

EMT untuk Statistika

Dalam buku catatan ini, kami menunjukkan plot, pengujian, dan distribusi statistik utama dalam Euler.

Mari kita mulai dengan beberapa statistik deskriptif. Ini bukan pengantar statistik. Jadi, Anda mungkin memerlukan beberapa latar belakang untuk memahami detailnya.

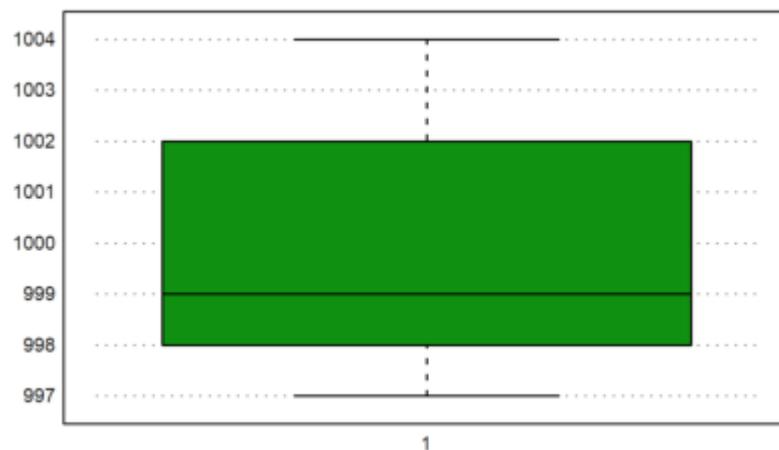
Asumsikan pengukuran berikut. Kami ingin menghitung nilai rata-rata dan deviasi standar yang diukur.

```
>M=[1000,1004,998,997,1002,1001,998,1004,998,997]; ...  
median(M), mean(M), dev(M),
```

```
999  
999.9  
2.72641400622
```

Kita dapat membuat diagram kotak dan kumis untuk data tersebut. Dalam kasus kita, tidak ada outlier.

```
>aspect(1.75); boxplot(M):
```



Kami menghitung probabilitas bahwa suatu nilai lebih besar dari 1005, dengan asumsi nilai terukur dari distribusi normal.

Semua fungsi untuk distribusi dalam Euler diakhiri dengan ...dis dan menghitung distribusi probabilitas kumulatif (CPF).

$$\text{normaldis}(x,m,d) = \int_{-\infty}^x \frac{1}{d\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{t-m}{d}\right)^2} dt.$$

Kami mencetak hasil dalam % dengan akurasi 2 digit menggunakan fungsi cetak.

```
>print((1-normaldis(1005,mean(M),dev(M)))*100,2,unit=" %")
```

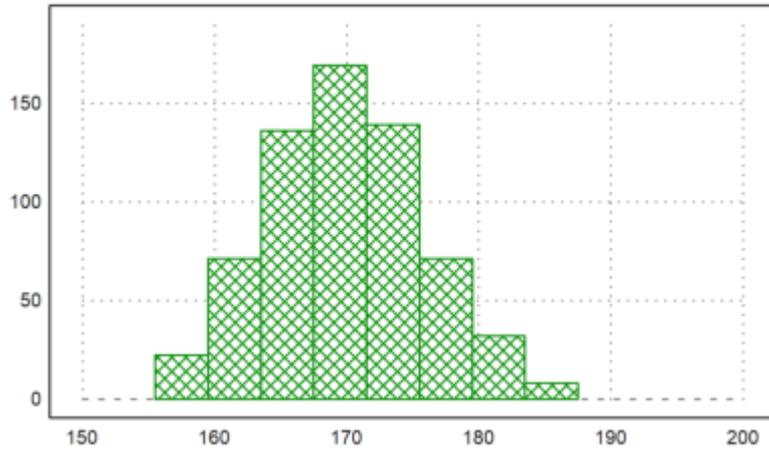
```
3.07 %
```

Untuk contoh berikutnya, kami mengasumsikan jumlah pria berikut dalam rentang ukuran tertentu.

```
>r=155.5:4:187.5; v=[22,71,136,169,139,71,32,8];
```

Berikut adalah plot distribusinya.

```
>plot2d(r,v,a=150,b=200,c=0,d=190,bar=1,style="\/"):
```



Kita dapat memasukkan data mentah tersebut ke dalam tabel.

Tabel adalah metode untuk menyimpan data statistik. Tabel kita harus berisi tiga kolom: Awal rentang, akhir rentang, jumlah orang dalam rentang.

Tabel dapat dicetak dengan tajuk. Kita menggunakan vektor string untuk mengatur tajuk.

```
>T:=r[1:8]' | r[2:9]' | v'; writetable(T,labc=["BB","BA","Frek"])
```

BB	BA	Frek
155.5	159.5	22
159.5	163.5	71
163.5	167.5	136
167.5	171.5	169
171.5	175.5	139
175.5	179.5	71
179.5	183.5	32
183.5	187.5	8

Jika kita memerlukan nilai rata-rata dan statistik ukuran lainnya, kita perlu menghitung titik tengah rentang. Kita dapat menggunakan dua kolom pertama tabel kita untuk ini.

Simbol "|" digunakan untuk memisahkan kolom, fungsi "writetable" digunakan untuk menulis tabel, dengan opsi "labc" untuk menentukan tajuk kolom.

```
>(T[,1]+T[,2])/2 // the midpoint of each interval
```

```
157.5  
161.5  
165.5  
169.5  
173.5  
177.5  
181.5  
185.5
```

Namun lebih mudah untuk melipat rentang dengan vektor [1/2,1/2].

```
>M=fold(r,[0.5,0.5])
```

```
[157.5, 161.5, 165.5, 169.5, 173.5, 177.5, 181.5, 185.5]
```

Sekarang kita dapat menghitung rata-rata dan deviasi sampel dengan frekuensi yang diberikan.

```
>{m,d}=meandev(M,v); m, d,
```

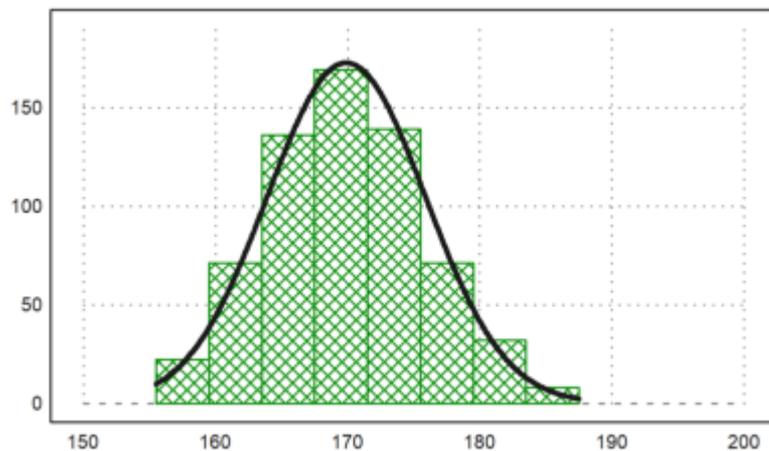
```
169.901234568  
5.98912964449
```

Mari kita tambahkan distribusi normal nilai-nilai tersebut ke diagram batang di atas. Rumus untuk distribusi normal dengan rata-rata m dan simpangan baku d adalah:

$$y = \frac{1}{d\sqrt{2\pi}} e^{-\frac{(x-m)^2}{2d^2}}.$$

Karena nilainya berada di antara 0 dan 1, untuk memplotnya pada diagram batang, nilainya harus dikalikan dengan 4 kali jumlah total data.

```
>plot2d("qnormal(x,m,d)*sum(v)*4", ...  
  xmin=min(r),xmax=max(r),thickness=3,add=1):
```



Tabel

Di direktori buku catatan ini, Anda akan menemukan berkas dengan tabel. Data tersebut merupakan hasil survei. Berikut adalah empat baris pertama berkas tersebut. Data tersebut berasal dari buku daring Jerman "Einführung in die Statistik mit R" karya A. Handl.

```
>printfile("table.dat",4);
```

```
Could not open the file  
table.dat  
for reading!  
Try "trace errors" to inspect local variables after errors.  
printfile:  
  open(filename,"r");
```

Tabel berisi 7 kolom angka atau token (string). Kita ingin membaca tabel dari file. Pertama, kita menggunakan terjemahan kita sendiri untuk token.

Untuk ini, kita mendefinisikan set token. Fungsi `strtokens()` mendapatkan vektor string token dari string yang diberikan.

```
>mf=["m","f"]; yn=["y","n"]; ev:=strtokens("g vg m b vb");
```

Sekarang kita baca tabel dengan terjemahan ini.

Argumen tok2, tok4, dst. adalah terjemahan kolom-kolom tabel. Argumen ini tidak ada dalam daftar parameter readtable(), jadi Anda perlu menyediakannya dengan ":=".

```
>{MT,hd}=readtable("table.dat",tok2:=mf,tok4:=yn,tok5:=ev,tok7:=yn);
```

```
Could not open the file
table.dat
for reading!
Try "trace errors" to inspect local variables after errors.
readtable:
  if filename!=none then open(filename,"r"); endif;
```

```
>load over statistics;
```

Untuk mencetak, kita perlu menentukan set token yang sama. Kita cetak empat baris pertama saja.

```
>writetable(MT[1:10],labc=hd,wc=5,tok2:=mf,tok4:=yn,tok5:=ev,tok7:=yn);
```

```
MT is not a variable!
Error in:
writetable(MT[1:10],labc=hd,wc=5,tok2:=mf,tok4:=yn,tok5:=ev,to ...
      ^
```

Titik "." mewakili nilai yang tidak tersedia.

Jika kita tidak ingin menentukan token untuk penerjemahan terlebih dahulu, kita hanya perlu menentukan kolom mana yang berisi token dan bukan angka.

```
>ctok=[2,4,5,7]; {MT,hd,tok}=readtable("table.dat",ctok=ctok);
```

```
Could not open the file
table.dat
for reading!
Try "trace errors" to inspect local variables after errors.
readtable:
  if filename!=none then open(filename,"r"); endif;
```

Fungsi readtable() sekarang mengembalikan serangkaian token.

```
>tok
```

```
m
n
f
y
g
vg
```

Tabel berisi entri dari berkas dengan token yang diterjemahkan ke angka.

String khusus NA="." ditafsirkan sebagai "Tidak Tersedia", dan mendapatkan NAN (bukan angka) dalam tabel. Terjemahan ini dapat diubah dengan parameter NA, dan NAval.

```
>MT[1]
```

```
[1, 1, 30, 2, NAN, 1.8, 2]
```

Berikut ini adalah isi tabel dengan angka yang belum diterjemahkan.

```
>writetable(MT,wc=5)
```

1	1	30	2	.	1.8	2
2	3	23	4	5	1.8	2
3	3	26	4	5	1.8	4
4	1	33	2	.	2.8	2
5	1	37	2	.	1.8	2
6	1	28	4	5	2.8	4
7	3	31	4	6	2.8	2
8	1	23	2	.	0.8	2
9	3	24	4	6	1.8	4
10	1	26	2	.	1.8	2
11	3	23	4	6	1.8	4
12	1	32	4	5	1.8	2
13	1	29	4	6	1.8	4
14	3	25	4	5	1.8	4
15	3	31	4	5	0.8	2
16	1	26	4	5	2.8	2
17	1	37	2	.	3.8	2
18	1	38	4	5	.	2
19	3	29	2	.	3.8	2
20	3	28	4	6	1.8	2
21	3	28	4	1	2.8	4
22	3	28	4	6	1.8	4
23	3	38	4	5	2.8	2
24	3	27	4	1	1.8	4
25	1	27	2	.	2.8	4

Demi kenyamanan, Anda dapat memasukkan output readtable() ke dalam daftar.

```
>Table={{readtable("table.dat",ctok=ctok)}};
```

Dengan menggunakan kolom token yang sama dan token yang dibaca dari berkas, kita dapat mencetak tabel. Kita dapat menentukan ctok, tok, dll. atau menggunakan daftar Tabel.

```
>writetable(Table,ctok=ctok,wc=5);
```

Person	Sex	Age	Titanic	Evaluation	Tip	Problem
1	m	30	n	.	1.8	n
2	f	23	y	g	1.8	n
3	f	26	y	g	1.8	y
4	m	33	n	.	2.8	n
5	m	37	n	.	1.8	n
6	m	28	y	g	2.8	y
7	f	31	y	vg	2.8	n
8	m	23	n	.	0.8	n
9	f	24	y	vg	1.8	y
10	m	26	n	.	1.8	n
11	f	23	y	vg	1.8	y
12	m	32	y	g	1.8	n
13	m	29	y	vg	1.8	y
14	f	25	y	g	1.8	y
15	f	31	y	g	0.8	n
16	m	26	y	g	2.8	n
17	m	37	n	.	3.8	n
18	m	38	y	g	.	n
19	f	29	n	.	3.8	n
20	f	28	y	vg	1.8	n
21	f	28	y	m	2.8	y
22	f	28	y	vg	1.8	y
23	f	38	y	g	2.8	n

24	f	27	y	m	1.8	y
25	m	27	n	.	2.8	y

Fungsi `tablecol()` mengembalikan nilai kolom tabel, melewati baris mana pun dengan nilai `NAN("."` dalam file), dan indeks kolom, yang berisi nilai-nilai ini.

```
>{c,i}=tablecol(MT,[5,6]);
```

Kita dapat menggunakan ini untuk mengekstrak kolom dari tabel untuk tabel baru.

```
>j=[1,5,6]; writetable(MT[i,j],labr=hd[j],ctok=[2],tok=tok)
```

Person	Evaluation	Tip
2	g	1.8
3	g	1.8
6	g	2.8
7	vg	2.8
9	vg	1.8
11	vg	1.8
12	g	1.8
13	vg	1.8
14	g	1.8
15	g	0.8
16	g	2.8
20	vg	1.8
21	m	2.8
22	vg	1.8
23	g	2.8
24	m	1.8

Tentu saja, kita perlu mengekstrak tabel itu sendiri dari daftar `Table` dalam kasus ini.

```
>MT=Table[1];
```

Tentu saja, kita juga dapat menggunakannya untuk menentukan nilai rata-rata kolom atau nilai statistik lainnya.

```
>mean(tablecol(MT,6))
```

```
2.175
```

Fungsi `getstatistics()` mengembalikan elemen dalam vektor dan jumlahnya. Kita menerapkannya pada nilai "m" dan "f" di kolom kedua tabel kita.

```
>{xu,count}=getstatistics(tablecol(MT,2)); xu, count,
```

```
[1, 3]
[12, 13]
```

Kita dapat mencetak hasilnya di tabel baru.

```
>writetable(count',labr=tok[xu])
```

m	12
f	13

f	y
f	y

Dengan `getstatistics()`, kita juga dapat menghubungkan jumlah pada dua kolom tabel satu sama lain.

```
>MT24=tablecol(MT,[2,4]); ...
{xu1,xu2,count}=getstatistics(MT24[1],MT24[2]); ...
writetable(count,labr=tok[xu1],labc=tok[xu2])
```

	n	y
m	7	5
f	1	12

Suatu tabel dapat ditulis ke dalam suatu berkas.

```
>filename="test.dat"; ...
writetable(count,labr=tok[xu1],labc=tok[xu2],file=filename);
```

Lalu kita dapat membaca tabel dari berkas tersebut.

```
>{MT2,hd,tok2,hdr}=readtable(filename,>clabs,>rlabs); ...
writetable(MT2,labr=hdr,labc=hd)
```

	n	y
m	7	5
f	1	12

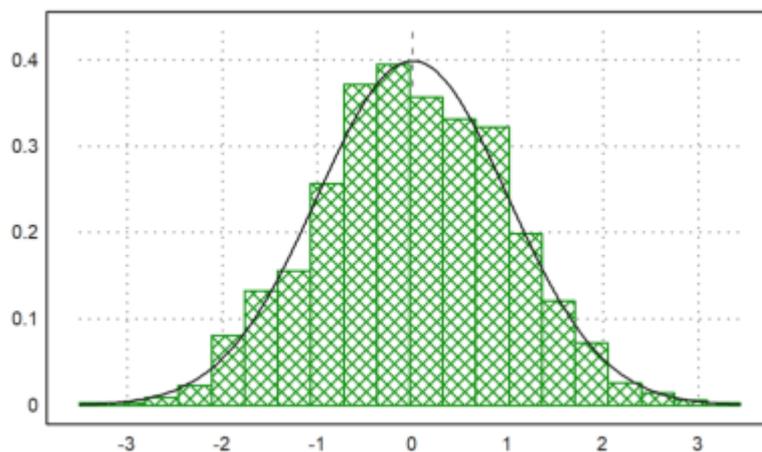
Dan hapus berkasnya.

```
>fileremove(filename);
```

Distribusi

Dengan `plot2d`, ada metode yang sangat mudah untuk memplot distribusi data eksperimen.

