

# **PENGENALAN BAHASA R UNTUK ANALISIS DATA DAN GRAPHIK**

*I G.A. Anom Yudistira*  
(D1392)

e-mail: anom1392@lecturer.binus.ac.id

**JURUSAN STATISTIKA  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS BINA NUSANTARA  
2005**

## 1. Pengantar

R adalah suatu sistem untuk analisis statistik dan grafik yang diciptakan oleh Ross Ihaka dan Robert Gentleman. Kelebihan yang diperoleh bila menggunakan bahasa R untuk pengajaran statistik adalah:

- R dapat diperoleh dengan gratis. R merupakan suatu *open-source* dan dapat digunakan pada berbagai sistem operasi seperti UNIX, Windows, Linux dan Macintosh.
- R memiliki sistem bantuan (*help*) yang canggih.
- R memiliki kemampuan membuat grafik yang canggih.
- Mahasiswa dapat dengan mudah berpindah ke sistem komersial *S-Plus*, bila software komersial diperlukan.
- Bahasa R mempunyai kemampuan yang tangguh, syntaxnya mudah dipelajari dengan banyak fungsi-fungsi statistik yang terpasang (*built-in*).
- Bahasa R dapat dengan mudah diperluas dengan menciptakan fungsi-fungsi buatan pengguna sendiri.
- R merupakan bahasa pemrograman komputer, sehingga bagi pemrogram menjadi lebih akrab, sedangkan bagi pemakai awal akan merupakan langkah yang mudah untuk memulai sebagai pemrogram komputer.

Apa yang menjadi kelemahan R dibandingkan dengan software-software statistik yang lain.

- R memiliki antarmuka untuk grafik yang terbatas (*S-Plus* memiliki lebih banyak).
- Tidak tersedia dukungan komersial (tetapi mailing list internasional dapat menggantikannya).
- Perintah-perintahnya merupakan bahasa pemrograman, jadi mahasiswa harus mempelajari sintaksnya.

Sebagian besar fungsi-fungsi *S-Plus* dapat digunakan pada R. Fungsi-fungsi tersebut dapat dilihat di: <http://www.insightful.com/products/splus/default.html> atau di <http://stat.cmu.edu/S/>. Sedangkan alamat web untuk bahasa R adalah <http://www.r-project.org>, file instalasi R dapat di download gratis di CRAN pada web ini.

## 2. Bagaimana R bekerja

R adalah bahasa pemrograman berorientasi objek, yang artinya semua peubah, data, fungsi, hasil dan sebagainya disimpan dalam memori aktif komputer dalam bentuk objek yang mempunyai nama. Pengguna dapat melakukan aksi terhadap objek ini dengan menggunakan **operator** (aritmatik, logikal, dan pembandingan) dan **fungsi** (yang dia sendiri merupakan objek). Semua aksi R dilakukan pada objek-objek yang ada pada memori aktif komputer: tanpa menggunakan file temporer (*temporary file*). Proses membaca dan menulis file hanya digunakan untuk input dan output data dan hasil (grafik,...). Pengguna mengeksekusi fungsi melalui serangkaian perintah dan hasilnya ditampilkan langsung pada layar, disimpan pada objek atau ditulis ke hard disk (khususnya grafik). Karena hasil itu sendiri merupakan objek, maka ia dapat dipandang sebagai data dan dianalisa sebagaimana halnya data. File-file data dapat dibaca dari disk lokal atau server melalui internet.

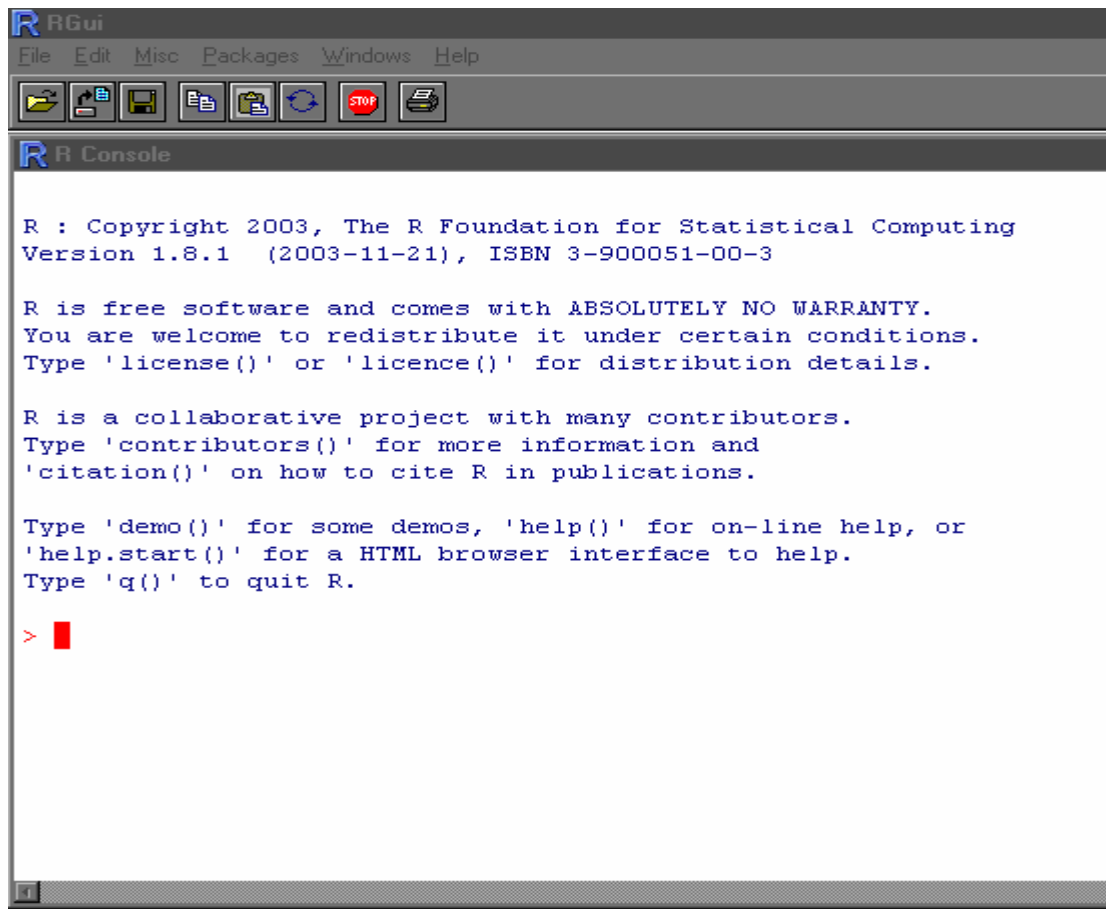
Fungsi-fungsi yang tersedia untuk pengguna disimpan pada sebuah library di disk dalam sebuah direktori bernama *R\_HOME/library* (*R\_HOME* adalah direktori dimana R terpasang). Direktori ini berisi fungsi-fungsi *packages*, yang mana mereka tersusun dalam direktori-direktori. Package yang bernama *base* merupakan inti dari R, yang berisi

