

Di buku catatan ini, kami mendemonstrasikan plot statistik utama, pengujian, dan distribusi di Euler.

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

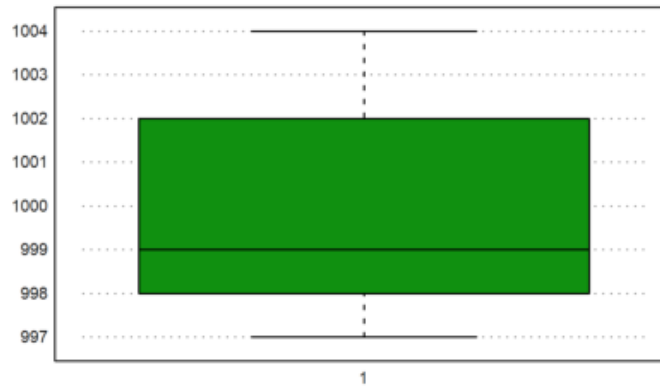
Asumsikan pengukuran berikut. Kami ingin menghitung nilai rata-rata dan deviasi standar yang diukur.

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

```
999  
999.9  
2.72641400622
```

Kita dapat memplot plot kotak-dan-kumis untuk datanya. Dalam kasus kami, tidak ada outlier.

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



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

Semua fungsi untuk distribusi di 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 hasilnya dalam % dengan akurasi 2 digit menggunakan fungsi print.

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

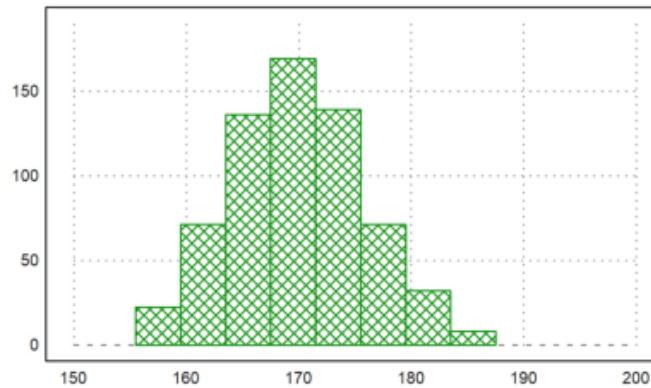
3.07 %

Untuk contoh berikutnya, kita asumsikan jumlah pria berikut dalam rentang ukuran tertentu.

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

Berikut adalah alur pendistribusiannya.

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



Kita bisa memasukkan data mentah tersebut ke dalam tabel.

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

Tabel dapat dicetak dengan header. Kami menggunakan vektor string untuk mengatur header.

```
>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 tersebut. Kita bisa menggunakan dua kolom pertama tabel kita untuk ini.

Symbol "|" digunakan untuk memisahkan kolom, fungsi "writetable" digunakan untuk menulis tabel, dengan opsi "labc" untuk menentukan header 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 akan lebih mudah jika menjumlahkan 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 mean dan deviasi sampel dengan frekuensi tertentu.

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

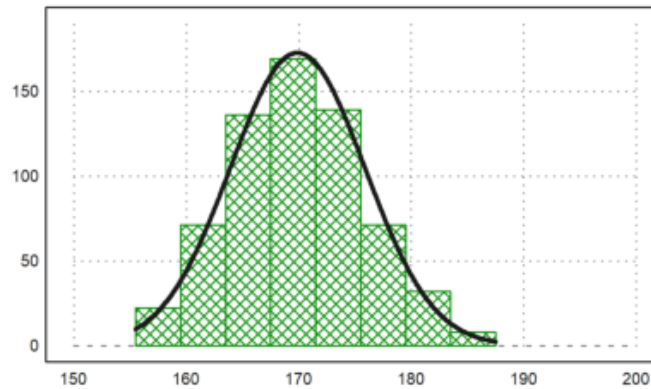
169.901234568  
5.98912964449

Mari kita tambahkan distribusi nilai normal ke diagram batang di atas. Rumus distribusi normal dengan mean  $m$  dan simpangan baku  $d$  adalah:

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

Karena nilainya antara 0 dan 1, maka untuk memplotnya pada bar plot harus dikalikan dengan 4 kali jumlah data.

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



Di direktori buku catatan ini Anda menemukan file dengan tabel. Data tersebut merupakan hasil survei. Berikut adalah empat baris pertama file tersebut. Datanya berasal dari buku online Jerman "Einführung in die Statistik mit R" oleh 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). Kami ingin membaca tabel dari file. Pertama, kami menggunakan terjemahan kami sendiri untuk tokennya.

Untuk ini, kami mendefinisikan kumpulan token. Fungsi `strtokens()` mendapatkan vektor string token dari string tertentu.

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

Sekarang kita membaca tabel dengan terjemahan ini.

Argumen tok2, tok4 dll. adalah terjemahan dari 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 kumpulan token yang sama. Kami mencetak 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-nilai, yang tidak tersedia.

Jika kita tidak ingin menentukan token yang akan diterjemahkan 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() kini mengembalikan sekumpulan token.

