumentasi dalam bentu HTML,

2. Pemrosesan Paket Jika piranti lunak yang dibutuhkan sudah berfungsi dengan baik, selanjutnya langkah untuk membuat paket adalah sebagai berikut.

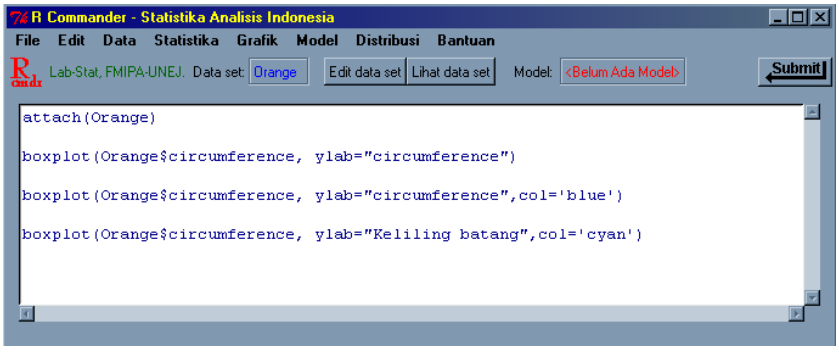
   (a) Buat direktori dengan nama sesuai dengan nama paket yang akan dibuat. Pada direktori ini ditempatkan file DESCRIPTION yang mengatur nama, serta versi paket. Selain itu pada direktori ini juga ditempatkan file-file bat (eksekusi) yang dibuat oleh Schnute (2006) dan (2007) yang secara semi otomatis nanti melakukan pengemasan paket.

   (b) Dalam direktori ini selanjutnya dibuat subdirektori yaitu subdirekori `man` (tempat menyimpan file-file panduan, dan

subdirektori `R` (tempat menyimpam file-file R), termasuk file zzz.r

(c) Selanjutnya dilakukan langkah pengemasan yang meliputi pengecekan paket, pembuatan, dan pengepakan paket. Untuk lengkapnya dapat dilihat pada Schnute(2006) dan Schnute(2007).

2.8 CLI dan RCommander



Gambar 2.11: Perintah Menu pada Jendela Skrip. Perintah-perintah yang dipanggil melalui menu dapat dicatat pada Jendela Skrip

Sesungguhnya setiap kali kita mengirim perintah ke R melalui RCommander, maka skrip CLInya disalin pada Jendela Skrip. Skrip ini dapat dimodifikasi dan selanjutnya dikirim ke R melalui panel "Submit". Modifikasi dapat dilakukan, misalnya, untuk mengganti judul grafik, judul absis, warna grafik, dan sebagainya (lihat Gambar 2.12. Untuk mengirim baris perintah pada Jendela Skrip dapat dilakukan langkah-langkah berikut:

1. tulis perintah R atau modifikasi perintah baris yang ada pada Jendela Skrip;

2. blok (*highlight*) baris perintah yang dimaksud;

3. klik panel "Submit".

Perhatikan bahwa kalau sebelum ini R Commander hanya mampu mengirim secara efektif 1 baris perintah, maka sejak versi 1.3-12, R Commander sudah bisa mengirim lebih dari 1 baris perintah. Untuk perintah lebih dari 1 baris, lakukan dengan:

1. blok (*highlight* bagian yang ingin dijalankan, lalu tekan tombol `kirim`;

2. tekan langsung kirim, jika semua skrip ingin dikirim sekaligus;