fungsi-fungsi dasar dari bahasa R untuk membaca dan memanipulasi data, beberapa fungsi-fungsi grafik, dan sebagian fungsi-fungsi statistik. Setiap package berada pada direktori R dan direktorinya diberi nama sama dengan nama package tersebut. Misal package base file-filenya ada pada R\_HOME/library/base/R/base.

### 3. Data

#### Starting R

Satelah anda membuka R, maka pada console akan tampil sebagai berikut



```
R : Copyright 2003, The R Foundation for Statistical Computing
Version 1.8.1 (2003-11-21), ISBN 3-900051-00-3

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 in publications.

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

> █
```

Tanda > disebut prompt. Tanda ini muncul secara otomatis tidak perlu diketik, yang mana berguna sebagai penunjuk dimana anda akan mengetikkan perintah-perintah R seperti pada teladan di bawah ini. Jika sebuah perintah terlalu panjang maka anda dapat melanjutkan ke baris berikutnya (dengan menekan enter), dan akan muncul tanda + sebagai prompt yang berarti kelanjutan dari baris di atasnya.

#### Menciptakan object dalam memori aktif

Sebuah object dapat diciptakan dengan operator “<- atau ->”, untuk versi 1.4 ke atas dapat dengan operator “=”.

Teladan: Buatlah objek n yang bernilai 15 (dalam bahasa sehari-hari objek n adalah variabel n)

```
> n<-15
> n
[1] 15
> 15->n
> n
[1] 15
> n=10      # nilai objek n yang lama (15) dihapus & diganti dengan 10
> n
[1] 10
```

Semua kata dibelakan tanda # tidak akan diproses oleh R. Bahasa R sensitif terhadap huruf besar atau kecil (*case sensitive*) walaupun menggunakan sistem operasi windows. Jadi objek x dan X merupakan objek yang berbeda.

```
> x<-1
> X<-10
> x
[1] 1
> X
[1] 10
```

Nilai yang diberikan pada sebuah objek dapat merupakan hasil dari suatu operasi dan/atau fungsi.

```
> n<-5+2
> n
[1] 7
> n<-2+rnorm(1) # fungsi rnorm(1) membangkitkan sebuah peubah acak normal baku
> n
[1] 2.723687
```

Anda dapat menuliskan ekspresi tanpa menyimpan hasilnya ke objek (memori), tetapi menampilkan secara langsung ke layar.

```
> (10+2)*3
[1] 36
> sqrt(16) # fungsi akar kuadrat
[1] 4
```

### Memasukkan Data dengan fungsi c

Data dengan jumlah kecil lebih efisien dimasukkan dengan menggunakan fungsi c. Misalkan anda akan memasukkan data jumlah salah ketik per halaman, untuk 8 lembar halaman ketikan, datanya adalah:

```
      2 3 0 3 1 0 0 1
> saltik<-c(2,3,0,3,1,0,0,1)
> saltik
[1] 2 3 0 3 1 0 0 1
```

Rata-rata, median, ragam dan simpangan baku dari data yang disimpan pada peubah saltik, diperoleh dengan menggunakan fungsi-fungsi berikut:

```
> mean(saltik)      # rata-rata
[1] 1.25
> median(saltik)    # median
[1] 1
> var(saltik)       # ragam sampel
[1] 1.642857
> sd(saltik)        # simpangan baku sampel
[1] 1.28174
```

Ringkasan statistiknya diperoleh dengan fungsi summary

```
> summary(saltik)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00   0.00   1.00   1.25   2.25   3.00
```

Bila anda bermaksud untuk mengambil satu persatu nilai ringkasan statistik dari saltik, maka lakukan serangkaian perintah berikut.

```
> ringkas<-summary(saltik)
> ringkas[1] # ambil nilai objek ringkas pada urutan 1
Min.
0
> ringkas[2] # ambil nilai objek ringkas pada urutan 2
1st Qu.
0
> ringkas[5] # ambil nilai objek ringkas pada urutan 5
3rd Qu.
2.25
```

### Data sebagai vektor

Sebagaimana telah terlihat sebelumnya data oleh R diperlakukan sebagai sebuah vektor. Sehingga urutan pemasukkan datanya akan teracak.

```
> saltik[2] # data saltik pada urutan 2, perhatikan gunakan kurung siku
[1] 3
> saltik[8] # data saltik pada urutan 8
[1] 1
```

Misalkan kita mempunyai dua buah data salah ketik sebagai berikut. 2,3,0,3,1,0,0,1 dimasukkan ke objek saltik.draf1 dan data 0,3,0,3,0,0,1 dimasukkan ke objek saltik.draf2. Peubah saltik.draf2 hanya berbeda pada data urutan 1 dengan saltik.draf1, maka lebih efisien memasukkan datanya adalah sebagai berikut.

```
> saltik.draf1<-c(2,3,0,3,1,0,0,1)
> saltik.draf2<-saltik.draf1 # copy data saltik.draf1 ke saltik.draf2
> saltik.draf2[1]<-0 # ganti nilai data pada urutan satu dengan 0
> saltik.draf1
[1] 2 3 0 3 1 0 0 1
```

```
> saltik.draf2
[1] 0 3 0 3 1 0 0 1
> saltik.draf2[4] # data urutan 4
[1] 3
> saltik.draf2[-4] # semua data kecuali urutan 4
[1] 0 3 0 1 0 0 1
> saltik.draf2[c(1,2,3)] # data urutan 1,2 dan 3
[1] 0 3 0
```

Misalkan anda ingin mengetahui jumlah salah ketik yang paling banyak, dan ada pada halaman mana saja? maka gunakan rangkain perintah berikut.

```
> max(saltik.draf2)
[1] 3
> saltik.draf2==3 # operator comparison == berarti sama dengan
[1] FALSE TRUE FALSE TRUE FALSE FALSE FALSE FALSE
> which(saltik.draf2==3) # fungsi ini mencari urutan data yang bernilai TRUE
[1] 2 4
```