```
>tok
```

```
Variable tok not found!
Error in:
tok ...
^
```

Tabel berisi entri dari file dengan token yang diterjemahkan ke dalam angka.

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

```
>MT[1]
```

```
MT is not a variable!  
Error in:  
MT[1] ...  
      ^
```

Berikut isi tabel dengan nomor yang belum diterjemahkan.

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

```
Variable or function MT not found.  
Error in:  
writetable(MT,wc=5) ...  
      ^
```

Untuk kenyamanan, Anda dapat memasukkan keluaran readtable() ke dalam daftar.

```
>Table={{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;
```

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

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

```
Variable or function Table not found.
Error in:
writetable(Table,ctok=ctok,wc=5); ...
      ^
```

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

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

```
Variable or function MT not found.  
Error in:  
{c,i}=tablecol(MT,[5,6]); ...  
                ^
```

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

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

```
Variable or function i not found.  
Error in:  
j=[1,5,6]; writetable(MT[i,j],lab=hd[j],ctok=[2],tok=tok) ...  
                        ^
```

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

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

```
Table is not a variable!  
Error in:  
MT=Table[1]; ...  
      ^
```

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

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

```
Variable or function MT not found.  
Error in:  
mean(tablecol(MT,6)) ...  
      ^
```

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

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

```
Variable or function MT not found.  
Error in:  
{xu,count}=getstatistics(tablecol(MT,2)); xu, count, ...  
^
```

Kita bisa mencetak hasilnya di tabel baru.

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

```
Variable count not found!  
Error in:  
writetable(count',labr=tok[xu]) ...  
^
```

Fungsi `selecttable()` mengembalikan tabel baru dengan nilai dalam satu kolom yang dipilih dari vektor indeks. Pertama kita mencari indeks dari dua nilai kita di tabel token.

```
>v:=indexof(tok,["g","vg"])
```

```
Variable or function tok not found.  
Error in:  
v:=indexof(tok,["g","vg"]) ...  
      ^
```

Sekarang kita dapat memilih baris tabel, yang memiliki salah satu nilai `v` pada baris ke-5.

```
>MT1:=MT[selectrows(MT,5,v)]; i:=sortedrows(MT1,5);
```

```
Variable or function MT not found.  
Error in:  
MT1:=MT[selectrows(MT,5,v)]; i:=sortedrows(MT1,5); ...  
      ^
```

Sekarang kita dapat mencetak tabel, dengan nilai yang diekstraksi dan diurutkan di kolom ke-5.

```
>writetable(MT1[i],labc=hd,ctok=ctok,tok=tok,wc=7);
```

```
Variable or function i not found.  
Error in:  
writetable(MT1[i],labc=hd,ctok=ctok,tok=tok,wc=7); ...  
      ^
```

Untuk statistik selanjutnya, kami ingin menghubungkan dua kolom tabel. Jadi kita ekstrak kolom 2 dan 4 dan urutkan tabelnya.

```
>i=sortedrows(MT,[2,4]); ...  
> writetable(tablecol(MT[i],[2,4]))',ctok=[1,2],tok=tok)
```

```
Variable or function MT not found.  
Error in:  
i=sortedrows(MT,[2,4]); writetable(tablecol(MT[i],[2,4]))',c ...  
      ^
```



Dengan `getstatistics()`, kita juga bisa menghubungkan jumlah dalam 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])
```

```
Variable or function MT not found.  
Error in:  
MT24=tablecol(MT,[2,4]); {xu1,xu2,count}=getstatistics(MT24[1] ...  
~
```

Sebuah tabel dapat ditulis ke file.

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

```
Variable or function count not found.  
Error in:  
filename="test.dat"; writetable(count,labr=tok[xu1],labc=tok[x ...  
~
```

Kemudian kita bisa membaca tabel dari file tersebut.