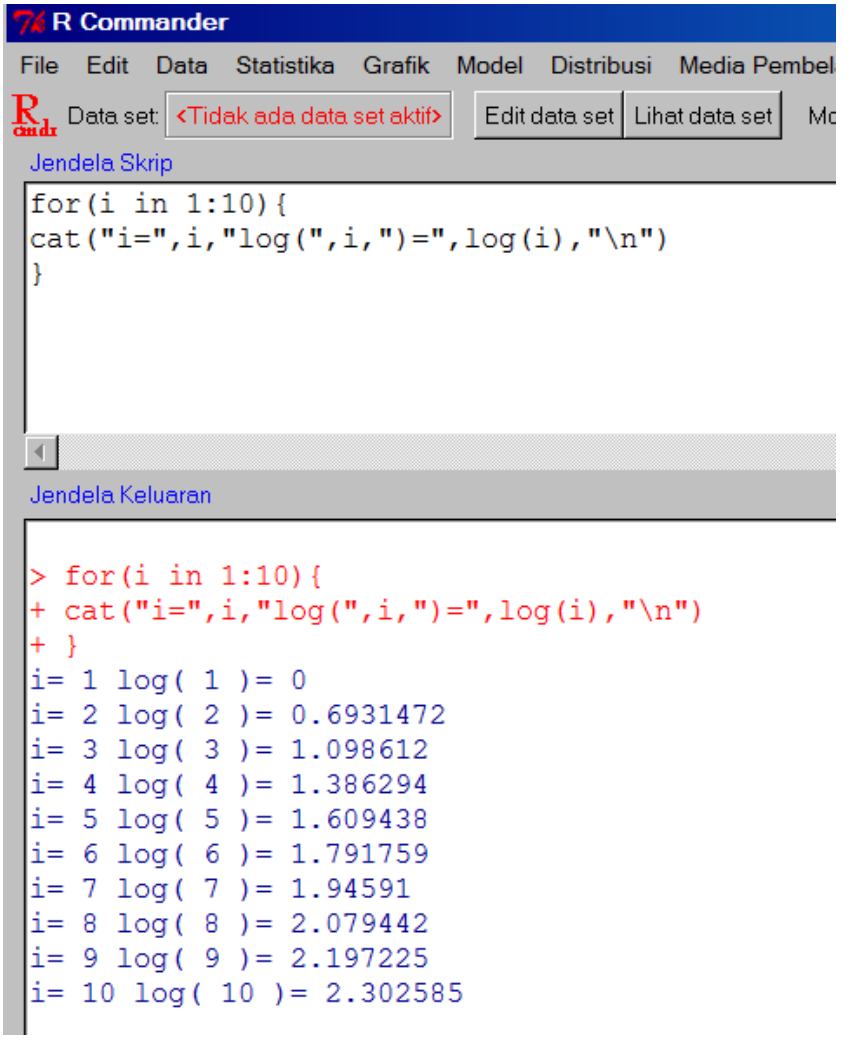
3. *copy-paste* perintah langsung ke Console R;

4. mengedit dan menjalankan skrip dengan Jendela Command dari SciViews atau Tinn-R.

2.9 Bacaan Lebih lanjut

Bagi yang ingin menekuni lebih jauh pemrograman dengan R atau S-Plus dapat membaca referensi dari Burns (1998). Khusus untuk kemampuan grafik R, dapat dibaca referensi Murrell (2006). Aplikasi R untuk analisis statistika, dengan menggunakan pendekatan CLI dapat juga dibaca pada beberapa referensi berikut di antaranya: Tirta (2000), Venables & Ripley(1994), Everitt(2001), Marazzi(1992), Chamber & Hastie(1992), Everitt (2005).

Selain itu masih banyak referensi *on-line* yang dapat dilihat pada situs `http://www.r-roject.org` seperti telah diuraikan sebelumnya.



Gambar 2.12: Perintah Menu pada Jendela Skrip dapat dieksekusi langsung

DAFTAR ISTILAH

D

Diagram Pencar (*Scattergram*) Diagram pencar adalah representasi grafik dari distribusi dua peubah acak yang disajikan dalam bentuk titik-titik dengan koordinat ditentukan oleh nilai observasi pasangan peubah acak tadi.

H

html HTML(*Hyper Text Markup Language*) adalah format dokumen/file yang dipergunakan untuk mengkonstruksi informasi pada situs internet.