Bila anda tidak menggunakan fungsi which, maka dapat dilakukan dengan cara yang lebih meutar yaitu.

```
> n<-length(saltik.draf2) # berapa banyak data (halaman)
> hal<-1:n # cara untuk mmmendapatkan nomor halaman
> hal
[1] 1 2 3 4 5 6 7 8
> hal[saltik.draf2==3]
[1] 2 4
```

perintah `hal<-1:n` adalah cara ringkas untuk menuliskan perintah `hal<-c(1:n)`. Cara yang lebih ringkas untuk menuliskan perintah-perintah untuk mendapatkan nomor halaman dengan salah ketik terbanyak adalah.

```
> (1:length(saltik.draf2))[saltik.draf2==3]
[1] 2 4
```

Mungkin anda ingin mengetahui berapa jumlah salah ketik dari seluruh draf 2, mungkin juga ingin mengetahui berapa banyak halaman dengan salah ketik lebih dari 2, dan terakhir berapa perbedaan salah ketik untuk setiap halaman yang bersesuaian. Untuk mendapatkan jawabannya perhatikan perintah-perintah berikut.

```
> sum(saltik.draf2) # berapa jumlah salah ketik (0+3+0+3+1+0+0+1)
[1] 8
> sum(saltik.draf2>2) # berapa banyak hal. dg. salah ketik lebih dari 2
[1] 2
> saltik.draf1-saltik.draf2
[1] 2 0 0 0 0 0 0 0
```

## **Latihan**

1. Masukkanlah data di bawah ini dengan fungsi c dan letakkan pada objek x

45, 43, 46, 48, 51, 46, 50, 47, 46, 45

2. Carilah nilai rata-rata, median, maksimum, dan minimum
3.
  - a. Tambahkan data objek x dengan 48, 49, 51, 50, 49
  - b. Berapa banyak datanya sekarang (gunakan fungsi `length`)?
  - c. Berikan nilai 41 pada objek x sebagai data ke 16 (gunakan `x[16]`).
  - d. Tambahkan datanya dengan 40, 38, 35, 40 (gunakan `x[17:20]`).

### **Antarmuka (interface) Data Entri Graphis**

R menyediakan banyak cara lain untuk mengedit data melalui antarmuka *spreadsheet*. Antar muka ini mungkin lebih disukai mahasiswa. Berikut ini adalah beberapa contohnya.

```
> data.entry(x) # muncul spreadsheet untuk mengedit data x
> x<-de(x) # idem, coba perhatikan perbedaannya dengan mengetik x
> x<-edit(x) # muncul notepad editor untuk mengedit x
```

Semua perintah di atas mudah untuk digunakan., hanya saja objek data yang akan diedit harus telah didefinisikan sebelumnya, bila objek belum ada maka akan timbul pesan kesalahan, seperti berikut ini.

```
> data.entry(z) # gagal, objek z belum terdefinisi
Error in de(..., Modes = Modes, Names = Names) :
  Object "z" not found
> data.entry(z=c(NA)) # muncul spreadsheet
> zz<-numeric() # definisikan variabel/objek zz dengan tipe numerik
> data.entry(zz) # muncul spreadsheet
```

Keterangan:

Jika objek zz bertipe karakter maka gunakan fungsi `character()`.

### **Latihan**

1. Data berikut ini adalah jumlah mahasiswa yang hadir kuliah pada suatu kelas tertentu di universitas Bina Nusantara, selama 12 kali pertemuan.

46 33 39 37 46 30 48 32 49 35 30 48

- a. Entri data ke variabel hadir, gunakan antarmuka spreadsheet.
  - b. Hitunglah berapa banyak pertemuan dengan kehadiran kurang dari 40 mahasiswa.
  - c. Tampilkan semua data kecuali data pertemuan 2, 4, 6, 8, dan 10.
  - d. Gunakan fungsi `range` untuk mendapatkan kehadiran terbanyak dan terendah

```
> range(hadir)
```

```
[1] 30 49
```

- e. Gunakan fungsi `diff` untuk mencari selisih kehadiran yang berurutan.

```
> diff(hadir)
```

```
[1] -13 6 -2 9 -16 18 -16 17 -14 -5 18
```

```
> diff(range(hadir)) # rentang data
```

```
[1] 19
```

- f. Gunakan fungsi `cummax` untuk mendapatkan kehadiran terbanyak secara kumulatif.

```
> cummax(hadir)
```

```
[1] 46 46 46 46 46 46 48 48 49 49 49 49
g. Gunakan fungsi cummin untuk mendapatkan kehadiran terendah secara komulatif.
> cummin(hadir)
[1] 46 33 33 33 33 30 30 30 30 30 30 30
h. Rentang antar kuartil diperoleh dengan menggunakan fungsi IQR
> iqr<-IQR(hadir) # nilai IQR disimpan di objek iqr
> iqr
[1] 13.75
```

### Menampilkan daftar objek dan menghapusnya.

Untuk melihat objek apa saja yang telah dibuat adalah sebagai berikut.

```
> ls() # melihat daftar objek yang telah dibuat
[1] "c(NA)"      "hadir"      "hal"        "iqr"        "last.warning"
[6] "n"          "ringkas"    "saltik"     "saltik.draf1" "saltik.draf2"
[11] "x"          "X"          "z2"         "zz"
```

Anda mungkin saja mendapatkan daftar objek yang berbeda dengan di atas.

```
> rm(hadir) # menghapus objek hadir
> ls() # objek hadir hilang dari daftar
[1] "c(NA)"      "hal"        "iqr"        "last.warning" "n"
[6] "ringkas"    "saltik"     "saltik.draf1" "saltik.draf2" "x"
[11] "X"          "z2"         "zz"

> rm(list=ls()) # menghapus semua objek dari memori aktif
> ls()
character(0)
```

Untuk membersihkan layar (console) tekan Ctr+L

## **4. Data Univariate**

Ada bermacam-macam tipe data dalam statistik dan R dapat membedakan hal itu. Khususnya data dapat dibagi dalam tiga tipe dasar yaitu, katagorik, numerik diskret, dan numerik kontinu.

### Data Katagorik

Data katagorik biasa disajikan dalam bentuk tabel, grafik batang (*bar graph*), diagram kue (*pie chart*) dan sebagainya.