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

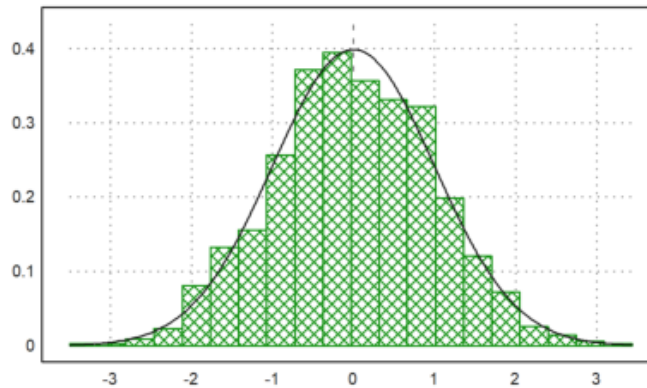
```
Could not open the file  
test.dat  
for reading!  
Try "trace errors" to inspect local variables after errors.  
readtable:  
    if filename!=none then open(filename,"r"); endif;
```

Dan hapus file tersebut.

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

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

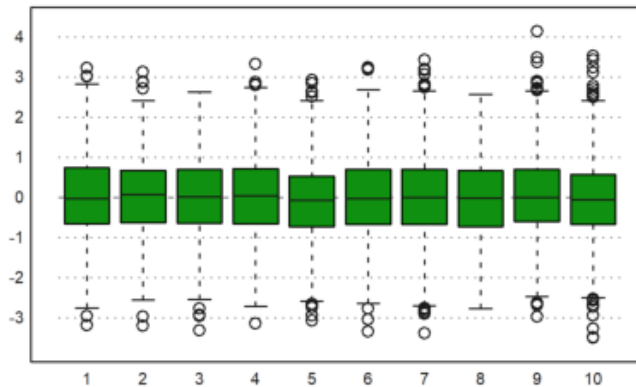
```
>p=normal(1,1000); //1000 random normal-distributed sample p  
>plot2d(p,distribution=20,style="/"); // plot the random sample p  
>plot2d("qnormal(x,0,1)",add=1): // add the standard normal distribution plot
```



Perlu diperhatikan perbedaan antara bar plot (sampel) dan kurva normal (distribusi sebenarnya). Masukkan kembali ketiga perintah untuk melihat hasil pengambilan sampel lainnya.

Berikut adalah perbandingan 10 simulasi dari 1000 nilai terdistribusi normal menggunakan apa yang disebut plot kotak. Plot ini menunjukkan median, kuartil 25% dan 75%, nilai minimal dan maksimal, serta outlier.

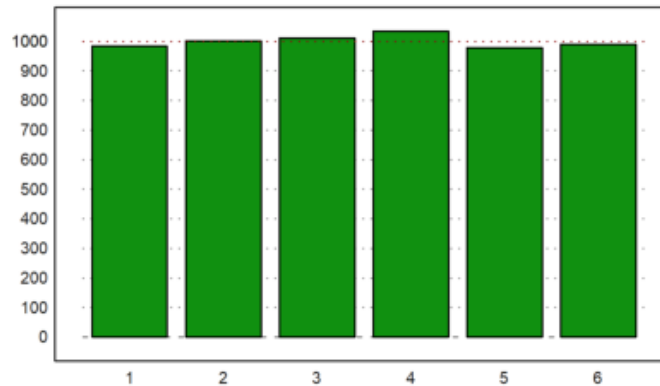
```
>p=normal(10,1000); boxplot(p):
```



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 `kolomplot()`.

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



Meskipun `inrandom(n,m,k)` mengembalikan bilangan bulat yang terdistribusi secara seragam dari 1 hingga k, distribusi bilangan bulat lainnya dapat digunakan 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.

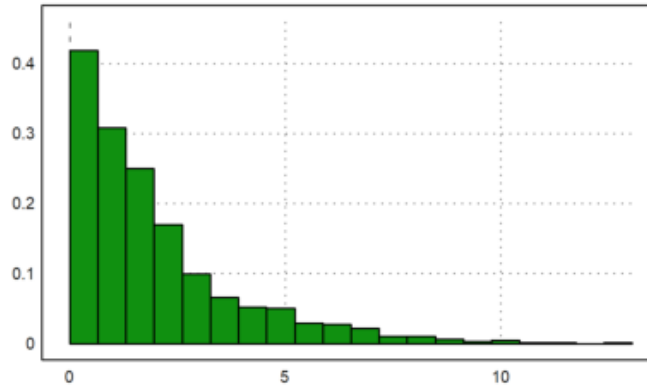
Misalnya, kita mencoba distribusi eksponensial. Variabel acak kontinu X dikatakan berdistribusi eksponensial, jika PDF-nya diberikan oleh