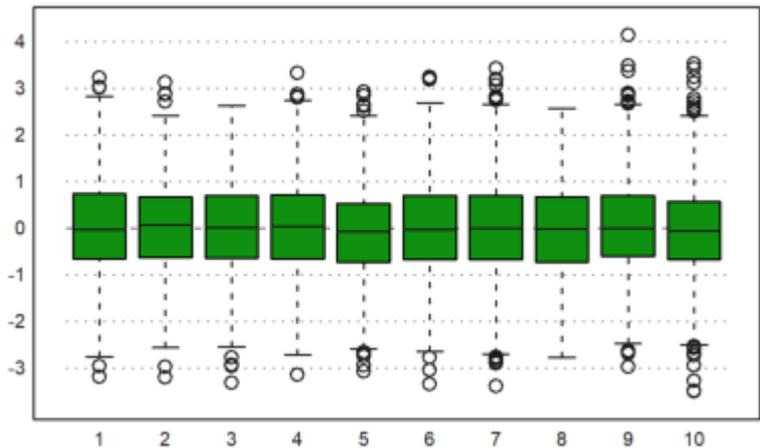
```
>p=normal(1,1000); //1000 random normal-distributed sample p
>plot2d(p,distribution=20,style="\"); // plot the random sample p
>plot2d("qnormal(x,0,1)",add=1): // add the standard normal distribution plot
```



Harap perhatikan perbedaan antara diagram batang (sampel) dan kurva normal (distribusi riil). Masukkan kembali ketiga perintah tersebut untuk melihat hasil sampel lainnya.

Berikut ini adalah perbandingan 10 simulasi dari 1000 nilai yang didistribusikan secara normal menggunakan apa yang disebut diagram kotak. Diagram ini menunjukkan median, kuartil 25% dan 75%, nilai minimal dan maksimal, dan outlier.

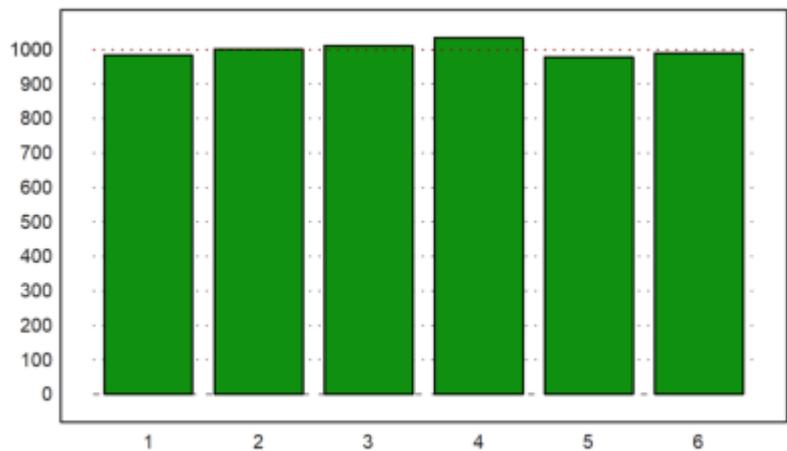
```
>p=normal(10,1000); boxplot(p):
```



Untuk menghasilkan bilangan bulat acak, Euler memiliki intrandom. Mari kita simulasikan lemparan dadu dan plot distribusinya.

Kita menggunakan fungsi getmultiplicities(v,x), yang menghitung seberapa sering elemen v muncul di x. Kemudian kita plot hasilnya menggunakan columnplot().

```
>k=intrandom(1,6000,6); ...
columnplot(getmultiplicities(1:6,k)); ...
ygrid(1000,color=red):
```



Sementara intrandom(n,m,k) mengembalikan bilangan bulat yang terdistribusi seragam dari 1 hingga k, dimungkinkan untuk menggunakan distribusi bilangan bulat lain yang diberikan dengan randpint().

Dalam contoh berikut, probabilitas untuk 1,2,3 masing-masing adalah 0,4,0,1,0,5.

```
>randpint(1,1000,[0.4,0.1,0.5]); getmultiplicities(1:3,%)
```

```
[378, 102, 520]
```

Euler dapat menghasilkan nilai acak dari lebih banyak distribusi. Lihat referensinya.

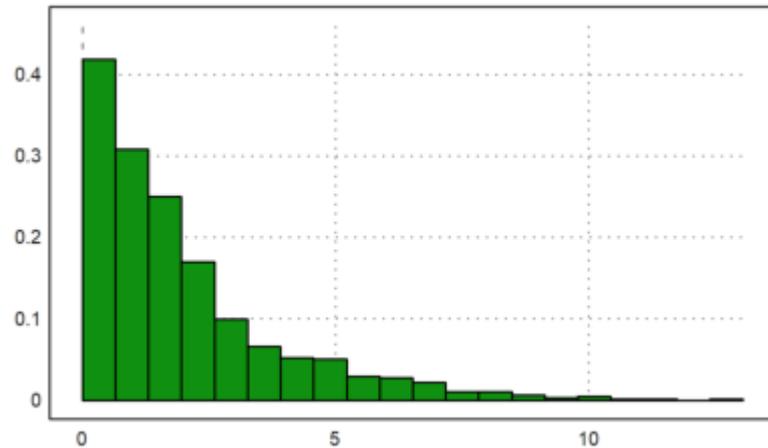
Misalnya, kita coba distribusi eksponensial. Variabel acak kontinu X dikatakan memiliki distribusi eksponensial, jika PDF-nya diberikan oleh

$$f_X(x) = \lambda e^{-\lambda x}, \quad x > 0, \quad \lambda > 0,$$

dengan parameter

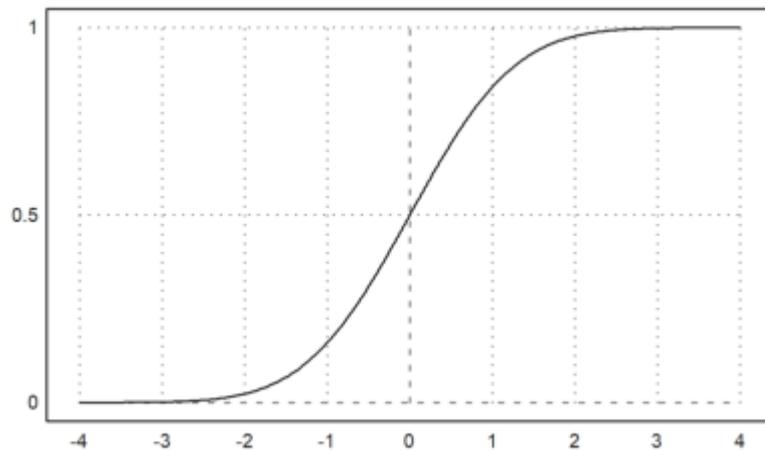
$\lambda = \frac{1}{\mu}$, μ adalah mean, dan dilambangkan dengan $X \sim \text{Ekspensial}(\lambda)$.

```
>plot2d(randexponential(1,1000,2),>distribution):
```



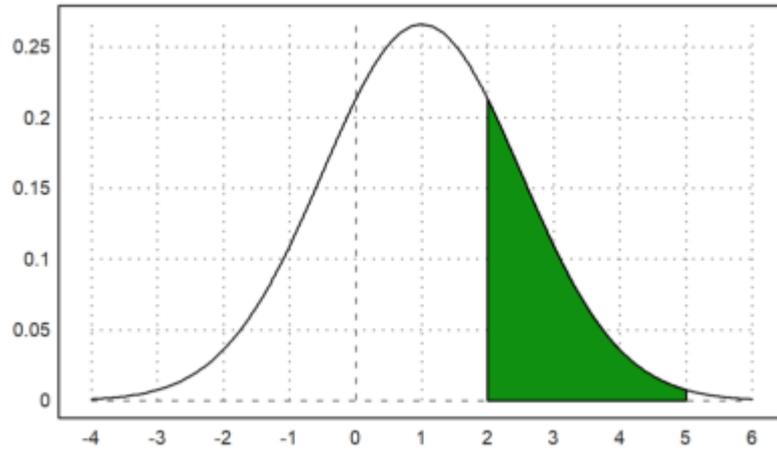
Untuk banyak distribusi, Euler dapat menghitung fungsi distribusi dan inversnya.

```
>plot2d("normaldis",-4,4):
```



Berikut ini adalah salah satu cara untuk memplot kuantil.

```
>plot2d("qnormal(x,1,1.5)",-4,6); ...
plot2d("qnormal(x,1,1.5)",a=2,b=5,>add,>filled):
```



lateks: $\text{normaldis}(x,m,d)=\int_{-\infty}^x \frac{1}{d\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{t-m}{d}\right)^2} dt$.

Peluang untuk berada di area hijau adalah sebagai berikut.

```
>normaldis(5,1,1.5)-normaldis(2,1,1.5)
```

```
0.248662156979
```

Hal ini dapat dihitung secara numerik dengan integral berikut.

$$\int_2^5 \frac{1}{1.5\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-1}{1.5}\right)^2} dx.$$

```
>gauss("qnormal(x,1,1.5)",2,5)
```

```
0.248662156979
```

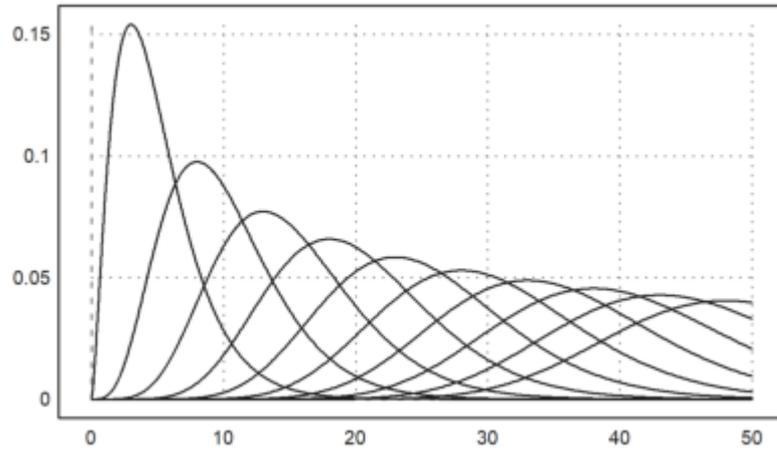
Mari kita bandingkan distribusi binomial dengan distribusi normal dengan nilai rata-rata dan deviasi yang sama. Fungsi `invbindis()` menyelesaikan interpolasi linier antara nilai integer.

```
>invbindis(0.95,1000,0.5), invnormaldis(0.95,500,0.5*sqrt(1000))
```

```
525.516721219
526.007419394
```

Fungsi `qdis()` adalah kerapatan distribusi chi-kuadrat. Seperti biasa, Euler memetakan vektor ke fungsi ini. Jadi, kita memperoleh plot semua distribusi chi-kuadrat dengan derajat 5 hingga 30 dengan mudah dengan cara berikut.

```
>plot2d("qchidis(x,(5:5:50)')",0,50):
```



Euler memiliki fungsi yang akurat untuk mengevaluasi distribusi. Mari kita periksa `chidis()` dengan integral.

Penamaannya mencoba agar konsisten. Misalnya,

- distribusi chi-kuadrat adalah `chidis()`,
- fungsi inversnya adalah `invchidis()`,
- densitasnya adalah `qchidis()`.

Komplemen distribusi (ekor atas) adalah `chicdis()`.

```
>chidis(1.5,2), integrate("qchidis(x,2)",0,1.5)
```

```
0.527633447259
0.527633447259
```

Distribusi Diskrit

Untuk menentukan distribusi diskrit Anda sendiri, Anda dapat menggunakan metode berikut.

Pertama, kita tetapkan fungsi distribusi.

```
>wd = 0 | ((1:6)+[-0.01,0.01,0,0,0,0])/6
```

```
[0, 0.165, 0.335, 0.5, 0.666667, 0.833333, 1]
```

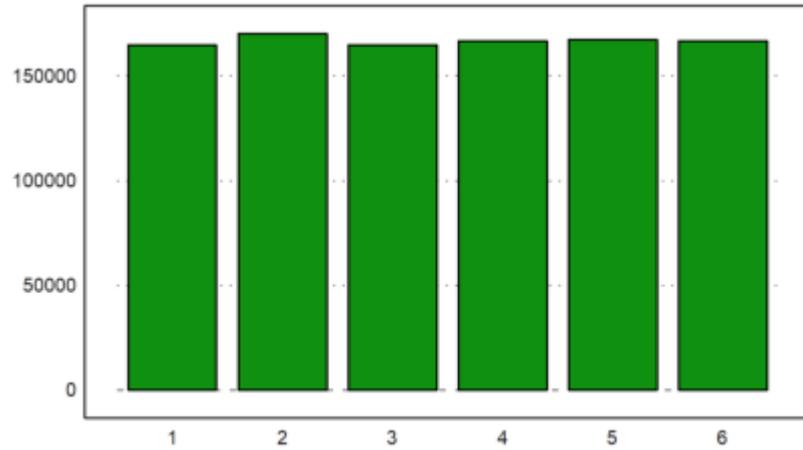
Artinya adalah bahwa dengan probabilitas $wd[i+1]-wd[i]$ kita menghasilkan nilai acak i .