### Penggunaan Tabel

Fungsi table digunakan menyajikan data katagorik dalam bentuk tabel.

### *Teladan. Survei Merokok*

Sebuah survei menanyakan 5 orang apakah ia seorang perokok, atau tidak. Datanya adalah: Ya, Tidak, Tidak, Ya, Ya.



Masukkan data tersebut dalam R dengan fungsi `c()`, dan ringkaskan dengan menggunakan fungsi `table()` sebagai berikut.

```
> x<-c("Ya", "Tidak", "Tidak", "Ya", "Ya")
> table(x)
x
Tidak    Ya
     2     3
```

Data katagorik umumnya mengklasifikasikan datanya dalam beberapa level atau faktor. Fungsi `factor` mendapatkan level dari data katagorik.

```
> factx<-factor(x)
> factx
[1] Ya    Tidak Tidak Ya    Ya
Levels: Tidak Ya
```

Objek `factx` sekarang mempunyai level, tetapi `x` tidak. Perintah berikut ini memperlihatkan hal itu.

```
> levels(factx)
[1] "Tidak" "Ya"
> levels(x)
NULL
```

Bila nama level diubah Tidak menjadi non-perokok, dan Ya menjadi perokok, hasilnya adalah.

```
> levels(factx)<-c("non-perokok", "perokok")
> levels(factx)
[1] "non-perokok" "perokok"
> factx
[1] perokok    non-perokok non-perokok perokok    perokok
Levels: non-perokok perokok
> x # nilai objek x tidak berubah
[1] "Ya"    "Tidak" "Tidak" "Ya"    "Ya"
> table(factx)
factx
non-perokok    perokok
          2          3
```

### Grafik Batang (*Bar Graph*)

Garfik batang menampilkan sebuah batang yang tingginya sesuai dengan pencacahan dalam tabel (yang dilakukan oleh fungsi `table()`). Tinggi batang dapat mencerminkan frekuensi atau proporsi, keduanya akan nampak sama kecuali skalanya.

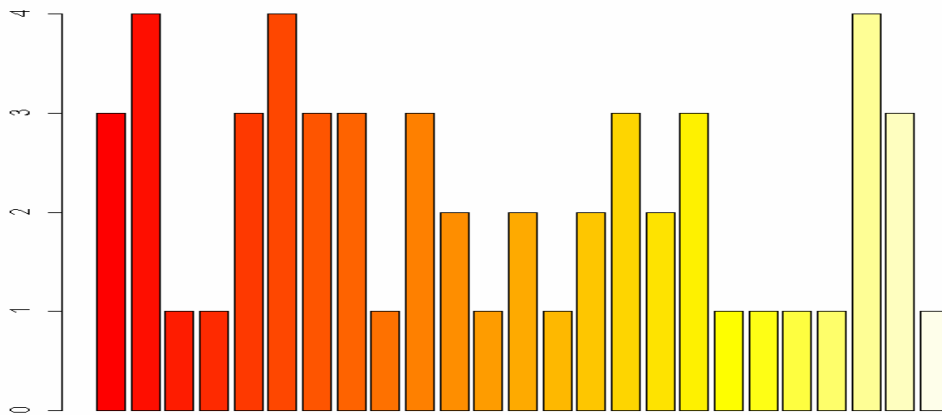
### *Teladan. Survei Olah Raga Permainan*

Ada 25 mahasiswa ditanyakan tentang permainan olah raga yang paling disukai. Kategorinya adalah sebagai berikut : (1). Sepak Bola, (2). Bola Voli, (3). Basket, (4).lainnya. Data mentahnya adalah sebagai berikut.

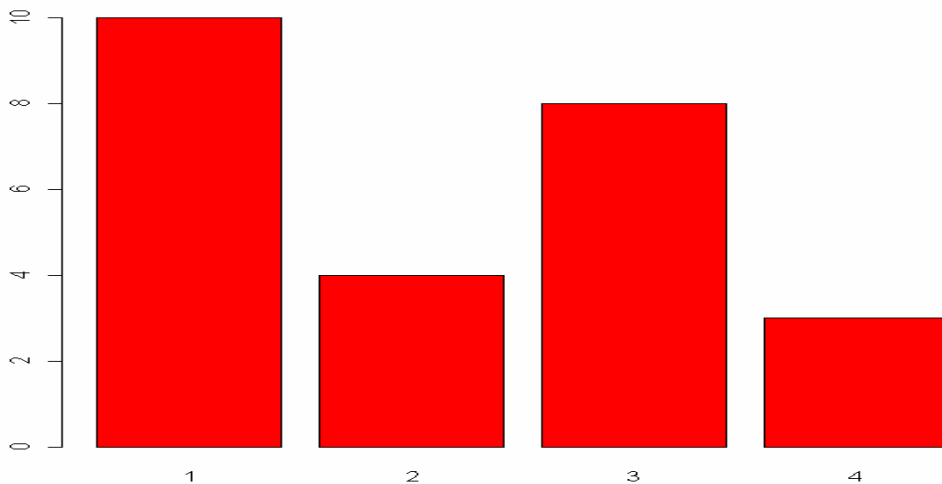
3 4 1 1 3 4 3 3 1 3 2 1 2 1 2 3 2 3 1 1 1 1 4 3 1

Berikut ini akan digunakan fungsi scan untuk membaca data di atas

```
> or<-scan()
1: 3 4 1 1 3 4 3 3 1 3 2 1 2 1 2 3 2 3 1 1 1 1 4 3 1
26:
Read 25 items
> barplot(or) # salah, yang ditampilkan data individual
```



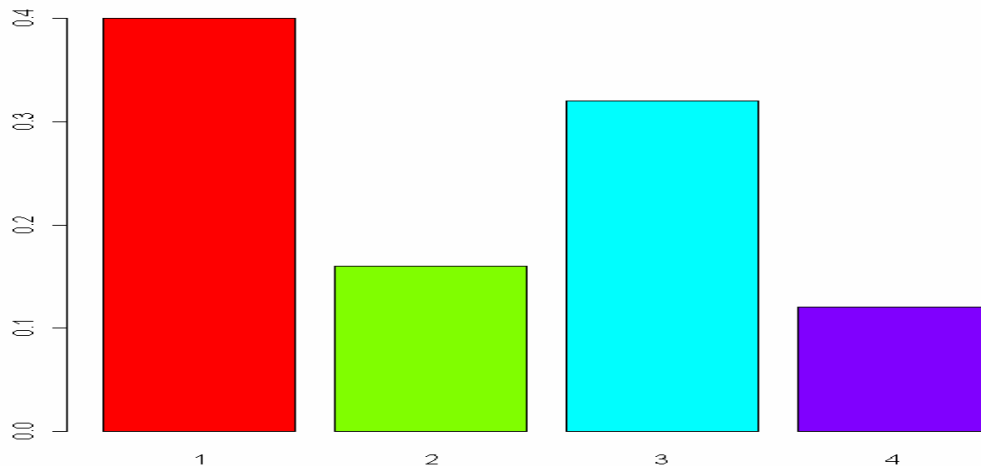
```
> barplot(table(or)) # benar, yang ditampilkan frekuensi
```



```
> barplot(table(or)/length(or),col=rainbow(4)) #benar, yang ditampilkan proporsi
```