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

dengan parameter

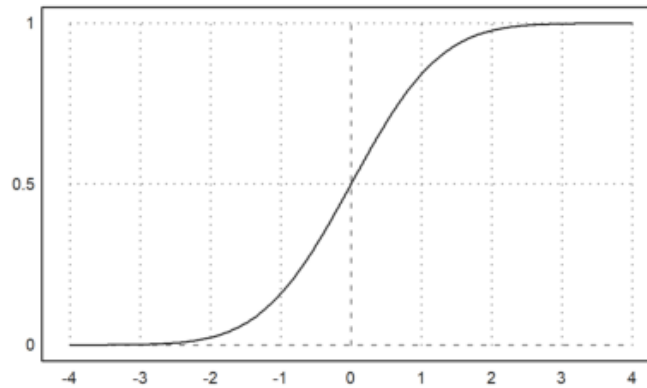
$\lambda = \frac{1}{\mu}$ ,  $\mu$  adalah mean, dan dilambangkan dengan  $X \sim \text{Exponential}(\lambda)$ .

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



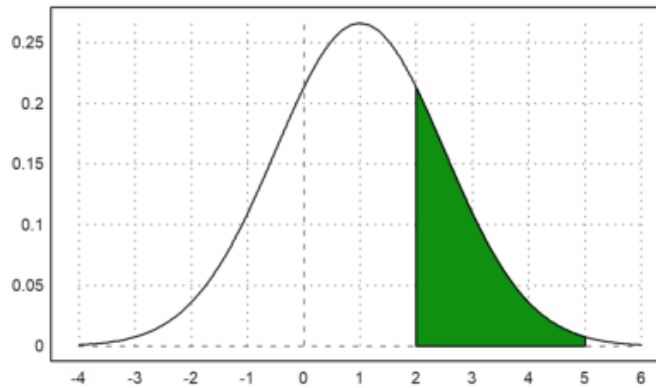
Untuk banyak distribusi, Euler dapat menghitung fungsi distribusi dan inversnya.

```
>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):
```



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

Peluang berada di kawasan hijau adalah sebagai berikut

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

0.248662156979



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
```

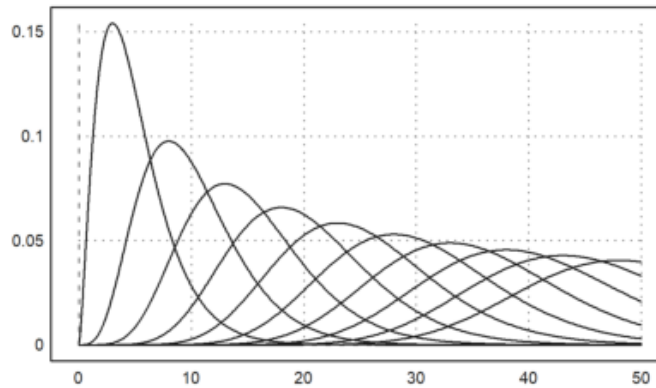
Mari kita bandingkan distribusi binomial dengan distribusi normal yang mean dan deviasinya sama. Fungsi `invbindis()` menyelesaikan interpolasi linier antara nilai integer.

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

```
525.516721219  
526.007419394
```

Fungsi `qdis()` adalah kepadatan distribusi chi-kuadrat. Seperti biasa, Euler memetakan vektor ke fungsi ini. Dengan demikian kita mendapatkan plot semua distribusi chi-kuadrat dengan derajat 5 sampai 30 dengan mudah dengan cara berikut.

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



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

Penamaannya mencoba untuk konsisten. Misalnya.,

- distribusi chi-kuadratnya adalah `chidis()`,
- fungsi kebalikannya adalah `invchidis()`,
- kepadatannya adalah `qchidis()`.

Pelengkap 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 atur fungsi distribusinya.