Ini hampir merupakan distribusi seragam. Mari kita definisikan generator angka acak untuk ini. Fungsi `find(v,x)` menemukan nilai x dalam vektor v . Fungsi ini juga berfungsi untuk vektor x .

```
>function wrongdice (n,m) := find(wd,random(n,m))
```

Kesalahannya begitu halus sehingga kita hanya melihatnya pada pengulangan yang sangat banyak.

```
>columnplot(getmultiplicities(1:6,wrongdice(1,1000000))):
```



Berikut ini adalah fungsi sederhana untuk memeriksa distribusi seragam nilai 1...K dalam v. Kita terima hasilnya, jika untuk semua frekuensi

$$\left| f_i - \frac{1}{K} \right| < \frac{\delta}{\sqrt{n}}$$

```
>function checkrandom (v, delta=1) ...
  K=max(v); n=cols(v);
  fr=getfrequencies(v,1:K);
  return max(fr/n-1/K)<delta/sqrt(n);
endfunction
```

Memang fungsi tersebut menolak distribusi seragam.

```
>checkrandom(wrongdice(1,1000000))
```

0

Dan menerima generator acak bawaan.

```
>checkrandom(inrandom(1,1000000,6))
```

1

Kita dapat menghitung distribusi binomial. Pertama ada binomialsum(), yang mengembalikan probabilitas i atau kurang dari n kali percobaan.

```
>bindis(410,1000,0.4)
```

0.751401349654

Fungsi Beta terbalik digunakan untuk menghitung interval kepercayaan Clopper-Pearson untuk parameter p. Level default adalah alpha.

Arti dari interval ini adalah jika p berada di luar interval, hasil yang diamati sebesar 410 dalam 1000 adalah langka.

```
>clopperpearson(410,1000)
```

[0.37932, 0.441212]

Perintah berikut adalah cara langsung untuk mendapatkan hasil di atas. Namun untuk n yang besar, penjumlahan langsung tidak akurat dan lambat.

```
>p=0.4; i=0:410; n=1000; sum(bin(n,i)*p^i*(1-p)^(n-i))
```

```
0.751401349655
```

Omong-omong, `invbinsum()` menghitung kebalikan dari `binomialsum()`.

```
>invbindis(0.75,1000,0.4)
```

```
409.932733047
```

Dalam Bridge, kita mengasumsikan 5 kartu yang beredar (dari 52) dalam dua tangan (26 kartu). Mari kita hitung probabilitas distribusi yang lebih buruk dari 3:2 (misalnya 0:5, 1:4, 4:1 atau 5:0).

```
>2*hypergeomsum(1,5,13,26)
```

```
0.321739130435
```

Ada juga simulasi distribusi multinomial.

```
>randmultinomial(10,1000,[0.4,0.1,0.5])
```

381	100	519
376	91	533
417	80	503
440	94	466
406	112	482
408	94	498
395	107	498
399	96	505
428	87	485
400	99	501

Merencanakan Data

Untuk merencanakan data, kami mencoba hasil pemilu Jerman sejak 1990, yang diukur dalam jumlah kursi.

```
>BW := [ ...  
1990,662,319,239,79,8,17; ...  
1994,672,294,252,47,49,30; ...  
1998,669,245,298,43,47,36; ...  
2002,603,248,251,47,55,2; ...  
2005,614,226,222,61,51,54; ...  
2009,622,239,146,93,68,76; ...  
2013,631,311,193,0,63,64];
```

Untuk para pihak, kami menggunakan serangkaian nama.

```
>P:=["CDU/CSU","SPD","FDP","Gr","Li"];
```

Mari kita cetak persentasenya dengan baik.

Pertama-tama kita ekstrak kolom-kolom yang diperlukan. Kolom 3 hingga 7 adalah kursi masing-masing partai, dan kolom 2 adalah jumlah total kursi. Kolom 3 adalah tahun pemilihan.

```
>BT:=BW[,3:7]; BT:=BT/sum(BT); YT:=BW[,1]';
```

Kemudian kami mencetak statistik dalam bentuk tabel. Kami menggunakan nama sebagai tajuk kolom, dan tahun sebagai tajuk untuk baris. Lebar default untuk kolom adalah $wc=10$, tetapi kami lebih suka keluaran yang lebih padat. Kolom akan diperluas untuk label kolom, jika perlu.

```
>writetable(BT*100,wc=6,dc=0,>fixed,labc=P,labr=YT)
```

	CDU/CSU	SPD	FDP	Gr	Li
1990	48	36	12	1	3
1994	44	38	7	7	4
1998	37	45	6	7	5
2002	41	42	8	9	0
2005	37	36	10	8	9
2009	38	23	15	11	12
2013	49	31	0	10	10

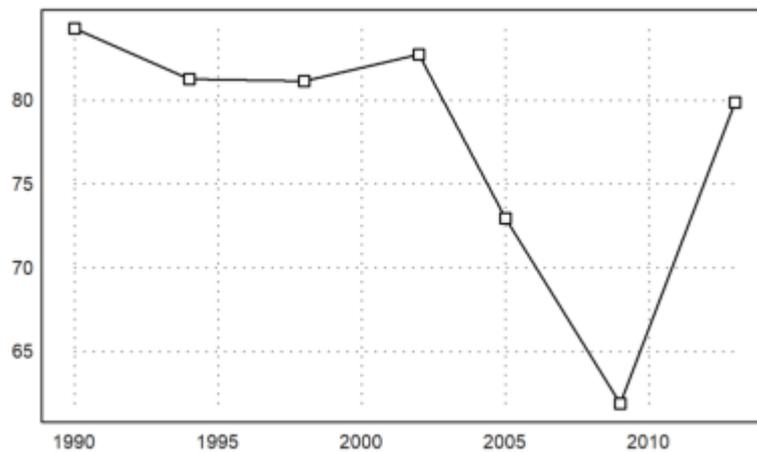
Perkalian matriks berikut ini mengekstrak jumlah persentase dari dua partai besar yang menunjukkan bahwa partai-partai kecil telah memperoleh dukungan di parlemen hingga tahun 2009.

```
>BT1:=(BT.[1;1;0;0;0])*100
```

```
[84.29, 81.25, 81.1659, 82.7529, 72.9642, 61.8971, 79.8732]
```

Ada juga plot statistik sederhana. Kita menggunakannya untuk menampilkan garis dan titik secara bersamaan. Alternatifnya adalah memanggil `plot2d` dua kali dengan `>add`.

```
>statplot(YT,BT1,"b"):
```



Tentukan beberapa warna untuk setiap pihak.

```
>CP:=[rgb(0.5,0.5,0.5),red,yellow,green,rgb(0.8,0,0)];
```

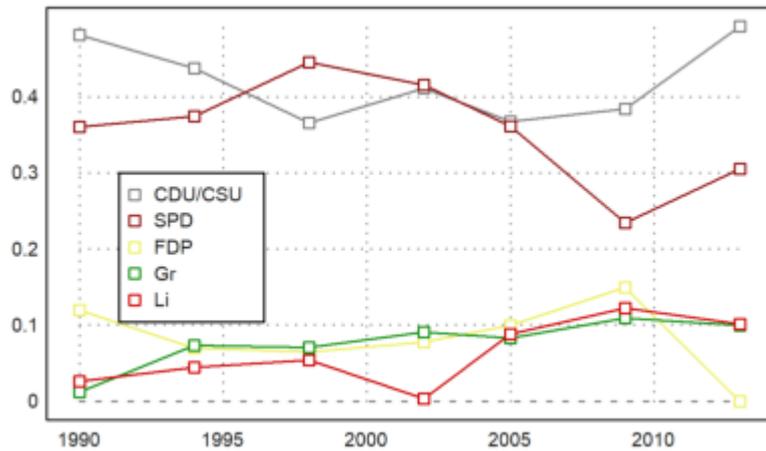
Sekarang kita dapat memetakan hasil pemilu 2009 dan perubahannya ke dalam satu plot menggunakan gambar. Kita dapat menambahkan vektor kolom ke setiap plot.

```
>figure(2,1); ...  
figure(1); columnsplot(BW[6,3:7],P,color=CP); ...  
figure(2); columnsplot(BW[6,3:7]-BW[5,3:7],P,color=CP); ...  
figure(0):
```



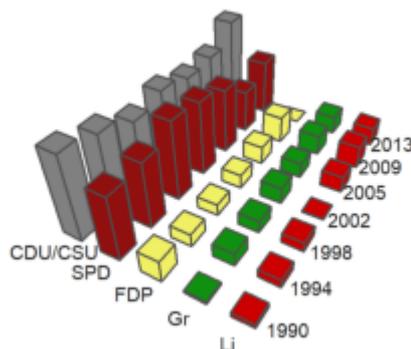
Plot data menggabungkan baris-baris data statistik dalam satu plot.

```
>J:=BW[,1]'; DP:=BW[,3:7]'; ...
  dataplot(YT,BT',color=CP); ...
  labelbox(P,colors=CP,styles="[]",>points,w=0.2,x=0.3,y=0.4):
```



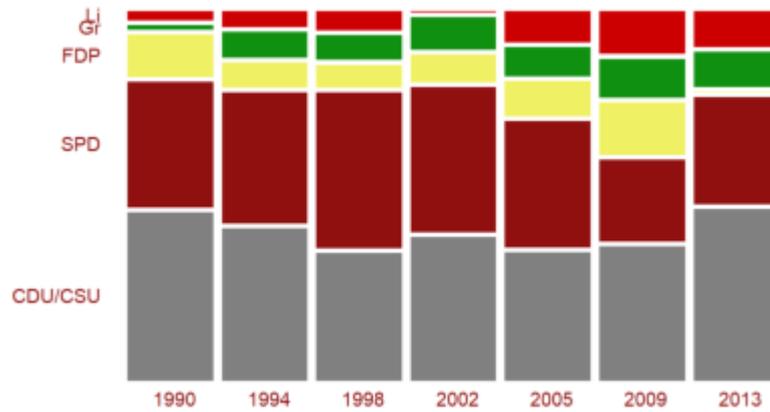
Plot kolom 3D menunjukkan baris data statistik dalam bentuk kolom. Kami memberikan label untuk baris dan kolom. Angle adalah sudut pandang.

```
>columnplot3d(BT,scols=P,srows=YT, ...
  angle=30°,ccols=CP):
```



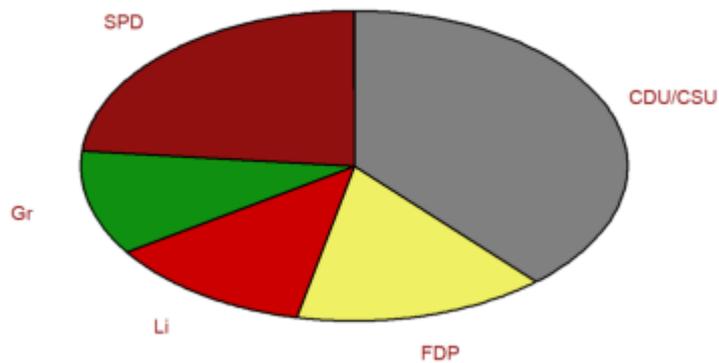
Representasi lainnya adalah plot mosaik. Perhatikan bahwa kolom-kolom plot mewakili kolom-kolom matriks di sini. Karena panjang label CDU/CSU, kami mengambil jendela yang lebih kecil dari biasanya.

```
>shrinkwindow(>smaller); ...  
mosaicplot(BT',srows=YT,scols=P,color=CP,style="#"); ...  
shrinkwindow():
```



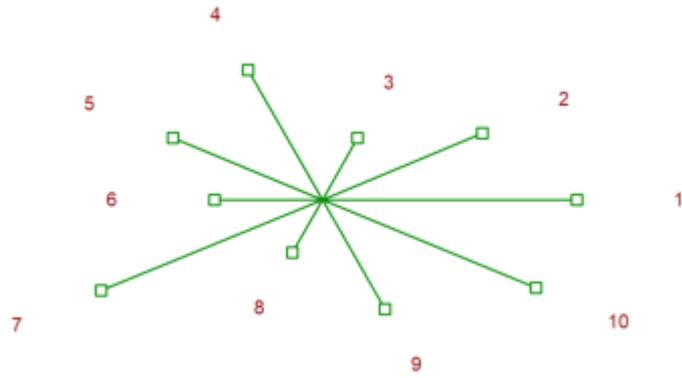
Kita juga bisa membuat diagram lingkaran. Karena hitam dan kuning membentuk koalisi, kita susun ulang unsur-unsurnya.

```
>i=[1,3,5,4,2]; piechart(BW[6,3:7][i],color=CP[i],lab=P[i]):
```



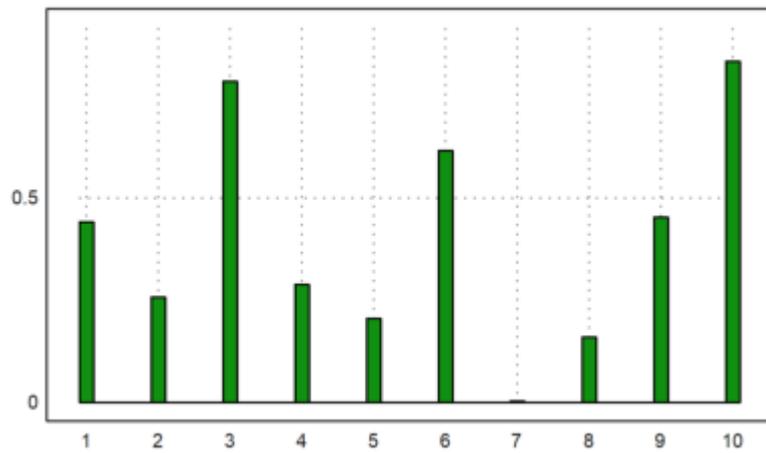
Berikut adalah jenis plot yang lain.

```
>starplot(normal(1,10)+4,lab=1:10,>rays):
```



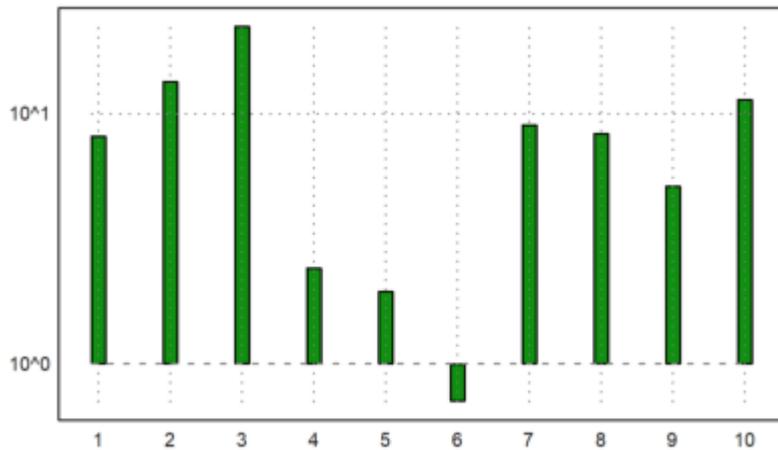
Beberapa plot dalam plot2d bagus untuk statika. Berikut adalah plot impuls data acak, yang didistribusikan secara seragam dalam [0,1].

```
>plot2d(makeimpulse(1:10, random(1,10)), >bar) :
```



Namun untuk data yang terdistribusi secara eksponensial, kita mungkin memerlukan plot logaritmik.