Keterangan:

argumen `col=rainbow(4)` pada fungsi `barplot`, akan memberikan tampilan warna yang berbeda-beda untuk tiap batang.



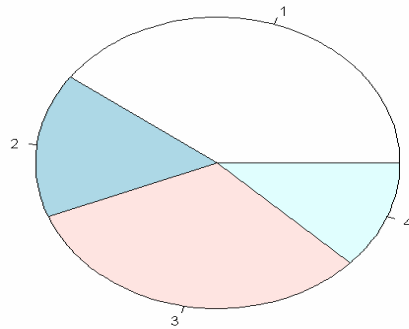
Tampaknya disini perintah `table(or)/length(or)` akan memberikan nilai tabel proporsional.

```
> table(or)/length(or)
or
 1    2    3    4
0.40 0.16 0.32 0.12
```

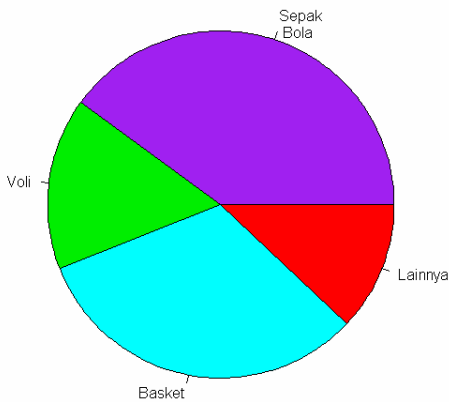
### **Diagram Roti (Pie Chart)**

Data yang sama dapat dianalisis dengan menggunakan diagram roti, dengan menggunakan fungsi `pie`. Berikut ini adalah sebuah teladan dalam penggunaan fungsi `pie`.

```
> pie(or.frek) # frekuensi data or disimpan pada objek or.frek
> pie(or.frek) # menampilkan diagram roti
```



```
> names(or.frek)<-c("Sepak\n Bola", "Voli","Basket","Lainnya") # beri nama
> pie(or.frek, col=c("purple","green2","cyan","red1") # tambah warna
```



Bila anda ingin mengetahui lebih jauh tentang fungsi barplot dan pie, gunakan perintah berikut.

```
> ?barplot
> ?pie
```

### **Data Numerik (Numerical data)**

Ada berbagai pilihan untuk memperagakan data numerik, antara lain ringkasan numerik, pemusatan (*center*) dan dispersi (*spread*).

## Ukuran numerik untuk pemusatan dan dispersi

Untuk menggambarkan distribusi data, perlu diketahui lebih dahulu dimana pemusatan datanya, dispersi datanya. Ukuran-ukuran itu dicari dengan menggunakan fungsi `mean`, `var`, `sd`, `median` atau dalam bentuk ringkasan seperti `summary` dan `fivenum`. Fungsi terakhir ini memberikan hasil ringkasan lima angka. R khususnya dalam package base menyediakan banyak fungsi terpasang (*built-in*). Fungsi `data()` digunakan untuk melihat semua data yang tersedia, seperti berikut ini.

```
> data() # melihat data terpasang apa saja yang tersedia
> data(faithful) # memanggil data faithful
> faithful # melihat isi dari objek data faithful
      eruptions waiting
1         3.600       79
2         1.800       54
.....      dst      .....
272        4.467       74
```

Untuk menampilkan object `eruptions`, gunakan perintah `faithful$eruptions`, perhatikan karakter `$` sebagai penyambung. Fungsi `attach()` digunakan agar dapat langsung memanggil variabel `eruptions` yang berada pada posisi di bawah `faithful`.

```
> eruptions # salah, objek tidak ditemukan
> faithful$eruptions # objek eruptions ada di bawah faithful
[1] 3.600 1.800 3.333 2.283 4.533 2.883 ... dst ...
> attach(faithful) # melakukan pencarian di bawah faithful
> eruptions # eruptions dapat dicari secara langsung
[1] 3.600 1.800 3.333 2.283 4.533 ... dst ...
> erup<-eruptions # menciptakan objek erup yang berisi data eruptions
> detach(faithful) # akhiri pencarian
```

Sekarang kita dapat bekerja pada objek `erup` saja walaupun objek `faithful` dihapus dari memori aktif.

```
> mean(erup)
[1] 3.487783
> mean(erup,.10) # rata-rata dg. memangkas 10% data di atas & di bawah
[1] 3.529807
> sd(erup)
[1] 1.141371
> summary(erup)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.600   2.163   4.000   3.488   4.454   5.100
> fivenum(erup) # min, engsel bawah, median, engsel atas, max
[1] 1.6000 2.1585 4.0000 4.4585 5.1000
```

Perhatikanlah perbedaan antara `summary` dan `fivenum`.

Median adalah quantile ke 50%, kuartil 1 adalah quantile 0.25, sedangkan kuartil 3 adalah quantile 0.75. Jadi quantile  $p$  adalah persentil ke  $100p\%$ . Fungsi `quantile` digunakan untuk menampilkan nilai persentil yang dimaksud.

```
> quantile(erup,0.25) # memberikan persentil ke 25%
```

```

    25%
2.16275
> quantile(erup,0.75) # memberikan persentil ke 75%
    75%
4.45425
> quantile(erup,0.50)
    50%
    4
> quantile(erup,0.15)
    15%
1.917
> quantile(erup,c(0.15,0.25,0.50,0.75,0.85)) # memperagakan sekaligus
    15%    25%    50%    75%    85%
1.91700 2.16275 4.00000 4.45425 4.60000

```

Simpangan rata-rata median (MAD) merupakan suatu ukuran dispersi yang resisten terhadap pencilan, rumusnya sebagai berikut

$$\text{median } |X_i - \text{median}(X)| \quad (1.4826)$$

Konstanta 1.4826 merupakan faktor pembanding terhadap simpangan baku normal.