I

Ikon ikon/*icon* adalah gambar/logo kecil yang mewakili suatu program dan apabila gambar ini di-klik, maka program terkait akan dibuka.

K

Keluarga Eksponensial Keluarga Eksponensial adalah distribusi yang merupakan kesatuan (unifikasi) distribusi-distribusi penting yang banyak dipakai seperti antara lain Normal, Gamma, Binomial, Poisson dalam satu bentuk distribusi.

L

LaTeX LaTeXadalah salah satu program OSS (*Open Source Software*) untuk pengolah kata. Program ini dipergunakanan secara luas di kalangan para metematisi.

Legend Legend adalah simbol/tanda yang dipergunakan dalam grafik untuk membedakan adanya subkelompok dari data yang digambar.

Linux Linux adalah salah satu program OSS (*Open Source Software*) untuk sistem operasi. Program mulai banyak diminati sebagai alternatif sistem operasi selain windows atau unix.

M

Matriks Diagram Pencar Matriks Diagram Pencar (*Scatter Plot Matrix*) adalah matriks yang menggambarkan diagram pencar lebih dari dua variabel. Pada diagonal biasanya disajikan densitas, histogram atau diagram kuantil, sedangkan pada *off diagonal* disajikan diagram pencar masing-masing pasangan variabel.

O

Outlier/pencilan Pencilan adalah data yang besarnya menyimpang dari kelompoknya melebihi batas kewajaran distribusi data.

P

Parameter Parameter (statistika) adalah ukuran deskriptif numerik dari populasi.

Parameter Parameter (fungsi pada R) adalah bagian dari fungsi yang nilainya dapat ditentukan pada saat pemanggilan fungsi tersebut. Misalnya dalam rumus abc ada tiga parameter yang diperlukan (a,b, dan c).

Plugin/Plug-in Adalah program yang dapat digabungkan menjadi bagian dari proram lain yang lebih besar. Dalam buku ini Paket-peket Plug-in R Commander, menunya dapat digabungkan menjadi bagian dari menu R Commander.

Populasi Populasi adalah himpunan semesta dari variabel yang menjadi perhatian peneliti.

Q

QQPlot QQplot atau Plot Kuantil adalah diagram yang menggambarkan hubungan antara quantil teoritis suatu distribusi dengan kuantil riil suatu data. Khusus untuk distribusi normal grafiknya disebut *QQnorm*.

S

skrip Skrip adalah naskah yang berisi berbagai perintah yang harus dilaksanakan oleh komputer melalui suatu bahasa atau program tertentu.

W

Widget Widget adalah komponen GUI yang dihasilkan oleh TclTk, yang meliputi jendela, tombol, slider dan sebagainya.

Workspace *Workspace* adalah kondisi kerja yang telah dilakukan oleh R yang menyangkut data aktif, pustaka aktif dan objek-objek yang dihasilkan. R dapat menyimpan informasi ini sebelum ditutup untuk dipergunakan pada eksekusi berikutnya.

DAFTAR PUSTAKA

Baron & Y. Li, J. (n.d.), *Notes on the Use of R for Psychology Experiment and Questionaire*, http://www.r-project.org.

Bowerman,B.L. R.T. Cornell and D.A. Dickey (1986), *Linear Statistical Models, an Appplied Approach*, Duxbury Press, Boston.

Burns, P. (1998), *S Poetry*, http://www.r-project.org.

Burns, P. (2002), *Guide for Unwilling S*, http://www.r-project.org.

Chamber, J. & Hastie, T. (1992), *Statistical Model in S*, Chapman and Hall, London.

Crawley (2004), *Statistical Computing: An Introduction to Data Analysis using S-Plus*, Wiley, England.

Dalgaard, P. (2001), 'A primer on the Tcl/Tk package', *R News* **1**(3), 27–31.

Dalgaard, P. (2002), 'Changes to the Tcl/Tk package', *R News* **2**(3), 25–27.

Diggle P.J., K-Y. Liang & Zeger, S. (1994), *Analysis of Longitudinal Data*, 1st edn, Oxford Science Publications, London.