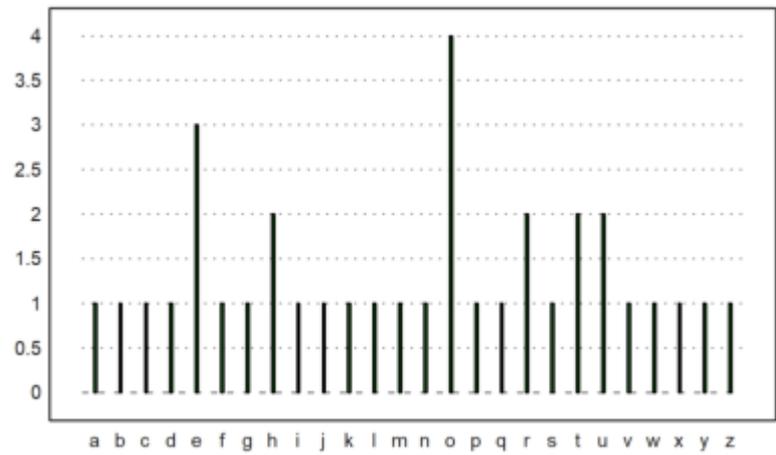
```
>logimpulseplot(1:10, -log(random(1,10))*10) :
```



Fungsi columnplot() lebih mudah digunakan, karena hanya memerlukan vektor nilai. Selain itu, fungsi ini dapat mengatur labelnya sesuai keinginan kita, kami telah menunjukkannya dalam tutorial ini.

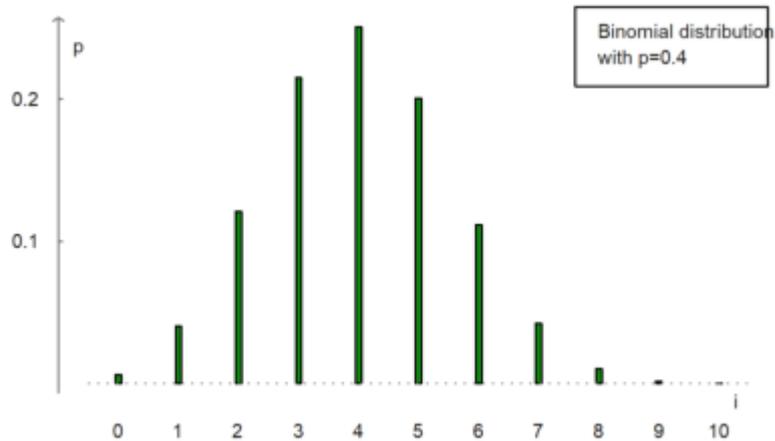
Berikut adalah aplikasi lain, tempat kita menghitung karakter dalam kalimat dan memplot statistik.

```
>v=strtochar("the quick brown fox jumps over the lazy dog"); ...
w=ascii("a"):ascii("z"); x=getmultiplicities(w,v); ...
cw=[]; for k=w; cw=cw|char(k); end; ...
columnspplot(x,lab=cw,width=0.05):
```



Anda juga dapat mengatur sumbu secara manual.

```
>n=10; p=0.4; i=0:n; x=bin(n,i)*p^i*(1-p)^(n-i); ...
columnspplot(x,lab=i,width=0.05,<frame,<grid); ...
yaxis(0,0:0.1:1,style="->",>left); xaxis(0,style="."); ...
label("p",0,0.25), label("i",11,0); ...
textbox(["Binomial distribution", "with p=0.4"]):
```



Berikut ini adalah cara untuk memetakan frekuensi angka dalam sebuah vektor.

Kita buat sebuah vektor bilangan acak integer 1 hingga 6.

```
>v:=inrandom(1,10,10)
```

```
[6, 10, 6, 4, 7, 7, 2, 2, 5, 3]
```

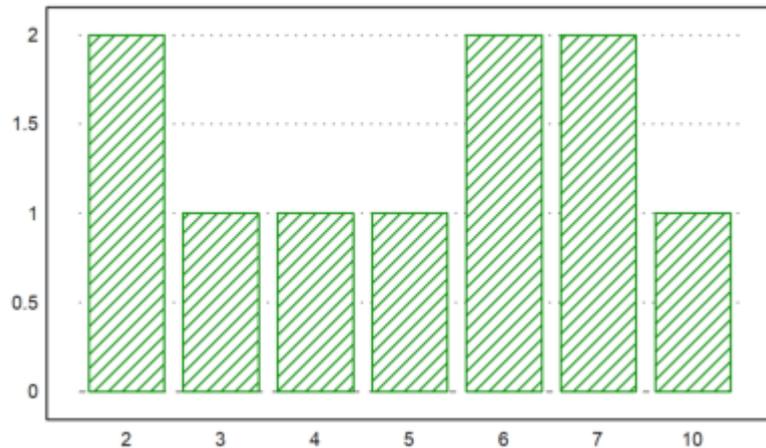
Lalu ekstrak angka-angka unik dalam v.

```
>vu:=unique(v)
```

```
[2, 3, 4, 5, 6, 7, 10]
```

Dan plot frekuensi pada kolom plot.

```
>columnsplot (getmultiplicities (vu,v) , lab=vu, style="/") :
```



Kami ingin menunjukkan fungsi untuk distribusi nilai empiris.

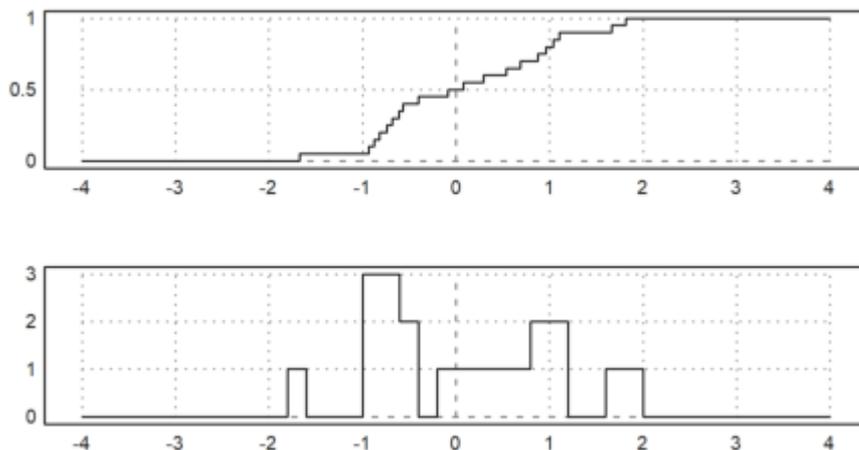
```
>x=normal (1,20) ;
```

Fungsi empdist(x,vs) memerlukan array nilai yang diurutkan. Jadi, kita harus mengurutkan x sebelum dapat menggunakannya.

```
>xs=sort (x) ;
```

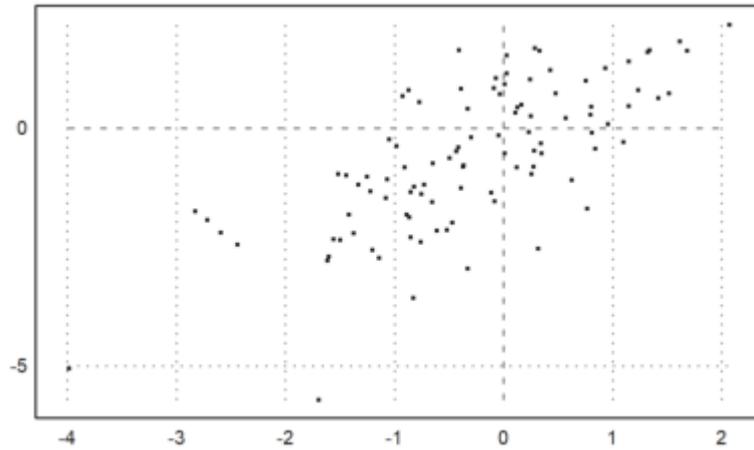
Kemudian kami memetakan distribusi empiris dan beberapa batang kepadatan ke dalam satu petak. Alih-alih menggunakan petak batang untuk distribusi, kali ini kami menggunakan petak gigi gergaji.

```
>figure (2,1) ; ...  
figure (1) ; plot2d ("empdist", -4,4;xs) ; ...  
figure (2) ; plot2d (histo (x,v=-4:0.2:4,<bar)) ; ...  
figure (0) :
```



Plot sebaran mudah dibuat di Euler dengan plot titik biasa. Grafik berikut menunjukkan bahwa X dan X+Y jelas berkorelasi positif.

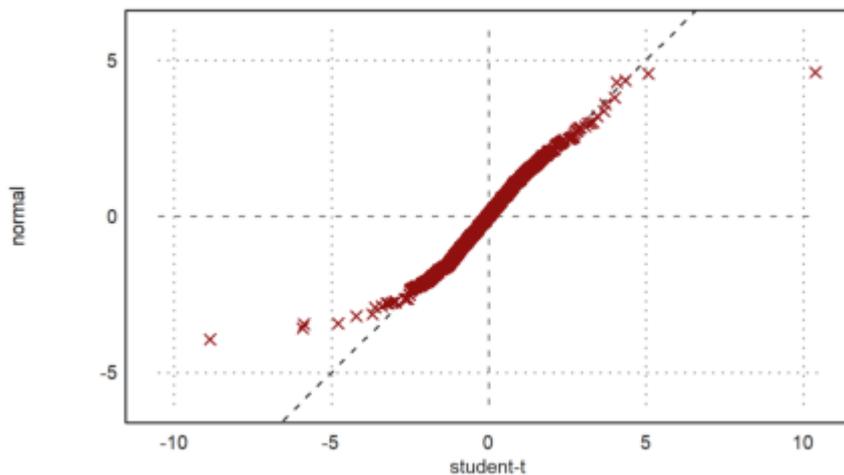
```
>x=normal (1,100) ; plot2d (x,x+rotright (x) ,>points, style="..") :
```



Sering kali, kita ingin membandingkan dua sampel dengan distribusi yang berbeda. Hal ini dapat dilakukan dengan plot kuantil-kuantil.

Untuk pengujian, kita mencoba distribusi t-student dan distribusi eksponensial.

```
>x=randt(1,1000,5); y=randnormal(1,1000,mean(x),dev(x)); ...
plot2d("x",r=6,style="--",yl="normal",xl="student-t",>vertical); ...
plot2d(sort(x),sort(y),>points,color=red,style="x",>add):
```



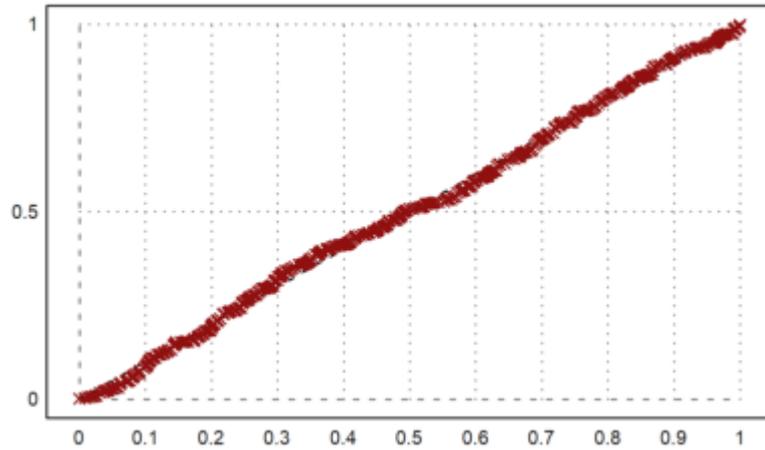
Plot tersebut dengan jelas menunjukkan bahwa nilai-nilai yang terdistribusi normal cenderung lebih kecil di ujung-ujung ekstrem.

Jika kita memiliki dua distribusi dengan ukuran yang berbeda, kita dapat memperluas yang lebih kecil atau mengecilkan yang lebih besar. Fungsi berikut ini bagus untuk keduanya. Fungsi ini mengambil nilai median dengan persentase antara 0 dan 1.

```
>function medianexpand (x,n) := median(x,p=linspace(0,1,n-1));
```

Mari kita bandingkan dua distribusi yang sama.

```
>x=random(1000); y=random(400); ...
plot2d("x",0,1,style="--"); ...
plot2d(sort(medianexpand(x,400)),sort(y),>points,color=red,style="x",>add):
```



Regresi dan Korelasi

Regresi linier dapat dilakukan dengan fungsi `polyfit()` atau berbagai fungsi fit.

Sebagai permulaan, kita mencari garis regresi untuk data univariat dengan `polyfit(x,y,1)`.

```
>x=1:10; y=[2,3,1,5,6,3,7,8,9,8]; writetable(x'|y',labc=["x","y"])
```

x	y
1	2
2	3
3	1
4	5
5	6
6	3
7	7
8	8
9	9
10	8

Kami ingin membandingkan kecocokan yang tidak tertimbang dan tertimbang. Pertama, koefisien kecocokan linier.

```
>p=polyfit(x,y,1)
```

```
[0.73333, 0.81212]
```

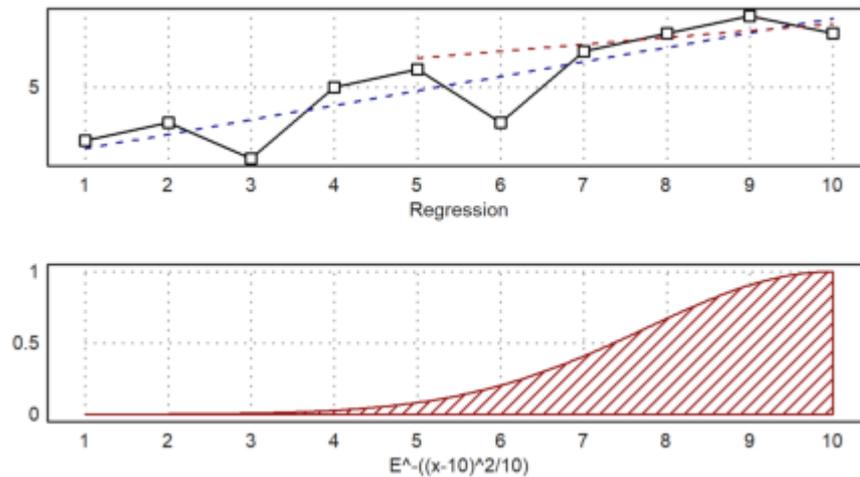
Sekarang koefisien dengan bobot yang menekankan nilai terakhir.

```
>w = "exp(-(x-10)^2/10)"; pw=polyfit(x,y,1,w=w(x))
```

```
[4.7157, 0.38319]
```

Kami memasukkan semuanya ke dalam satu plot untuk titik dan garis regresi, dan untuk bobot yang digunakan.

```
>figure(2,1); ...
figure(1); statplot(x,y,"b",xl="Regression"); ...
plot2d("evalpoly(x,p)",>add,color=blue,style="--"); ...
plot2d("evalpoly(x,pw)",5,10,>add,color=red,style="--"); ...
figure(2); plot2d(w,1,10,>filled,style="/",fillcolor=red,xl=w); ...
figure(0):
```



Untuk contoh lain, kami membaca survei siswa, usia mereka, usia orang tua mereka, dan jumlah saudara kandung dari sebuah berkas.

Tabel ini berisi "m" dan "f" di kolom kedua. Kami menggunakan variabel tok2 untuk mengatur terjemahan yang tepat alih-alih membiarkan readtable() mengumpulkan terjemahan.

```
>{MS,hd}:=readtable("table1.dat",tok2:["m","f"]); ...
writetable(MS,labc=hd,tok2:["m","f"]);
```

```
Could not open the file
table1.dat
for reading!
Try "trace errors" to inspect local variables after errors.
readtable:
  if filename!=none then open(filename,"r"); endif;
```

Bagaimana usia saling bergantung? Kesan pertama datang dari diagram sebaran berpasangan.

```
>scatterplots(tablecol(MS,3:5),hd[3:5]):

Variable or function MS not found.
Error in:
scatterplots(tablecol(MS,3:5),hd[3:5]): ...
^
```

Jelas bahwa usia ayah dan ibu saling bergantung. Mari kita tentukan dan gambarkan garis regresinya.

```
>cs:=MS[,4:5]'; ps:=polyfit(cs[1],cs[2],1)

MS is not a variable!
Error in:
cs:=MS[,4:5]'; ps:=polyfit(cs[1],cs[2],1) ...
^
```

Ini jelas model yang salah. Garis regresi adalah $s=17+0,74t$, di mana t adalah usia ibu dan s adalah usia ayah. Perbedaan usia mungkin sedikit bergantung pada usia, tetapi tidak terlalu banyak.