Fungsi `mad ( )` mendapatkan nilai simpangan tersebut.

```

> mad(erup)
[1] 0.9510879

```

### Diagram dahan-dan-daun (*stem-and-leaf chart*)

Untuk melihat pola data, khususnya untuk data dengan jumlah sedikit biasa digunakan diagram dahan-dan-daun. Fungsi `stem ( )` memperagakan diagram ini.

Teladan: Data berikut ini adalah lamanya waktu (dalam menit) yang diperlukan untuk menunggu lift di UBINUS kampus anggrek, sehingga dapat digunakan.

2 3 16 23 14 12 4 13 2 0 0 0 6 28 31 14 4 8 2 5

```

> lift<-scan()
1: 2 3 16 23 14 12 4 13 2 0 0 0 6 28 31 14 4 8 2 5
21:
Read 20 items

```

Bila lupa nama persisnya dari fungsi `stem`, maka dapat dicari dengan menggunakan fungsi `apropos ( )`.

```

> apropos("stem")
[1] "stem"          "system"         "system.file"    "system.time"
> stem(lift)

```

The decimal point is 1 digit(s) to the right of the |

```

0 | 000222344568
1 | 23446
2 | 38
3 | 1
> stem(lift,scale=2)

```

The decimal point is 1 digit(s) to the right of the |

```
0 | 000222344
0 | 568
1 | 2344
1 | 6
2 | 3
2 | 8
3 | 1
```

### Membuat data numerik dalam bentuk kategorik

Kembali pada data erup, disini kita akan membaginya dalam 9 kelas (karena  $2^8 < \text{length(erup)} < 2^9$ ). Lebar kelasnya dipilih 0.4 (pembulatan ke atas dari  $\text{diff(range(erup))/9}$ ). Sehingga diperoleh kelas-kelas sebagai berikut.

```
[1.5, 1.9] (1.9, 2.3] (2.3, 2.7] (2.7, 3.1] (3.1, 3.5] (3.5, 3.9]
(3.9, 4.3] (4.3, 4.7] (4.7, 5.1]
```

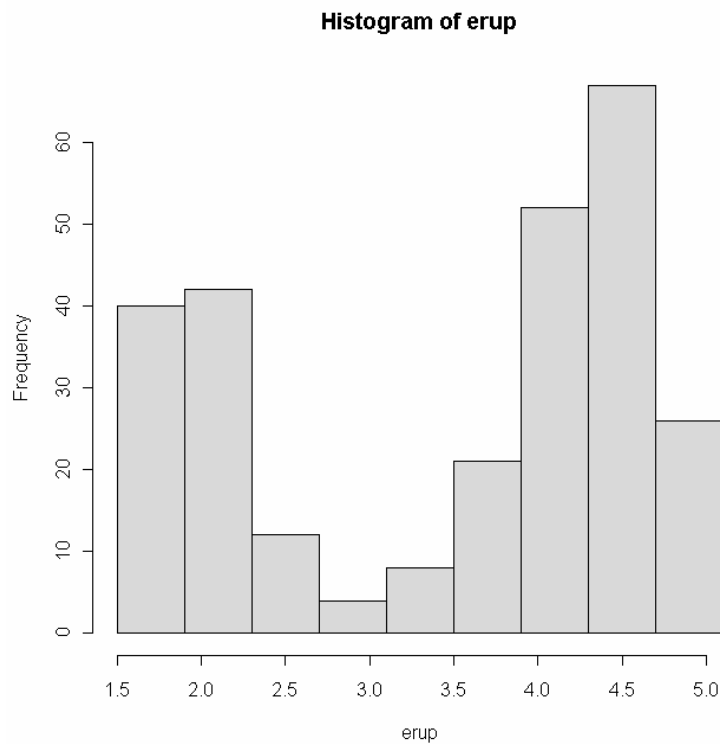
```
> erup.kat<-cut(erup,breaks=c(1.5,1.9,2.3,2.7,3.1,3.5,3.9,4.3,4.7,5.1))
> erup.kat
 [1] (3.5,3.9] (1.5,1.9] (3.1,3.5] (1.9,2.3] (4.3,4.7] (2.7,3.1] (4.3,4.7]
      ....      ....      ....      ....      dst      ....      ....      ....      ....
[267] (4.7,5.1] (3.9,4.3] (1.9,2.3] (4.3,4.7] (1.5,1.9] (4.3,4.7]
9 Levels: (1.5,1.9] (1.9,2.3] (2.3,2.7] (2.7,3.1] (3.1,3.5] ... (4.7,5.1]
```

```
> table(erup.kat)
erup.kat
(1.5,1.9] (1.9,2.3] (2.3,2.7] (2.7,3.1] (3.1,3.5] (3.5,3.9] (3.9,4.3] (4.3,4.7]
      40       42       12        4        8       21       52       67
(4.7,5.1]
      26
```

### Histogram

Bentuk grafik yang paling sering digunakan untuk memperagakan data numerik adalah histogram. Fungsi `hist()` menghasilkan histogram yang dimaksud.

```
> hist(erup,col=gray(0.8)) # kelas ditentukan secara otomatis
> # jika kelas ditentukan oleh user, misalkan seperti kelas yg telah
> # dicari sebelumnya maka ...
> hist(erup,breaks=c(1.5,1.9,2.3,2.7,3.1,3.5,3.9,4.3,4.7,5.1),col=gray(0.85))
```

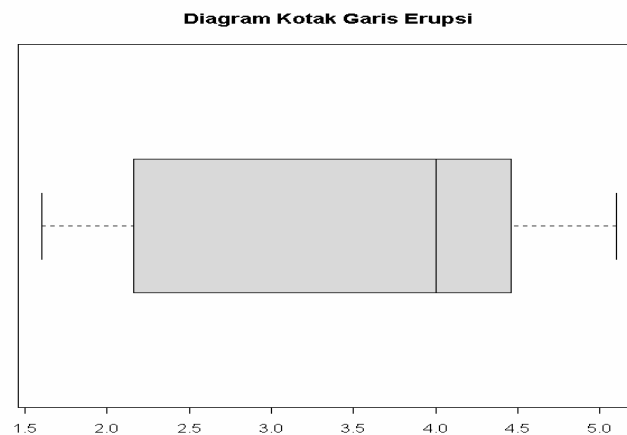


```
> hist(erup,col=gray(0.8),probability=TRUE) # memperagakan proporsi
> rug(jitter(erup)) # menambahkan tanda tick
```