```
>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]
```

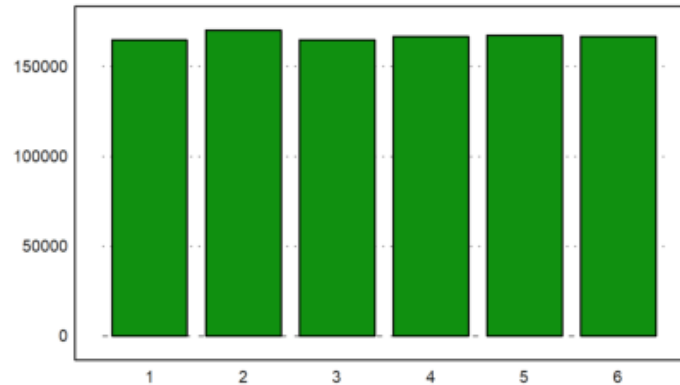
Artinya dengan probabilitas  $wd[i+1]-wd[i]$  kita menghasilkan nilai acak  $i$ .

Ini hampir merupakan distribusi yang seragam. Mari kita tentukan generator nomor acak untuk ini. Fungsi `find(v,x)` mencari nilai  $x$  pada vektor  $v$ . Fungsi ini juga berfungsi untuk vektor  $x$ .

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

Kesalahannya sangat halus sehingga kita hanya melihatnya dengan banyak iterasi.

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



Berikut adalah fungsi sederhana untuk memeriksa keseragaman distribusi nilai 1...K dalam v. Kita menerima 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 fungsinya menolak distribusi seragam.

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

0

Dan ia menerima generator acak bawaan.

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

1

Kita dapat menghitung distribusi binomial. Pertama ada `binomialsun()`, yang mengembalikan probabilitas  $i$  atau kurang hit dari  $n$  percobaan.

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

0.751401349654

Fungsi Beta terbalik digunakan untuk menghitung interval kepercayaan Clopper-Pearson untuk parameter  $p$ . Tingkat defaultnya adalah alfa.

Arti dari interval ini adalah jika  $p$  berada di luar interval, hasil pengamatan 410 dalam 1000 jarang terjadi.

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

```
[0.37932, 0.441212]
```

Perintah berikut adalah cara langsung untuk mendapatkan hasil di atas. Namun untuk  $n$  yang besar, penjumlahan langsungnya 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 `binomsum()`.

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

```
409.932733047
```

Di Bridge, kami mengasumsikan 5 kartu beredar (dari 52) di 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 memetakan data, kami mencoba hasil pemilu Jerman sejak tahun 1990, 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 pesta, kami menggunakan rangkaian nama.

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



Mari kita cetak persentasenya dengan baik.

Pertama kita mengekstrak kolom yang diperlukan. Kolom 3 sampai 7 adalah kursi masing-masing partai, dan kolom 2 adalah jumlah kursi seluruhnya. kolom adalah tahun pemilihan.

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

Kemudian statistiknya kita cetak dalam bentuk tabel. Kami menggunakan nama sebagai header kolom, dan tahun sebagai header untuk baris. Lebar default untuk kolom adalah wc=10, tetapi kami lebih memilih 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

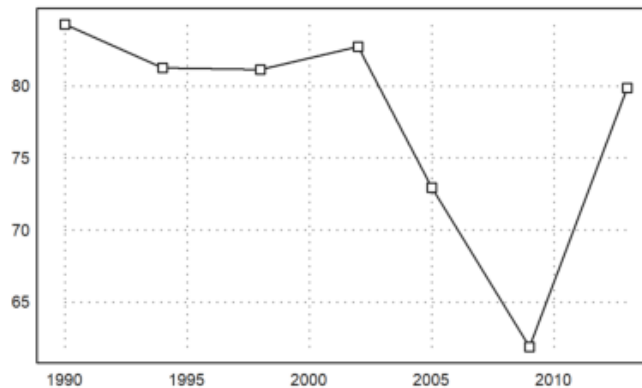
Perkalian matriks berikut ini menjumlahkan persentase dua partai besar yang menunjukkan bahwa partai-partai kecil berhasil memperoleh suara di parlemen hingga tahun 2009.

```
>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. Kami menggunakannya untuk menampilkan garis dan titik secara bersamaan. Alternatifnya adalah memanggil plot2d dua kali dengan >add.