Sebaliknya, kami menduga fungsi seperti $s=a+t$. Maka a adalah rata-rata $s-t$. Itu adalah perbedaan usia rata-rata antara ayah dan ibu.

```
>da:=mean(cs[2]-cs[1])
```

```
cs is not a variable!  
Error in:  
da:=mean(cs[2]-cs[1]) ...  
      ^
```

Mari kita gambarkan ini menjadi satu diagram sebar.

```
>plot2d(cs[1],cs[2],>points); ...  
plot2d("evalpoly(x,ps)",color=red,style=".",>add); ...  
plot2d("x+da",color=blue,>add):
```

```
cs is not a variable!  
Error in:  
plot2d(cs[1],cs[2],>points); plot2d("evalpoly(x,ps)",color=re ...  
      ^
```

Berikut ini adalah diagram kotak dari dua zaman tersebut. Ini hanya menunjukkan bahwa zamannya berbeda.

```
>boxplot(cs, ["mothers", "fathers"]):
```

Menariknya bahwa perbedaan median tidak sebesar perbedaan mean.

```
>median(cs[2]) - median(cs[1])
```

1.5

Koefisien korelasi menunjukkan korelasi positif.

```
>correl(cs[1],cs[2])
```

0.7588307236

Korelasi peringkat adalah ukuran untuk urutan yang sama di kedua vektor. Korelasi ini juga cukup positif.

```
>rankcorrel(cs[1],cs[2])
```

0.758925292358

Membuat Fungsi baru

Tentu saja, bahasa EMT dapat digunakan untuk memprogram fungsi baru. Misalnya, kita mendefinisikan fungsi kemiringan.

$$sk(x) = \frac{\sqrt{n} \sum_i (x_i - m)^3}{(\sum_i (x_i - m)^2)^{3/2}}$$

di mana m adalah rata-rata x .

```
>function skew (x:vector) ...  
m=mean(x);  
return sqrt(cols(x))*sum((x-m)^3)/(sum((x-m)^2)^(3/2));  
endfunction
```

Seperti yang Anda lihat, kita dapat dengan mudah menggunakan bahasa matriks untuk mendapatkan implementasi yang sangat singkat dan efisien. Mari kita coba fungsi ini.

```
>data=normal(20); skew(normal(10))
```

```
-0.4733078755
```

Berikut adalah fungsi lainnya, yang disebut koefisien kemiringan Pearson.

```
>function skew1 (x) := 3*(mean(x)-median(x))/dev(x)
>skew1(data)
```

```
-0.236464036143
```

Simulasi Monte Carlo

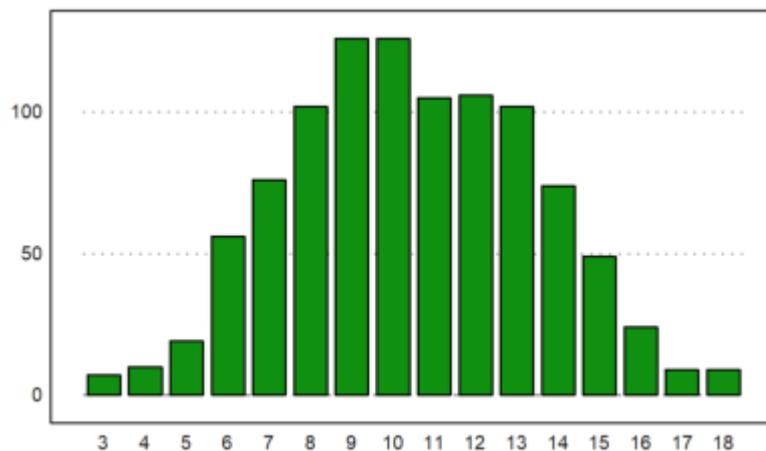
Euler dapat digunakan untuk mensimulasikan kejadian acak. Kita telah melihat contoh sederhana di atas. Berikut ini contoh lain, yang mensimulasikan 1000 kali lemparan 3 dadu, dan menanyakan distribusi jumlahnya.

```
>ds:=sum(intrandom(1000,3,6))'; fs=getmultiplicities(3:18,ds)
```

```
[7, 10, 19, 56, 76, 102, 126, 126, 105, 106, 102, 74, 49,
24, 9, 9]
```

Kita bisa merencanakannya sekarang.

```
>columnplot(fs,lab=3:18):
```



Menentukan distribusi yang diharapkan tidaklah mudah. ??Kami menggunakan rekursi tingkat lanjut untuk ini.

Fungsi berikut menghitung jumlah cara bilangan k dapat direpresentasikan sebagai jumlah n bilangan dalam rentang 1 hingga m. Fungsi ini bekerja secara rekursif dengan cara yang jelas.

```
>function map countways (k; n, m) ...
  if n==1 then return k>=1 && k<=m
  else
    sum=0;
    loop 1 to m; sum=sum+countways(k-#,n-1,m); end;
    return sum;
```

```
end;  
endfunction
```

Berikut ini hasil dari tiga kali lemparan dadu.

```
>countways (5:25,5,5)
```

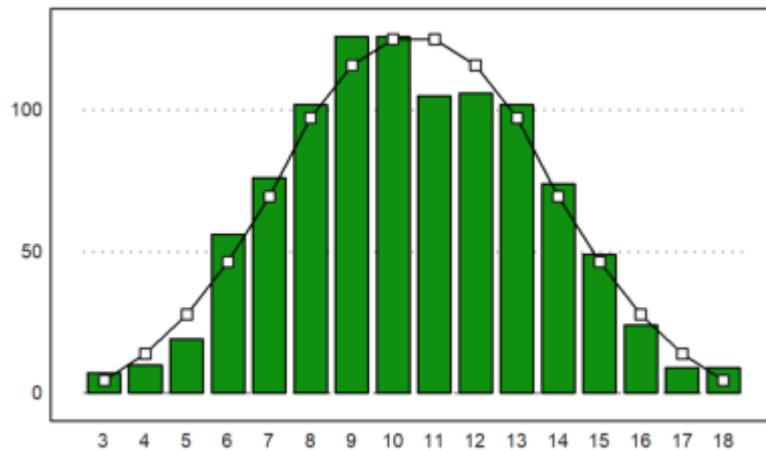
```
[1, 5, 15, 35, 70, 121, 185, 255, 320, 365, 381, 365, 320,  
255, 185, 121, 70, 35, 15, 5, 1]
```

```
>cw=countways (3:18,3,6)
```

```
[1, 3, 6, 10, 15, 21, 25, 27, 27, 25, 21, 15, 10, 6, 3,  
1]
```

Kami menambahkan nilai yang diharapkan ke plot.

```
>plot2d(cw/6^3*1000,>add); plot2d(cw/6^3*1000,>points,>add):
```



Untuk simulasi lain, deviasi nilai rata-rata dari n variabel acak berdistribusi normal 0-1 adalah $1/\sqrt{n}$.

```
>longformat; 1/sqrt(10)
```

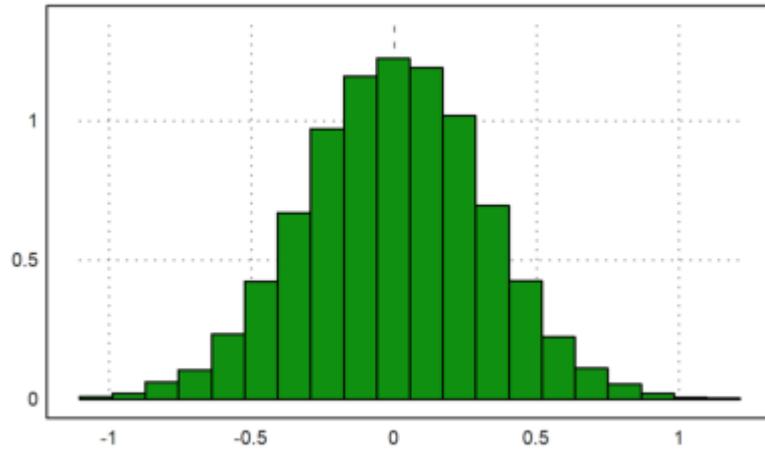
```
0.316227766017
```

Mari kita periksa ini dengan simulasi. Kita hasilkan 10000 kali 10 vektor acak.

```
>M=normal(10000,10); dev(mean(M)')
```

```
0.317960985884
```

```
>plot2d(mean(M) ',>distribution):
```



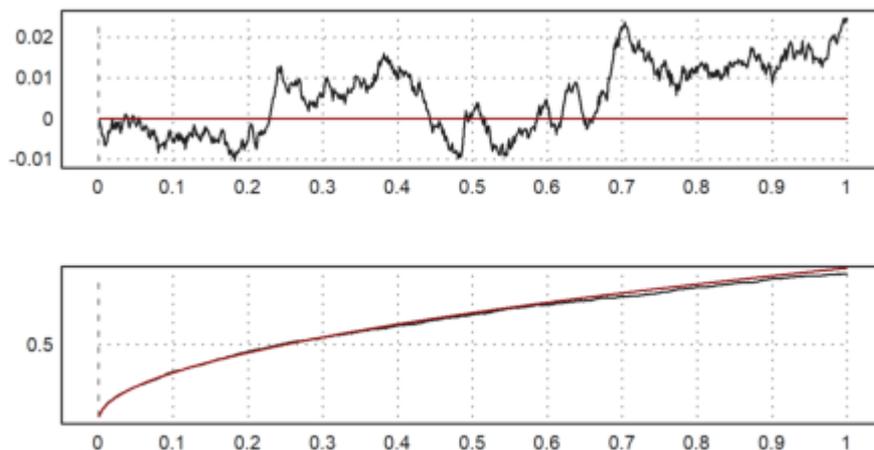
Median dari 10 bilangan acak berdistribusi normal 0-1 memiliki deviasi yang lebih besar.

```
>dev (median (M) ')
```

```
0.374409447037
```

Karena kita dapat dengan mudah menghasilkan lintasan acak, kita dapat mensimulasikan proses Wiener. Kita mengambil 1000 langkah dari 1000 proses. Kemudian kita memetakan deviasi standar dan rata-rata langkah ke-n dari proses ini bersama dengan nilai yang diharapkan dalam warna merah.

```
>n=1000; m=1000; M=cumsum(normal(n,m)/sqrt(m)); ...
t=(1:n)/n; figure(2,1); ...
figure(1); plot2d(t,mean(M)'); plot2d(t,0,color=red,>add); ...
figure(2); plot2d(t,dev(M)'); plot2d(t,sqrt(t),color=red,>add); ...
figure(0):
```



Pengujian

Pengujian merupakan alat penting dalam statistik. Dalam Euler, banyak pengujian yang diterapkan. Semua pengujian ini menghasilkan galat yang kita terima jika kita menolak hipotesis nol.

Sebagai contoh, kita menguji lemparan dadu untuk distribusi seragam. Pada 600 lemparan, kita memperoleh nilai berikut, yang kita masukkan ke dalam uji chi-kuadrat.

```
>chitest([90,103,114,101,103,89],dup(100,6)')
```

```
0.498830517952
```

Uji chi-square juga memiliki modus, yang menggunakan simulasi Monte Carlo untuk menguji statistik. Hasilnya harus hampir sama. Parameter >p menginterpretasikan vektor y sebagai vektor probabilitas.

```
>chitest ([90,103,114,101,103,89],dup(1/6,6)',>p,>montecarlo)
```

0.504

Kesalahan ini terlalu besar. Jadi kita tidak dapat menolak distribusi seragam. Ini tidak membuktikan bahwa dadu kita adil. Namun, kita tidak dapat menolak hipotesis kita.

Selanjutnya, kita menghasilkan 1000 lemparan dadu menggunakan generator angka acak, dan melakukan pengujian yang sama.

```
>n=1000; t=random([1,n*6]); chitest(count(t*6,6),dup(n,6)')
```

0.0820207573363

Mari kita uji nilai rata-rata 100 dengan uji-t.

```
>s=200+normal([1,100])*10; ...  
ttest(mean(s),dev(s),100,200)
```

0.0125894183808

Fungsi ttest() memerlukan nilai rata-rata, deviasi, jumlah data, dan nilai rata-rata yang akan diuji.

Sekarang mari kita periksa dua pengukuran untuk nilai rata-rata yang sama. Kita tolak hipotesis bahwa keduanya memiliki nilai rata-rata yang sama, jika hasilnya <0,05.

```
>tcomparedata(normal(1,10),normal(1,10))
```

0.479024465276

Jika kita menambahkan bias pada satu distribusi, kita akan mendapatkan lebih banyak penolakan. Ulangi simulasi ini beberapa kali untuk melihat efeknya.

```
>tcomparedata(normal(1,10),normal(1,10)+2)
```

0.000209130401913

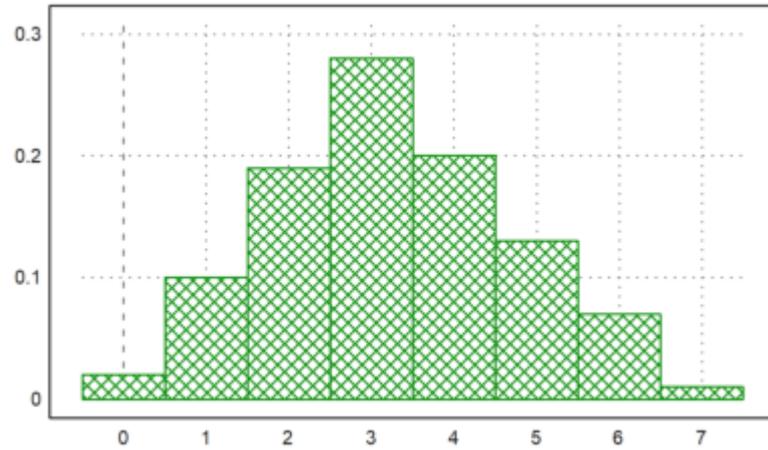
Pada contoh berikutnya, kita buat 20 lemparan dadu acak sebanyak 100 kali dan hitung angka-angka yang ada di dalamnya. Rata-rata harus ada $20/6=3,3$ angka.

```
>R=random(100,20); R=sum(R*6<=1)'; mean(R)
```

3.26

Sekarang kita bandingkan jumlah angka satu dengan distribusi binomial. Pertama kita gambarkan distribusi angka satu.

```
>plot2d(R,distribution=max(R)+1,even=1,style="\/"):
```



```
>t=count(R,21);
```

Lalu kami hitung nilai yang diharapkan.

```
>n=0:20; b=bin(20,n)*(1/6)^n*(5/6)^(20-n)*100;
```

Kita harus mengumpulkan beberapa angka untuk mendapatkan kategori yang cukup besar.

```
>t1=sum(t[1:2])|t[3:7]|sum(t[8:21]); ...
b1=sum(b[1:2])|b[3:7]|sum(b[8:21]);
```