### Diagram Kotak Garis (Boxplots)

Peragaan diagram kotak garis memberikan informasi secara cepat, mengenai kesimetrian data dan munculnya pencilan (*outliers*). Fungsi `boxplot()` memberikan peragaan diagram kotak garis.

```
> boxplot(erup,main="Diagram Kotak Garis Erupsi",horizontal=TRUE,
+ col=gray(0.85))
```



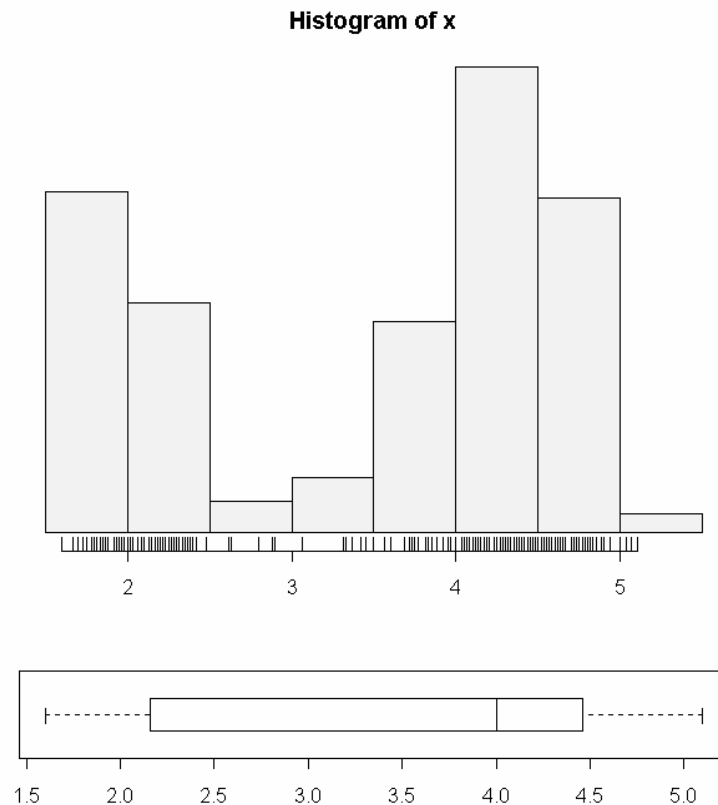


## Peragaan Histogram dan Kotak Garis secara bersamaan

Package Simple menyediakan fungsi `simple.hist.and.boxplot()` untuk memperagakan histogram dan kotak garis secara bersama-sama. Perintah `library()` adalah untuk melihat library apa saja yang telah di install. Perintah `library(Simple)` akan memanggil package Simple.

```
> library() # melihat package apa saja yg tersedia
> library(Simple) # memanggil package Simple
> apropos("simple") # fungsi simple apa saja yang tersedia
[1] "simple.chutes"          "simple.densityplot"
[3] "simple.densityplot.default" "simple.densityplot.formula"
[5] "simple.eda"             "simple.eda.ts"
[7] "simple.fancy.stripchart" "simple.freqpoly"
[9] "simple.hist.and.boxplot" "simple.lag"
[11] "simple.lm"              "simple.median.test"
[13] "simple.plot.hist.and.box" "simple.scatterplot"
..... dst .....

> simple.hist.and.boxplot(erup)
```



## Poligon Frekuensi

Poligon frekuensi menggambarkan garis-garis lurus yang menghubungkan puncak-puncak dari segi-empat pada histogram. Berikut ini adalah rangkaian perintah untuk membuat poligon frekuensi.

```
> his.erup<-hist(erup,col=gray(0.8))
> str(his.erup) # melihat apa saja yang dikandung oleh objek his.erup
List of 7
 $ breaks      : num [1:9] 1.5 2 2.5 3 3.5 4 4.5 5 5.5
 $ counts      : int [1:8] 55 37 5 9 34 75 54 3
 $ intensities: num [1:8] 0.4044 0.2721 0.0368 0.0662 0.2500 ...
 $ density     : num [1:8] 0.4044 0.2721 0.0368 0.0662 0.2500 ...
 $ mids        : num [1:8] 1.75 2.25 2.75 3.25 3.75 4.25 4.75 5.25
 $ xname       : chr "erup"
 $ equidist    : logi TRUE
 - attr(*, "class")= chr "histogram"
> lines(c(min(his.erup$breaks),his.erup$mids,max(his.erup$breaks)),
+ c(0,his.erup$counts,0),type="l") # menambahkan garis lurus
```

## **5. Data Acak (*Random Data*)**

Einstein berkata Tuhan tidak bermain dadu, tidak demikian halnya dengan R, ia dapat melakukannya.

Teladan: Berikut ini adalah percobaan melambungkan sebuah dadu setimbang sebanyak 10 kali.

```
> sample(1:6,10,replace=TRUE) # anda mungkin saja mendapatkan hasil berbeda
[1] 5 5 1 1 2 1 6 6 4 2
```

berikut ini adalah fungsi buatan untuk mendapatkan sampel dari pelambungan sebuah dadu sebanyak n kali.

```
> Dadu<-function(n) sample(1:6,n,replace=TRUE)
> Dadu(5) # melambungkan sebuah dadu sebanyak 5 kali
[1] 3 3 4 5 4
```

Jika percobaan pelambungan sebuah dadu sebanyak 10 kali dilakukan 100 kali, dan peubah acak x didefinisikan sebagai banyaknya mata 6 yang muncul pada setiap kali pelambungan. R dapat melakukannya sebagai berikut.

```
> dad6<-Dadu(10)==6
> x<-length(dad6[dad6==TRUE])
> for (j in 1:99) {dad6<-Dadu(10)==6
+ x6<-length(dad6[dad6==TRUE])
+ x<-c(x,x6)}
> x
[1] 2 1 1 3 1 3 1 0 1 1 2 1 0 1 0 1 4 0 1 2 1 2 3 1 1 0 2 1 1 1 0 3 1 3 2 0 1
[38] 4 1 1 0 1 2 1 1 3 2 0 3 3 0 2 1 4 1 1 2 2 1 3 1 2 1 4 2 1 1 3 1 3 1 3 0 2
[75] 3 1 1 1 2 0 1 2 2 2 1 1 1 4 3 1 0 2 2 1 1 4 2 4 1 1
```