Everitt, B. (2001), *A Handbook of Statistical Analyses using S-Plus*, 2nd edn, Chapman & Hall.

Everitt, B. (2005), *An R and S-Plus Companion to Multivariate Analysis*, Springer.

Faraway, J. (2002), *Practical Regression and Anova Using R*, http://www.stat. Isa.umic.edu/∼faraway/book/.

Hadi, S. (1982), *Statistika*, Andi Offset, Yogyakarta.

Hastie & R.J. Tibshirani, T. (1990), *Generalized Additive Models*, 5th edn, Chapman & Hall, London.

Kuhnert, P. & Venables, B. (2005), *An Introduction to R: Software for Statistical Modelling & Computing*, CSIRO, http://cran.r-project.org/doc/contrib/Kuhnert+Venables-R_Course_Notes.zip. [17 April 2006].

Maindonald, J. (2001), *Using R for Data Analysis and Graphics An Introduction*, ANU-Australia.

Marazzi, A. (1992), *Algorithms, Routines and S Functions for Robust Statistics*, Wadsworth and Brook/Cole, Pacific Grove.

McCullagh, P. & Nelder, J. (1989), *Generalized Linear Models*, 2nd edn, Chapman and Hall, London.

Mendenhall, W. (1979), *Introduction to Probability and Statistics*, 5th edn, Duxbury, Belmont USA.

Mendenhall, W. (1993), *Beginning Statistics A to Z*, Duxbury, Belmont USA.

Murrell, P. (2006), *R Graphics*, Chapman& Hall/CRC.

Neter J., W. Wasserman and M.H. Kutner (1985), *Applied Linear Statistical Models*, 2nd edn, Irwin, Illinois.

Ousterhout, J. K. (1994), *Tcl and the Tk Toolkit*, Addison-Wisley.

Paradis, E. (2002), *R for Beginner*, http://www.r-project.org.

R-Teams (n.d.*a*), *Data Import/ Export*, R Project, http://www.r-project.org.

R-Teams (n.d.*b*), *An Introduction to R*, R Project, http://www.r-project.org.

R-Teams (n.d.*c*), *R Instalation and Administration*, R Project, http://www.r-project.org.

R-Teams (n.d.*d*), *R Language Manual*, R Project, http://www.r-project.org.

R-Teams (n.d.*e*), *Reference Manual*, R Project, http://www.r-project.org.

R-Teams (n.d.*f*), *Writing R Extension*, R Project, http://www.r-project.org.

Schnute, J.T. Couture-Beil, A. & Haigh R (2006), *PBS Modelling 1: User's Guide*.

Schnute, J.T. Couture-Beil, A. & Haigh R (2007), *PBS Modelling 1: Developer's Guide*.

Sudjana (1996), *Metode Statistika*, Tarsito, Bandung.

Tirta, I. M. (2000), *Pemrograman Statistika dengan S-Plus 4.5*, FMIPA Universitas Jember. Diktat Kuliah.

Tirta, I. M. (2005*a*), *Buku Panduan Program Statistika R*, Penerbit Universitas Jember, Jember. ISBN 979-8176-37-5.

Tirta, I. M. (2005*b*), 'Pemanfaatan OSS-R dalam pengajaran statistika', *Seminar Nasional Matematika UNESA*.

Tirta, I. M. (2007), *R-GUI. Mendesain PAket Analisis Data dan Media Pembelajaran Statistika*, Penerbit Universitas Jember, Jember.

Venables, W. & Ripley, B. (1994), *Modern Applied Statistics with S-plus*, Springer, New York.

Vezalini, J. (2002), *Using R for Introductory Statistics*, http://www.r-project.org.

Wang, K. (n.d.), *R for Windows Users*, http://www.r-project.org.

Welch, B. (1995), *Practical Programming in Tcl and Tk*, Pretice Hall.