Uji chi-kuadrat menolak hipotesis bahwa distribusi kami adalah distribusi binomial, jika hasilnya < 0,05.

```
>chitest(t1,b1)
```

```
0.822292761051
```

Contoh berikut berisi hasil dari dua kelompok orang (misalnya pria dan wanita) yang memilih satu dari enam partai.

```
>A=[23,37,43,52,64,74;27,39,41,49,63,76]; ...
writetable(A,wc=6,labr=["m","f"],labc=1:6)
```

	1	2	3	4	5	6
m	23	37	43	52	64	74
f	27	39	41	49	63	76

Kami ingin menguji independensi suara dari jenis kelamin. Uji tabel chi² melakukan hal ini. Hasilnya terlalu besar untuk menolak independensi. Jadi, kami tidak dapat mengatakan, apakah pemungutan suara bergantung pada jenis kelamin dari data ini.

```
>tabletest(A)
```

```
0.990701632326
```

Berikut ini adalah tabel yang diharapkan, jika kita mengasumsikan frekuensi pemungutan suara yang diamati.

```
>writetable(expectedtable(A),wc=6,dc=1,labr=["m","f"],labc=1:6)
```

	1	2	3	4	5	6
m	24.9	37.9	41.9	50.3	63.3	74.7
f	25.1	38.1	42.1	50.7	63.7	75.3

Kita dapat menghitung koefisien kontingensi yang dikoreksi. Karena sangat mendekati 0, kita simpulkan bahwa pemungutan suara tidak bergantung pada jenis kelamin.

```
>contingency(A)
```

```
0.0427225484717
```

Beberapa Pengujian Lainnya

Selanjutnya, kami menggunakan analisis varians (uji F) untuk menguji tiga sampel data berdistribusi normal untuk nilai rata-rata yang sama. Metode ini disebut ANOVA (analisis varians). Dalam Euler, fungsi `varanalysis()` digunakan.

```
>x1=[109,111,98,119,91,118,109,99,115,109,94]; mean(x1),
```

```
106.545454545
```

```
>x2=[120,124,115,139,114,110,113,120,117]; mean(x2),
```

```
119.111111111
```

```
>x3=[120,112,115,110,105,134,105,130,121,111]; mean(x3)
```

```
116.3
```

```
>varanalysis(x1,x2,x3)
```

```
0.0138048221371
```

Artinya, kita menolak hipotesis nilai rata-rata yang sama. Kita melakukan ini dengan probabilitas kesalahan sebesar 1,3%.

Ada juga uji median, yang menolak sampel data dengan distribusi rata-rata yang berbeda dengan menguji median sampel gabungan.

```
>a=[56,66,68,49,61,53,45,58,54];
>b=[72,81,51,73,69,78,59,67,65,71,68,71];
>mediantest(a,b)
```

```
0.0241724220052
```

Uji kesetaraan lainnya adalah uji peringkat. Uji peringkat jauh lebih tajam daripada uji median.

```
>ranktest(a,b)
```

```
0.00199969612469
```

Dalam contoh berikut, kedua distribusi memiliki rata-rata yang sama.

```
>ranktest (random (1, 100) , random (1, 50) *3-1)
```

```
0.161402579346
```

Sekarang, mari kita coba simulasikan dua perawatan a dan b yang diterapkan pada orang yang berbeda.

```
>a=[8.0,7.4,5.9,9.4,8.6,8.2,7.6,8.1,6.2,8.9];  
>b=[6.8,7.1,6.8,8.3,7.9,7.2,7.4,6.8,6.8,8.1];
```

Uji signum memutuskan, apakah a lebih baik dari b.

```
>signtest (a,b)
```

```
0.0546875
```

Ini adalah kesalahan yang sangat besar. Kita tidak dapat menolak bahwa a sama baiknya dengan b.

Uji Wilcoxon lebih tajam daripada uji ini, tetapi bergantung pada nilai kuantitatif perbedaannya.

```
>wilcoxon (a,b)
```

```
0.0296680599405
```

Mari kita coba dua pengujian lagi menggunakan seri yang dihasilkan.

```
>wilcoxon (normal (1, 20) , normal (1, 20) -1)
```

```
0.0726998875325
```

```
>wilcoxon (normal (1, 20) , normal (1, 20) )
```

```
0.396919522771
```

Angka Acak

Berikut ini adalah pengujian untuk generator angka acak. Euler menggunakan generator yang sangat bagus, jadi kita tidak perlu mengharapkan masalah apa pun.

Pertama-tama kita menghasilkan sepuluh juta angka acak dalam [0,1].

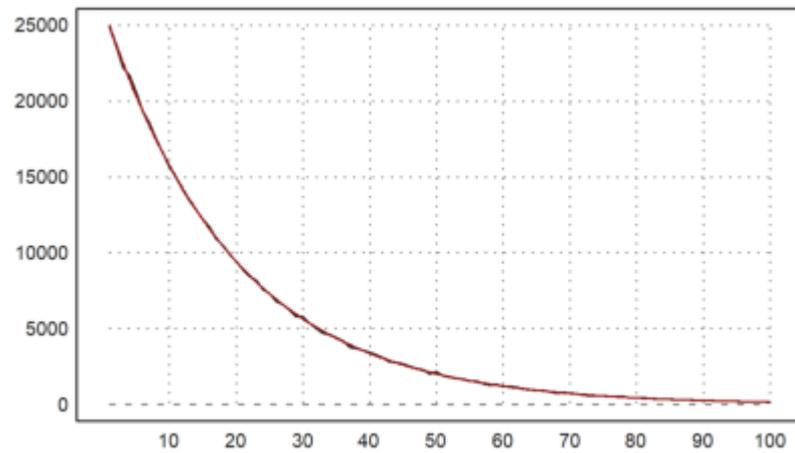
```
>n:=10000000; r:=random (1, n);
```

Berikutnya kita hitung jarak antara dua angka kurang dari 0,05.

```
>a:=0.05; d:=differences (nonzeros (r<a) );
```

Terakhir, kami memplot berapa kali setiap jarak terjadi, dan membandingkannya dengan nilai yang diharapkan.

```
>m=getmultiplicities (1:100,d); plot2d(m); ...  
plot2d("n*(1-a)^(x-1)*a^2",color=red,>add):
```



Hapus data.

```
>remvalue n;
```

Pendahuluan bagi Pengguna Proyek R

Jelas, EMT tidak bersaing dengan R sebagai paket statistik. Akan tetapi, ada banyak prosedur dan fungsi statistik yang tersedia di EMT juga. Jadi, EMT dapat memenuhi kebutuhan dasar. Lagi pula, EMT dilengkapi dengan paket numerik dan sistem aljabar komputer.

Buku catatan ini ditujukan bagi Anda yang sudah familier dengan R, tetapi perlu mengetahui perbedaan sintaksis EMT dan R. Kami mencoba memberikan gambaran umum tentang hal-hal yang jelas dan kurang jelas yang perlu Anda ketahui.

Selain itu, kami melihat cara untuk bertukar data antara kedua sistem.

Harap dicatat bahwa ini adalah pekerjaan yang masih dalam tahap pengerjaan.

Sintaksis Dasar

Hal pertama yang Anda pelajari di R adalah membuat vektor. Dalam EMT, perbedaan utamanya adalah operator `:` dapat mengambil ukuran langkah. Selain itu, operator ini memiliki daya pengikatan yang rendah.

```
>n=10; 0:n/20:n-1
```

```
[0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5,
7, 7.5, 8, 8.5, 9]
```

Fungsi `c()` tidak ada. Dimungkinkan untuk menggunakan vektor guna menggabungkan berbagai hal.

Contoh berikut ini, seperti banyak contoh lainnya, berasal dari "Interoduction to R" yang disertakan dalam proyek R. Jika Anda membaca PDF ini, Anda akan menemukan bahwa saya mengikuti alurnya dalam tutorial ini.

```
>x=[10.4, 5.6, 3.1, 6.4, 21.7]; [x,0,x]
```

```
[10.4, 5.6, 3.1, 6.4, 21.7, 0, 10.4, 5.6, 3.1, 6.4, 21.7]
```

Operator titik dua dengan ukuran langkah EMT digantikan oleh fungsi `seq()` di R. Kita dapat menulis fungsi ini dalam EMT.

```
>function seq(a,b,c) := a:b:c; ...
seq(0,-0.1,-1)
```

```
[0, -0.1, -0.2, -0.3, -0.4, -0.5, -0.6, -0.7, -0.8, -0.9, -1]
```

Fungsi `rep()` dari R tidak ada dalam EMT. Untuk input vektor, dapat ditulis sebagai berikut.

```
>function rep(x:vector,n:index) := flatten(dup(x,n)); ...  
rep(x,2)
```

```
[10.4, 5.6, 3.1, 6.4, 21.7, 10.4, 5.6, 3.1, 6.4, 21.7]
```

Perhatikan bahwa "=" atau "==" digunakan untuk penugasan. Operator "->" digunakan untuk unit dalam EMT.

```
>125km -> " miles"
```

```
77.6713990297 miles
```

Operator "<-" untuk penugasan menyesatkan dan bukan ide yang baik untuk R. Berikut ini akan membandingkan a dan -4 dalam EMT.

```
>a=2; a<-4
```

```
0
```

Dalam R, "`a<-4<3`" berfungsi, tetapi "`a<-4<-3`" tidak. Saya juga mengalami ambiguitas serupa dalam EMT, tetapi mencoba menghilangkannya sedikit demi sedikit.

EMT dan R memiliki vektor bertipe boolean. Namun dalam EMT, angka 0 dan 1 digunakan untuk mewakili false dan true. Dalam R, nilai true dan false tetap dapat digunakan dalam aritmatika biasa seperti dalam EMT.

```
>x<5, %*x
```

```
[0, 0, 1, 0, 0]  
[0, 0, 3.1, 0, 0]
```

EMT memunculkan kesalahan atau menghasilkan NAN, tergantung pada tanda "kesalahan".

```
>errors off; 0/0, isNaN(sqrt(-1)), errors on;
```

```
NAN  
1
```

String sama di R dan EMT. Keduanya berada di lokal saat ini, bukan di Unicode.

Di R ada paket untuk Unicode. Di EMT, string dapat berupa string Unicode. String unicode dapat diterjemahkan ke pengodean lokal dan sebaliknya. Selain itu, u"..." dapat berisi entitas HTML.

```
>u"René Grothmann"
```

```
© René Grothmann
```

Berikut ini mungkin atau mungkin tidak ditampilkan dengan benar pada sistem Anda sebagai A dengan titik dan garis di atasnya. Hal ini bergantung pada font yang Anda gunakan.

```
>chartoutf([480])
```

Ä

Penggabungan string dilakukan dengan "+" atau "|". String dapat menyertakan angka, yang akan dicetak dalam format saat ini.

```
>"pi = "+pi
```

```
pi = 3.14159265359
```

Pengindeksan

Sering kali, ini akan berfungsi seperti di R.

Namun EMT akan menginterpretasikan indeks negatif dari belakang vektor, sementara R menginterpretasikan $x[n]$ sebagai x tanpa elemen ke- n .

```
>x, x[1:3], x[-2]
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3]
9
```

Perilaku R dapat dicapai dalam EMT dengan `drop()`.

```
>drop(x,2)
```

```
[1, 3, 4, 5, 6, 7, 8, 9, 10]
```

Vektor logika tidak diperlakukan secara berbeda sebagai indeks dalam EMT, berbeda dengan R. Anda perlu mengekstrak elemen bukan nol terlebih dahulu dalam EMT.

```
>x, x>5, x[nonzeros(x>5)]
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[0, 0, 0, 0, 0, 1, 1, 1, 1, 1]
[6, 7, 8, 9, 10]
```

Sama seperti di R, vektor indeks dapat berisi pengulangan.

```
>x[[1,2,2,1]]
```

```
[1, 2, 2, 1]
```

Namun, nama untuk indeks tidak dimungkinkan dalam EMT. Untuk paket statistik, hal ini mungkin sering diperlukan untuk memudahkan akses ke elemen vektor.

Untuk meniru perilaku ini, kita dapat mendefinisikan fungsi sebagai berikut.

```
>function sel (v,i,s) := v[indexof(s,i)]; ...
s=["first","second","third","fourth"]; sel(x,["first","third"],s)
```

```

Trying to overwrite protected function sel!
Error in:
function sel (v,i,s) := v[indexof(s,i)]; ... ...
      ^

Trying to overwrite protected function sel!
Error in:
function sel (v,i,s) := v[indexof(s,i)]; ... ...
      ^

Trying to overwrite protected function sel!
Error in:
function sel (v,i,s) := v[indexof(s,i)]; ... ...
      ^
[10.4, 3.1]

```

Tipe Data

EMT memiliki lebih banyak tipe data tetap daripada R. Jelas, di R terdapat vektor yang terus bertambah. Anda dapat menetapkan vektor numerik kosong v dan menetapkan nilai ke elemen $v[17]$. Hal ini tidak mungkin dilakukan di EMT.

Berikut ini agak tidak efisien.

```
>v=[]; for i=1 to 10000; v=v|i; end;
```

EMT sekarang akan membuat vektor dengan v dan i yang ditambahkan pada tumpukan dan menyalin vektor itu kembali ke variabel global v .

Yang lebih efisien mendefinisikan vektor terlebih dahulu.

```
>v=zeros(10000); for i=1 to 10000; v[i]=i; end;
```

Untuk mengubah jenis tanggal di EMT, Anda dapat menggunakan fungsi seperti `complex()`.

```
>complex(1:4)

[ 1+0i , 2+0i , 3+0i , 4+0i ]
```

Konversi ke string hanya dimungkinkan untuk tipe data dasar. Format saat ini digunakan untuk penggabungan string sederhana. Namun, ada fungsi seperti `print()` atau `frac()`.

Untuk vektor, Anda dapat dengan mudah menulis fungsi Anda sendiri.

```
>function tostr (v) ...
s="[";
loop 1 to length(v);
  s=s+print(v[#],2,0);
  if #<length(v) then s=s+","; endif;
end;
return s+"]";
endfunction
```

```
>tostr(linspace(0,1,10))

[0.00,0.10,0.20,0.30,0.40,0.50,0.60,0.70,0.80,0.90,1.00]
```

Untuk komunikasi dengan Maxima, terdapat fungsi `convertmxm()`, yang juga dapat digunakan untuk memformat vektor untuk keluaran.

```
>convertmxm(1:10)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Untuk Latex perintah `tex` dapat digunakan untuk mendapatkan perintah Latex.

```
>tex(&[1,2,3])
```

```
\left[ 1 , 2 , 3 \right]
```

Faktor dan Tabel

Dalam pengantar R terdapat contoh dengan apa yang disebut faktor.

Berikut ini adalah daftar wilayah dari 30 negara bagian.

```
>austates = ["tas", "sa", "qld", "nsw", "nsw", "nt", "wa", "wa", ...
"qld", "vic", "nsw", "vic", "qld", "qld", "sa", "tas", ...
"sa", "nt", "wa", "vic", "qld", "nsw", "nsw", "wa", ...
"sa", "act", "nsw", "vic", "vic", "act"];
```

Asumsikan, kita memiliki pendapatan yang sesuai di setiap negara bagian.