Peluang  $x=0$  dapat dicari dengan pendekatan frekuensi relatif, adalah sebagai berikut.

```
> length(x[x==0])/length(x)
[1] 0.13
```

Hasil di atas dapat dibandingkan dengan peluang  $x=0$  untuk percobaan binomial dengan  $n=10$  dan  $p=1/6$ .

```
> ?dbinom # apa saja argumen fungsi ini?
> dbinom(x=0,10,1/6)
[1] 0.1615056
```

## **Latihan**

1. Sebuah percobaan melambungkan sebuah koin 3 kali. Peluang munculnya Angka ("A") adalah  $\frac{1}{4}$ , dan peluang munculnya Gambar ("G") adalah  $\frac{3}{4}$  untuk setiap kali pelambungan.
  - a. Perintahkanlah R untuk melakukan percobaan tersebut sebanyak 1000 kali dan  $x$  didefinisikan sebagai banyaknya "G" yang muncul.
  - b. Hitunglah frekuensi relatif untuk  $x \leq 2$ .
  - c. Bandingkan dengan nilai  $F(2)$  dari suatu percobaan binom dengan  $n=3$  dan  $p=3/4$ .

```
> sample(c("A", "G"), 3, prob=c(1/4, 3/4), replace=TRUE) # satu kali pelambungan
> pbinom(2, 3, 3/4) # nilai F(2) untuk binom n=3, p=3/4
```

## **Pembangkit Peubah Acak Uniform**

R dapat membangkitkan nilai-nilai peubah acak dengan fungsi `r+nama` sebaran. Misalkan untuk membangkitkan angka acak –yaitu nilai peubah acak yang berasal dari sebaran uniform (0,1)- dilakukan oleh fungsi `runif(n)`. Bentuk umum fungsi ini adalah `runif(n,min=0,max=1)`, jika `min` dan `max` tidak dinyatakan pada fungsi maka nilai `min=0` dan `max=1` yang digunakan.

```
> u<-runif(10) # bangkitkan angka acak sebanyak 10 buah
> runif(5,2,4) # bangkitkan u~uniform(2,4)
[1] 3.823740 3.881565 2.584486 2.782039 2.068949
```

## **Pembangkit Peubah Acak Normal**

Sebagaimana telah diketahui sebaran normal mempunyai dua parameter yaitu rata-rata  $\mu$  dan simpangan baku  $\sigma$ . Keduanya tersebut masing-masing merupakan parameter lokasi dan dispersi. Perintah R untuk membangkitkan peubah acak normal dengan rata-rata 100 dan simpangan baku 16 adalah.

```
> rnorm(1,100,16)
[1] 118.709
> rnorm(10,mean=280,sd=10) # bangkitkan 10 p.a x~N(280,10)
[1] 256.0841 294.6784 284.2276 300.4937 282.6516 280.7631 274.1319 280.6776
[9] 275.8532 271.9212
> rnorm(5) # bankitkan 5 peubah acak normal baku
[1] 1.6297930 -0.6989020 -0.3279155 1.6134212 -1.9865211
```

Teladan: Perintahkanlah R untuk mengambil 100 sampel yang berasal dari sebaran normal baku, kemudian buatlah histogram dari sampel tersebut dan gambarkan kurva normal baku pada histogram tersebut.

```
> x<-rnorm(100)
> hist(x,probability=TRUE,col="cyan3",
+ main="Histogram Sampel Normal Baku")
> curve(dnorm(x),add=TRUE)
```

Teladan: Pada sebaran diskret seperti binomial nampaknya kurang tepat menggambarkan sampelnya dengan menggunakan histogram. Perintah R berikut menggunakan fungsi `plot()` untuk menggambarkan sampel binomial  $n=5$ ,  $p=.25$ .

```
> n<-5; p<-1/4
> x<-rbinom(100,n,p)
> plot(table(x)/length(x),type="h",lwd=8,
+ col="blue3",main="Grafik Sebaran Sampling Binomial n=5,p=.25")
```

Bandingkan dengan grafik sebaran binomialnya yaitu

```
> nilx<-0:n
> plot(nilx,dbinom(nilx,n,p),type="h",lwd=8,
+ col="blue3",main="Grafik Sebaran Binomial n=5,p=.25",
+ ylab="Peluang",xlab="nilai x")
```

Pada kebanyakan buku-buku teks statistik selalu dilengkapi dengan tabel normal baku  $z$ . Tabel itu tidak diperlukan lagi karena R menyediakan fungsi yang canggih untuk medapatkan nilai  $z$ , dengan menggunakan fungsi `qnorm()` yaitu fungsi untuk mendapatkan nilai  $z$  jika peluang ekor kirinya diketahui.

Teladan: Perintahkanlah R untuk mendapatkan  $z$ , sehingga peluang ekor kirinya adalah 0.05 atau  $F(z)=0.05$ .

```
> qnorm(0.05)
[1] -1.644854
```

Pada berbagai sebaran normal dapat kita cari nilai quantilenya, sebagai berikut.

```
> qnorm(c(0.025,0.25,0.50,0.75,0.975),mean=100,sd=10)
[1] 80.40036 93.25510 100.00000 106.74490 119.59964
```

## **Latihan**

Bila 1000 sampel masing-masing berukuran 100 diambil dari sebaran eksponensial dengan  $\text{rate}=2$ . Kemudian setiap sampel dihitung rata-ratanya. Gambarkan histogram untuk rata-rata tersebut, bandingkan dengan kurva normal dengan  $\text{mean}=1/2$  dan  $\text{sigma}=1/20$ . Apakah dalil limit pusat berlaku? Perintahkan R untuk memeriksanya.

petunjuk:

```
> rexp(100,2.) # sebuah sampel berukuran 100 dari  $x \sim \text{Exp}(2.0)$ 
```

## 6. Referensi

Verzani, John. 2002. simpleR – Using R for Introductory Statistics.

URL: <http://www.math.csi.cuny.edu/Statistics/R/simpleR>

Venable, W. N. and D.M. Smith. 2003. An Introduction to R

URL: <http://www.r-project.org>

Maindonald, J.H. 2001. Using R for Data Analysis and Graphics An Introduction. Australian National University.

Paradis, Emanuel. 2002. R for Beginners. Institut des Sciences de l'Evolution. Universite Montpellier II. France.