INDEKS PENULIS

Baron, 34
Bowerman, 68
Burns, 33, 117

Chamber, 35, 141

Dalgaard, 156

Everitt, 35, 141

Faraway, 3, 34
Fox, 38, 73

Grosjean, 70, 73

Hadi, 67
Hastie, 35, 141

Kuhnert, 34

Li, 34

Maindonald, 34
Marazzi, 141
McCullagh, 68
Mendenhall, 68
Murrell, 141

Nelder, 68
Netter, 68

Ousterhout, 156

Paradis, 33

R-Team, 32, 33
Ripley, 34, 68, 141

Schnute, 139, 150
Sudjana, 67

Tibshirani, 35
Tirta, 141, 150, 151

Venables, 34, 68, 141

Vezalini, 34

Welch, 156

INDEKS SUBJEK

aplot, 220

barisan, 87
    berpola, 88
berorietasi objek, 3
biplot, 221

CLI, 2

data, 61
    memanggil, 61
    menyusun, 63
demo, 30
diagram
    pencar
        glosari, 55

emacs, 20
ESS, 6, 20

fungsi, 103, 110
    file, 101
    grafik, 95
    matematika, 85, 86
    matriks, 87, 88
    R, 219
    statistika, 91

Geostatistika, 32
glm, 31
grafik
    dinamik, 147
GUI, 2, 3

hplot, 221, 223

internasionalisasi, 148

Jendela Command
    SciViws, 82

kelengkapan R, 5

latex, ix, 61
library, 2, 5
  tar.gz, 5
  tcltk, 144
  zip, 5
Linux, 4
  Debian, 5
  emacs, 20
    ESS, 20
  instal
    pustaka, 22
  install
    paket utama, 22
  Mandrake, 5, 42
  R Commander, 42
  Shuse, 5

multivariat, 221

objek, 3

parameter, 93
  glosari, 92
plugin, 66, 67, 149
program
  algoritma, 104
  diagram alir, 104
  komponen, 102
  langkah, 104
pustaka, 2, 5, 131
  aktifkan, 133
  daftar seluruh, 165
  fungsi, 133
  instal
    Windows, 22
  install
    Linux, 22
  kepustakaan, 135
  rangkuman, 26
  SciViews
    dock, 16
  tcltk, 144

R Commander, 38
  Indonesia
    Linux, 38
    Windows, 38
  SciViews, 73
Rcli, 82
RGUI, 6
  Linux, 6
  R Commander, 9
  SciViews, 3, 9, 12, 71
  WinEdt, 3

S, 2
S-Plus, 85
SciViews, 69
  dock, 70, 71
    pustaka, 16
  dokumentasi, 80
  file eksplorer, 74
  kartu referensi, 77
  objek exssplorer, 76
  R Commander, 73
  RGUI, 71
Sciviews, 6
simulasi, 103

normal
multivariat, 241
univariat, 239
StatDemo, 151

tcl, 144
tcltk, 144
menu dialog, 144
R Commander, 146
tcltk2, 156
Tinn-R, 6
tk, 144
TkRPlot, 150

widget, 151

INDEKS FUNGSI R

%*%, 88

a, 220
axis(), 95

biplot(), 221

cat(), 101
cor(), 91

demo(), 30, 101
det(), 88
diag(), 88
dnorm(), 93, 239

edit(), 63
example(), 101

gamma(), 86

interaction.plot(), 223

layout(), 96
legend(), 95
list(), 115
log10(), 86

matrix(), 88
max(), 91
mean(), 91
median(), 91
min(), 91
mvrnorm(), 241

names(), 116

par(), 95, 227
plot(), 95
pnorm(), 93, 239
points(), 95
print(), 101
prod(), 88

qnorm(), 93, 239

read.table(), 101
rep(), 88
rnorm(), 93, 239

sample(), 91
seq(), 88
sink(), 101
solve(), 88
source(), 101
split.screen(), 96
sum(), 88

t(), 88
tcltk2, 156
text(), 95
tr(), 88
tranc(), 86

var(), 91