```
>incomes = [60, 49, 40, 61, 64, 60, 59, 54, 62, 69, 70, 42, 56, ...
61, 61, 61, 58, 51, 48, 65, 49, 49, 41, 48, 52, 46, ...
59, 46, 58, 43];
```

Sekarang, kita ingin menghitung rata-rata pendapatan di wilayah tersebut. Sebagai program statistik, R memiliki `factor()` dan `tapply()` untuk ini.

EMT dapat melakukan ini dengan menemukan indeks wilayah dalam daftar wilayah yang unik.

```
>auterr=sort(unique(austates)); f=indexofsorted(auterr,austates)
```

```
[6, 5, 4, 2, 2, 3, 8, 8, 4, 7, 2, 7, 4, 4, 5, 6, 5, 3,
8, 7, 4, 2, 2, 8, 5, 1, 2, 7, 7, 1]
```

Pada titik tersebut, kita dapat menulis fungsi loop kita sendiri untuk melakukan sesuatu hanya untuk satu faktor.

Atau kita dapat meniru fungsi `tapply()` dengan cara berikut.

```
>function map_tappl (i; f$:call, cat, x) ...
u=sort(unique(cat));
f=indexof(u,cat);
return f$(x[nonzeros(f==indexof(u,i))]);
endfunction
```

Agak tidak efisien, karena menghitung wilayah unik untuk setiap `i`, tetapi berhasil.

```
>tappl(auterr,"mean",austates,incomes)
```

```
[44.5, 57.3333333333, 55.5, 53.6, 55, 60.5, 56, 52.25]
```

Perhatikan bahwa ini berfungsi untuk setiap vektor wilayah.

```
>tappl(["act", "nsw"], "mean", austates, incomes)
```

```
[44.5, 57.3333333333]
```

Sekarang, paket statistik EMT mendefinisikan tabel seperti di R. Fungsi `readtable()` dan `writetable()` dapat digunakan untuk input dan output.

Jadi kita dapat mencetak pendapatan negara rata-rata di wilayah dengan cara yang mudah.

```
>writetable(tappl(auterr, "mean", austates, incomes), labc=auterr, wc=7)
```

```
      act      nsw      nt      qld      sa      tas      vic      wa
44.5  57.33  55.5  53.6  55  60.5  56  52.25
```

Kita juga dapat mencoba meniru perilaku R sepenuhnya.

Faktor-faktor tersebut harus disimpan dalam suatu koleksi dengan jenis dan kategori (negara bagian dan teritori dalam contoh kita). Untuk EMT, kita tambahkan indeks yang telah dihitung sebelumnya.

```
>function makef (t) ...
## Factor data
## Returns a collection with data t, unique data, indices.
## See: tapply
u=sort(unique(t));
return {{t,u,indexofsorted(u,t)}};
endfunction
```

```
>statef=makef(austates);
```

Sekarang elemen ketiga dari koleksi akan berisi indeks.

```
>statef[3]
```

```
[6, 5, 4, 2, 2, 3, 8, 8, 4, 7, 2, 7, 4, 4, 5, 6, 5, 3,
8, 7, 4, 2, 2, 8, 5, 1, 2, 7, 7, 1]
```

Sekarang kita dapat meniru `tapply()` dengan cara berikut. Fungsi ini akan mengembalikan tabel sebagai kumpulan data tabel dan judul kolom.

```
>function tapply (t:vector,tf,f$:call) ...
## Makes a table of data and factors
## tf : output of makef()
## See: makef
uf=tf[2]; f=tf[3]; x=zeros(length(uf));
for i=1 to length(uf);
  ind=nonzeros(f==i);
  if length(ind)==0 then x[i]=NAN;
  else x[i]=f$(t[ind]);
endif;
end;
return {{x,uf}};
endfunction
```

Kami tidak menambahkan banyak pemeriksaan tipe di sini. Satu-satunya tindakan pencegahan menyangkut kategori (faktor) tanpa data. Namun, seseorang harus memeriksa panjang `t` yang benar dan kebenaran koleksi `tf`.

Tabel ini dapat dicetak sebagai tabel dengan `writetable()`.

```
>writetable(tapply(incomes, statef, "mean"), wc=7)
```

act	nsw	nt	qld	sa	tas	vic	wa
44.5	57.33	55.5	53.6	55	60.5	56	52.25

Array

EMT hanya memiliki dua dimensi untuk array. Tipe data ini disebut matriks. Akan mudah untuk menulis fungsi untuk dimensi yang lebih tinggi atau pustaka C untuk ini.

R memiliki lebih dari dua dimensi. Dalam R, array adalah vektor dengan bidang dimensi.

Dalam EMT, vektor adalah matriks dengan satu baris. Vektor dapat dibuat menjadi matriks dengan `redim()`.

```
>shortformat; X=redim(1:20,4,5)
```

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

Ekstraksi baris dan kolom, atau sub-matriks, sangat mirip di R.

```
>X[,2:3]
```

2	3
7	8
12	13
17	18

Namun, dalam R dimungkinkan untuk menetapkan daftar indeks vektor tertentu ke suatu nilai. Hal yang sama dimungkinkan dalam EMT hanya dengan loop.

```
>function setmatrixvalue (M, i, j, v) ...  
  loop 1 to max(length(i),length(j),length(v))  
    M[i{#},j{#}] = v{#};  
end;  
endfunction
```

Kami mendemonstrasikan ini untuk menunjukkan bahwa matriks dilewatkan dengan referensi dalam EMT. Jika Anda tidak ingin mengubah matriks asli M, Anda perlu menyalinnya dalam fungsi tersebut.

```
>setmatrixvalue(X,1:3,3:-1:1,0); X,
```

1	2	0	4	5
6	0	8	9	10
0	12	13	14	15
16	17	18	19	20

Produk luar dalam EMT hanya dapat dilakukan antara vektor. Hal ini dilakukan secara otomatis karena bahasa matriks. Satu vektor harus berupa vektor kolom dan yang lainnya berupa vektor baris.

```
>(1:5)*(1:5)'
```

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15

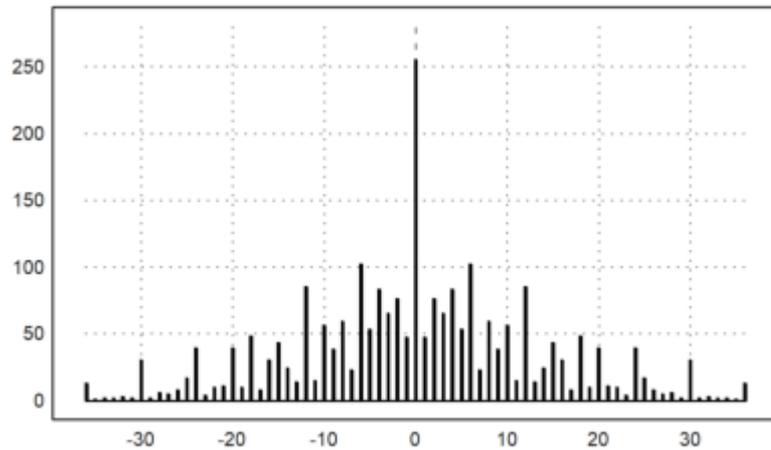
4	8	12	16	20
5	10	15	20	25

Dalam pengantar PDF untuk R terdapat sebuah contoh, yang menghitung distribusi ab-cd untuk a,b,c,d yang dipilih secara acak dari 0 hingga n. Solusi dalam R adalah membentuk matriks 4 dimensi dan menjalankan table() di atasnya.

Tentu saja, ini dapat dicapai dengan loop. Namun, loop tidak efektif dalam EMT atau R. Dalam EMT, kita dapat menulis loop dalam C dan itu akan menjadi solusi tercepat.

Namun, kita ingin meniru perilaku R. Untuk ini, kita perlu meratakan perkalian ab dan membuat matriks ab-cd.

```
>a=0:6; b=a'; p=flatten(a*b); q=flatten(p-p'); ...
u=sort(unique(q)); f=getmultiplicities(u,q); ...
statplot(u,f,"h"):
```



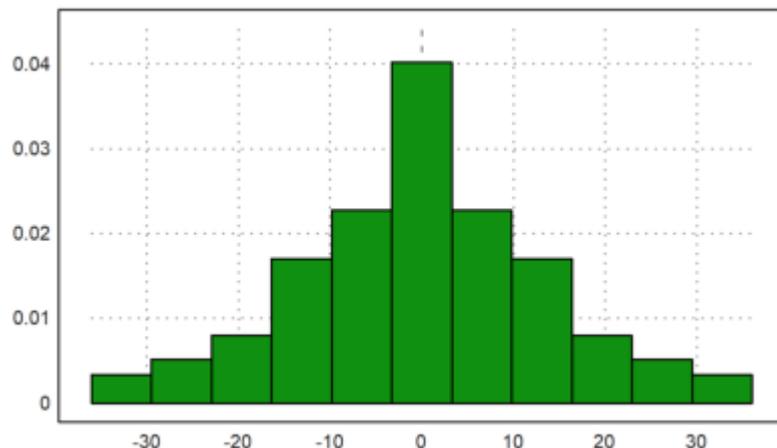
Selain multiplisitas yang tepat, EMT dapat menghitung frekuensi dalam vektor.

```
>getfrequencies(q,-50:10:50)
```

```
[0, 23, 132, 316, 602, 801, 333, 141, 53, 0]
```

Cara termudah untuk memplot ini sebagai distribusi adalah sebagai berikut.

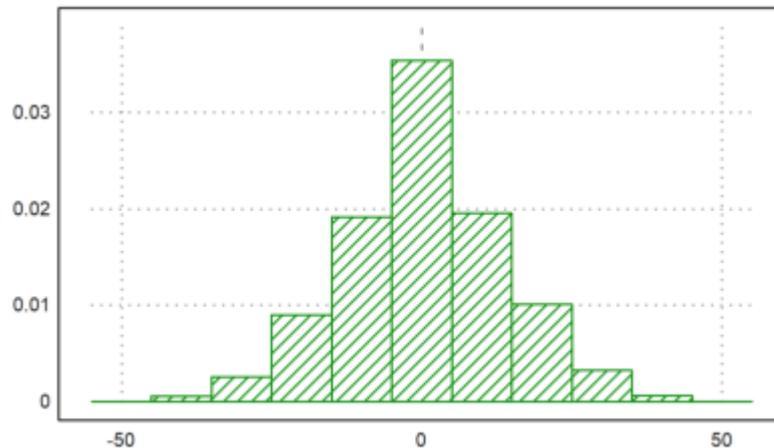
```
>plot2d(q,distribution=11):
```



Namun, Anda juga dapat menghitung terlebih dahulu jumlah dalam interval yang dipilih. Tentu saja, berikut ini menggunakan `getfrequencies()` secara internal.

Karena fungsi `histo()` mengembalikan frekuensi, kita perlu menskalakannya sehingga integral di bawah grafik batang adalah 1.

```
>{x,y}=histo(q,v=-55:10:55); y=y/sum(y)/differences(x); ...  
plot2d(x,y,>bar,style="/"):
```



Daftar

EMT memiliki dua jenis daftar. Satu adalah daftar global yang dapat diubah, dan yang lainnya adalah jenis daftar yang tidak dapat diubah. Kami tidak peduli dengan daftar global di sini.

Jenis daftar yang tidak dapat diubah disebut koleksi dalam EMT. Ia berperilaku seperti struktur dalam C, tetapi elemennya hanya diberi nomor dan tidak diberi nama.

```
>L={{ "Fred", "Flintstone", 40, [1990, 1992] }}
```

```
Fred  
Flintstone  
40  
[1990, 1992]
```

Saat ini unsur-unsur tersebut tidak memiliki nama, meskipun nama dapat ditetapkan untuk tujuan khusus. Unsur-unsur tersebut diakses dengan angka.

```
>(L[4])[2]
```

```
1992
```

Input dan Output File (Membaca dan Menulis Data)

Anda sering kali ingin mengimpor matriks data dari sumber lain ke EMT. Tutorial ini memberi tahu Anda tentang berbagai cara untuk mencapainya. Fungsi sederhana adalah `writematrix()` dan `readmatrix()`.

Mari kita tunjukkan cara membaca dan menulis vektor bilangan real ke dalam file.

```
>a=random(1,100); mean(a), dev(a),
```

```
0.48034  
0.29635
```

Untuk menulis data ke dalam berkas, kami menggunakan fungsi `writematrix()`.

Karena pengantar ini kemungkinan besar berada di dalam direktori, tempat pengguna tidak memiliki akses tulis, kami menulis data ke direktori beranda pengguna. Untuk buku catatan sendiri, ini tidak diperlukan, karena berkas data akan ditulis ke dalam direktori yang sama.

```
>filename="test.dat";
```

Sekarang kita tulis vektor kolom `a'` ke dalam berkas. Ini menghasilkan satu angka di setiap baris berkas.

```
>writematrix(a',filename);
```

Untuk membaca data, kita menggunakan `readmatrix()`.

```
>a=readmatrix(filename)';
```

Dan hapus berkasnya.

```
>fileremove(filename);  
>mean(a), dev(a),
```

```
0.48034  
0.29635
```

Fungsi `writematrix()` atau `writetable()` dapat dikonfigurasi untuk bahasa lain.

Misalnya, jika Anda memiliki sistem bahasa Indonesia (titik desimal dengan koma), Excel Anda memerlukan nilai dengan koma desimal yang dipisahkan oleh titik koma dalam file csv (nilai default dipisahkan dengan koma). File berikut "test.csv" akan muncul di folder Anda saat ini.

```
>filename="test.csv"; ...  
writematrix(random(5,3),file=filename,separator=",");
```

Anda sekarang dapat membuka berkas ini langsung dengan Excel Indonesia.

```
>fileremove(filename);
```

Terkadang kita memiliki string dengan token seperti berikut.

```
>s1="f m m f m m m f f f m m f"; ...  
s2="f f f m m f f";
```

Untuk mengeniknisi ini, kami mendefinisikan vektor token.

```
>tok=["f","m"]
```

```
f  
m
```

Lalu kita dapat menghitung berapa kali setiap token muncul dalam string, dan memasukkan hasilnya ke dalam tabel.

```
>M:=getmultiplicities(tok,strtokens(s1))_ ...  
getmultiplicities(tok,strtokens(s2));
```

Tulis tabel dengan tajuk token.

```
>writetable(M, labc=tok, labr=1:2, wc=8)
```

```
      f      m
1     6     7
2     5     2
```

Untuk statika, EMT dapat membaca dan menulis tabel.

```
>file="test.dat"; open(file,"w"); ...
writeln("A,B,C"); writematrix(random(3,3)); ...
close();
```

Berkasnya tampak seperti ini.

```
>printfile(file)
```

```
A,B,C
0.5633325767159784,0.5995128294260197,0.1841418587845089
0.7639732295357271,0.3415267565426297,0.1048586908322706
0.4358711000760289,0.4213578444046047,0.4133000262689321
```

Fungsi `readtable()` dalam bentuk yang paling sederhana dapat membacanya dan mengembalikan kumpulan nilai dan baris judul.

```
>L=readtable(file,>list);
```

Koleksi ini dapat dicetak dengan `writetable()` ke buku catatan, atau ke berkas.

```
>writetable(L,wc=10,dc=5)
```

```
      A      B      C
0.56333 0.59951 0.18414
0.76397 0.34153 0.10486
0.43587 0.42136 0.4133
```

Matriks nilai adalah elemen pertama L. Perhatikan bahwa `mean()` dalam EMT menghitung nilai rata-rata baris matriks.

```
>mean(L[1])
```

```
0.449
0.40345
0.42351
```

Berkas CSV

Pertama, mari kita tulis matriks ke dalam berkas. Untuk output, kita buat berkas di direktori kerja saat ini.

```
>file="test.csv"; ...
M=random(3,3); writematrix(M,file);
```

Berikut ini isi berkas tersebut.

```
>printfile(file)
```

```
0.7013483758152518,0.4175390095872878,0.1734211634568865
0.5767819667358264,0.1382128823306854,0.9354348571561104
0.5578454320397516,0.8742005728683661,0.07049758219732646
```

CVS ini dapat dibuka pada sistem bahasa Inggris ke Excel dengan mengklik dua kali. Jika Anda mendapatkan berkas tersebut pada sistem bahasa Jerman, Anda perlu mengimpor data ke Excel dengan memperhatikan titik desimal.

Namun, titik desimal juga merupakan format default untuk EMT. Anda dapat membaca matriks dari berkas dengan `readmatrix()`.

```
>readmatrix(file)
```

```
0.70135 0.41754 0.17342
0.57678 0.13821 0.93543
0.55785 0.8742 0.070498
```

bermaksud untuk menulis beberapa matriks ke dalam satu berkas. Perintah `open()` dapat membuka file untuk ditulis dengan parameter "w". Nilai default untuk membaca adalah "r".

```
>open(file,"w"); writematrix(M); writematrix(M'); close();
```

Matriks dipisahkan oleh baris kosong. Untuk membaca matriks, buka berkas dan panggil `readmatrix()` beberapa kali.

```
>open(file); A=readmatrix(); B=readmatrix(); A==B, close();
```

```
1 0 0
0 1 0
0 0 1
```

Di Excel atau lembar kerja serupa, Anda dapat mengekspor matriks sebagai CSV (nilai yang dipisahkan koma). Di Excel 2007, gunakan "simpan sebagai" dan "format lain", lalu pilih "CSV". Pastikan, tabel saat ini hanya berisi data yang ingin Anda ekspor.

Berikut ini contohnya.

```
>printfile("excel-data.csv")
```

```
Could not open the file
excel-data.csv
for reading!
Try "trace errors" to inspect local variables after errors.
printfile:
  open(filename,"r");
```

Seperti yang Anda lihat, sistem Jerman saya menggunakan titik koma sebagai pemisah dan koma desimal. Anda dapat mengubahnya di pengaturan sistem atau di Excel, tetapi tidak diperlukan untuk membaca matriks ke EMT.

Cara termudah untuk membaca ini ke Euler adalah `readmatrix()`. Semua koma diganti dengan titik dengan parameter `>comma`. Untuk CSV bahasa Inggris, cukup abaikan parameter ini.

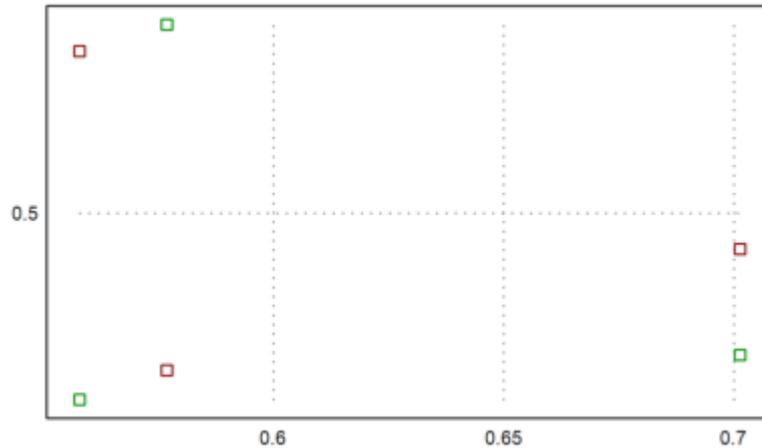
```
>M=readmatrix("excel-data.csv",>comma)
```

```
Could not open the file
excel-data.csv
for reading!
```

```
Try "trace errors" to inspect local variables after errors.
readmatrix:
  if filename<>" then open(filename,"r"); endif;
```

Mari kita plot ini.

```
>plot2d(M'[1],M'[2:3],>points,color=[red,green]'):
```



Ada cara yang lebih mendasar untuk membaca data dari sebuah berkas. Anda dapat membuka berkas dan membaca angka baris demi baris. Fungsi `getvectorline()` akan membaca angka dari sebaris data. Secara default, fungsi ini mengharapkan titik desimal. Namun, fungsi ini juga dapat menggunakan koma desimal, jika Anda memanggil `setdecimaldot(",")` sebelum menggunakan fungsi ini.

Fungsi berikut adalah contohnya. Fungsi ini akan berhenti di akhir berkas atau baris kosong.

```
>function myload (file) ...
  open(file);
  M=[];
  repeat
    until eof();
    v=getvectorline(3);
    if length(v)>0 then M=M_v; else break; endif;
  end;
  return M;
  close(file);
endfunction
```

```
>myload(file)
```

```
0.86445    0.8234    0.79024
0.59062    0.16769    0.62631
0.25795    0.84554    0.72172
```

Semua angka dalam berkas itu juga dapat dibaca dengan `getvector()`.

```
>open(file); v=getvector(10000); close(); redim(v[1:9],3,3)
```

```
0.86445    0.8234    0.79024
0.59062    0.16769    0.62631
0.25795    0.84554    0.72172
```

Jadi sangat mudah untuk menyimpan vektor nilai, satu nilai di setiap baris dan membaca kembali vektor ini.

```
>v=random(1000); mean(v)
```

```
0.51267
```

```
>writematrix(v',file); mean(readmatrix(file)')
```

```
0.51267
```

Menggunakan Tabel

Tabel dapat digunakan untuk membaca atau menulis data numerik. Misalnya, kita menulis tabel dengan tajuk baris dan kolom ke dalam sebuah berkas.

```
>file="test.tab"; M=random(3,3); ...
open(file,"w"); ...
writetable(M,separator=",",labc=["one","two","three"]); ...
close(); ...
printfile(file)
```

```
one,two,three
0.44, 0.86, 0.66
0.54, 0.4, 0.75
0.33, 0.89, 0.86
```

Ini dapat diimpor ke Excel.

Untuk membaca berkas di EMT, kami menggunakan `readtable()`.

```
>{M,headings}=readtable(file,>clabs); ...
writetable(M,labc=headings)
```

```
one two three
0.44 0.86 0.66
0.54 0.4 0.75
0.33 0.89 0.86
```

Menganalisis Garis

Anda bahkan dapat mengevaluasi setiap garis secara manual. Misalkan, kita memiliki garis dengan format berikut.

```
>line="2020-11-03,Tue,1'114.05"
```

```
2020-11-03,Tue,1'114.05
```

Pertama, kita dapat membuat token pada baris tersebut.

```
>vt=strtokens(line)
```

```
2020-11-03
Tue
1'114.05
```

Kemudian kita dapat mengevaluasi setiap elemen garis menggunakan evaluasi yang tepat.

```
>day(vt[1]), ...
indexof(["mon","tue","wed","thu","fri","sat","sun"],tolower(vt[2])), ...
strrepl(vt[3],"","") ()

7.3816e+05
2
1114
```

Dengan menggunakan ekspresi reguler, dimungkinkan untuk mengekstrak hampir semua informasi dari sebaris data.

Asumsikan kita memiliki baris berikut sebagai dokumen HTML.

```
>line="<tr><td>1145.45</td><td>5.6</td><td>-4.5</td><tr>"
```

```
<tr><td>1145.45</td><td>5.6</td><td>-4.5</td><tr>
```

Untuk mengekstraknya, kami menggunakan ekspresi reguler, yang mencari

- tanda kurung tutup >,
- string apa pun yang tidak mengandung tanda kurung dengan sub-kecocokan "(...)",
- tanda kurung buka dan tutup menggunakan solusi terpendek,
- lagi-lagi string apa pun yang tidak mengandung tanda kurung,
- dan tanda kurung buka <.

Ekspresi reguler agak sulit dipelajari tetapi sangat ampuh.

```
>{pos,s,vt}=strxfind(line,">([^\<]+)<.+?>([^\<]+)<");
```

Hasilnya adalah posisi kecocokan, string yang cocok, dan vektor string untuk sub-kecocokan.

```
>for k=1:length(vt); vt[k](), end;
```

```
1145.5
5.6
```

Berikut adalah fungsi yang membaca semua item numerik antara <td> dan </td>.

```
>function readtd (line) ...
v=[]; cp=0;
repeat
    {pos,s,vt}=strxfind(line,"<td.*?>(.+?)</td>",cp);
    until pos==0;
    if length(vt)>0 then v=v|vt[1]; endif;
    cp=pos+strlen(s);
end;
return v;
endfunction
```

```
>readtd(line+"<td>non-numerical</td>")
```

```
1145.45
5.6
-4.5
non-numerical
```

Situs web atau berkas dengan URL dapat dibuka di EMT dan dapat dibaca baris demi baris.

Dalam contoh ini, kami membaca versi terkini dari situs EMT. Kami menggunakan ekspresi reguler untuk memindai "Versi ..." dalam judul.

```
>function readversion () ...
urlopen("http://www.euler-math-toolbox.de/Programs/Changes.html");
repeat
  until urleof();
  s=urlgetline();
  k=strfind(s,"Version ",1);
  if k>0 then substring(s,k,strfind(s,"<",k)-1), break; endif;
end;
urlclose();
endfunction
```

```
>readversion
```

```
Version 2024-01-12
```

Input dan Output Variabel

Anda dapat menulis variabel dalam bentuk definisi Euler ke dalam file atau ke baris perintah.

```
>writevar(pi,"mypi");
```

```
mypi = 3.141592653589793;
```

Untuk pengujian, kami membuat file Euler di direktori kerja EMT.

```
>file="test.e"; ...
writevar(random(2,2),"M",file); ...
printfile(file,3)
```

```
M = [ ..
0.862419983048783, 0.8091888822526927;
0.9199870220380355, 0.3288140832209946];
```

Sekarang kita dapat memuat berkas tersebut. Berkas tersebut akan mendefinisikan matriks M.

```
>load(file); show M,
```

```
M =
0.86242  0.80919
0.91999  0.32881
```

Ngomong-ngomong, jika `writevar()` digunakan pada suatu variabel, ia akan mencetak definisi variabel dengan nama variabel ini.

```
>writevar(M); writevar(inch$)
```

```
M = [ ..
0.862419983048783, 0.8091888822526927;
0.9199870220380355, 0.3288140832209946];
inch$ = 0.0254;
```

Kita juga dapat membuka berkas baru atau menambahkannya ke berkas yang sudah ada. Dalam contoh ini, kita menambahkannya ke berkas yang dibuat sebelumnya.

```
>open(file,"a"); ...
writevar(random(2,2),"M1"); ...
writevar(random(3,1),"M2"); ...
close();
>load(file); show M1; show M2;
```

```
M1 =
  0.74977  0.39968
  0.55012  0.45253
M2 =
  0.21567
  0.45676
  0.92323
```

Untuk menghapus file apa pun gunakan `fileremove()`.

```
>fileremove(file);
```

Vektor baris dalam sebuah berkas tidak memerlukan koma, jika setiap angka berada di baris baru. Mari kita buat berkas seperti itu, tulis setiap baris satu per satu dengan `writeln()`.

```
>open(file,"w"); writeln("M = ["); ...
for i=1 to 5; writeln(""+random()); end; ...
writeln("];"); close(); ...
printfile(file)
```

```
M = [
0.669616938065
0.637593557683
0.624482568391
0.75029746386
0.370022542985
];
```

```
>load(file); M
```

```
[0.66962, 0.63759, 0.62448, 0.7503, 0.37002]
```

Contoh Soal

1.

```
>X=[50,92,68,72,84,80,96,64,70,48,88,66,56,84]
```

```
[50, 92, 68, 72, 84, 80, 96, 64, 70, 48, 88, 66, 56, 84]
```

```
>mean(X)
```

```
72.7142857143
```

2.

```
>X=[89,54,98,76,87,87,82,90,78,80]
```

```
[89, 54, 98, 76, 87, 87, 82, 90, 78, 80]
```

```
>mean(X), dev(X)
```

```
82.1  
11.8269936069
```

3.

```
>X=[68,72,65,80,99,76,57,84,88,66,82,95,73,71,68,70,86,93]
```

```
[68, 72, 65, 80, 99, 76, 57, 84, 88, 66, 82, 95, 73, 71,  
68, 70, 86, 93]
```

```
>dev(X)
```

```
11.6323023309
```

4.

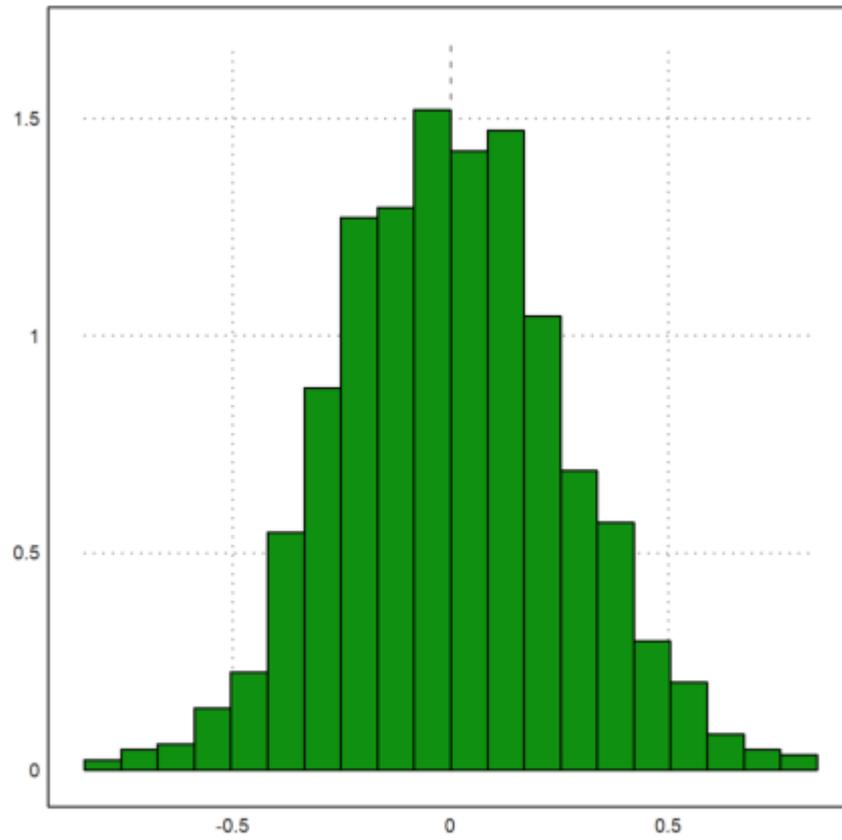
```
>longformat; 1/sqrt(15)
```

```
0.258198889747
```

```
>M=normal(1000,15); dev(mean(M)')
```

```
0.26024464726
```

```
>plot2d(mean(M) ',>distribution):
```



5.

```
>M=[100,500,473,365,852,187,943,800,765,589]; ...
  median(M), mean(M), dev(M),
```

```
544.5
557.4
285.243443785
```

```
>aspect(1.75); boxplot(M):
```