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

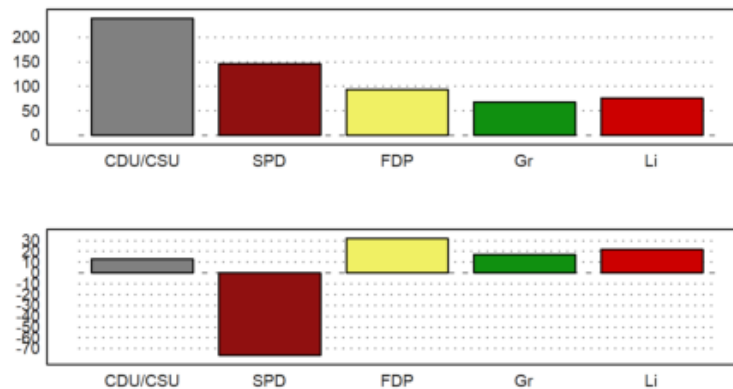


Tentukan beberapa warna untuk setiap pesta

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

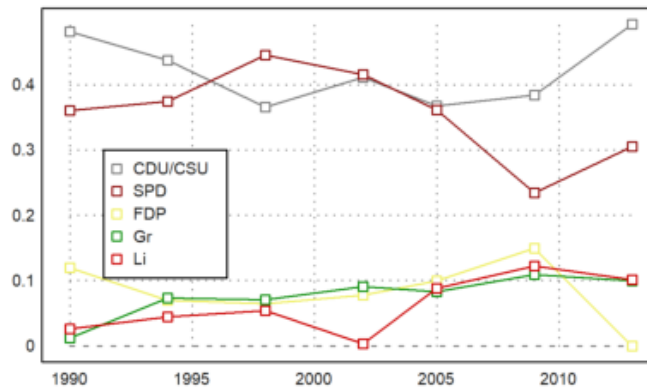
Sekarang kita bisa memplot hasil pemilu 2009 dan perubahannya menjadi satu plot dengan menggunakan gambar. Kita dapat menambahkan vektor kolom ke setiap plot.

```
>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):
```



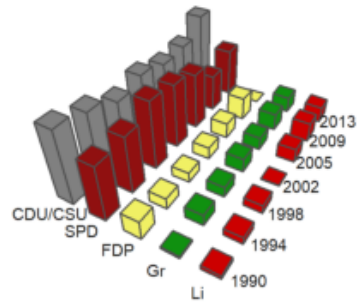
Plot data menggabungkan deretan data statistik dalam satu plot.

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



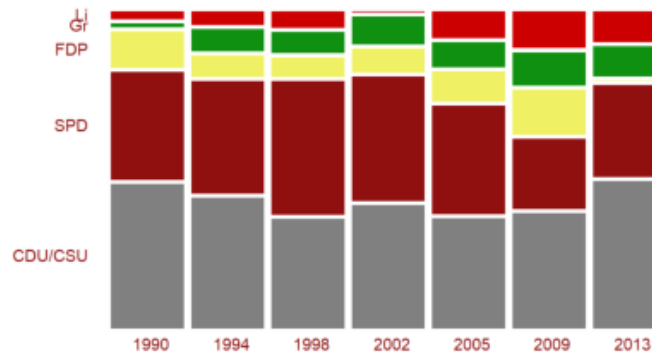
Plot kolom 3D memperlihatkan baris data statistik dalam bentuk kolom. Kami memberikan label untuk baris dan kolom. sudut adalah sudut pandang.

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



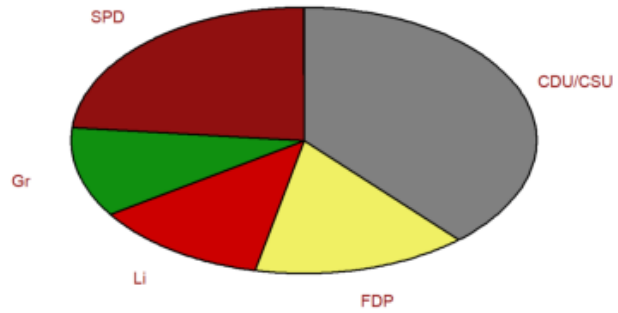
Representasi lainnya adalah plot mosaik. Perhatikan bahwa kolom plot mewakili kolom matriks di sini. Karena panjang label CDU/CSU, kami mengambil jendela yang lebih kecil dari biasanya.

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



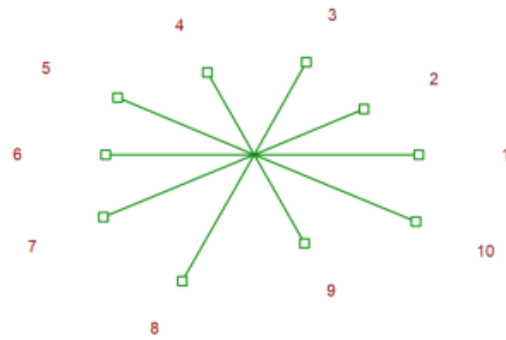
Kita juga bisa membuat diagram lingkaran. Karena hitam dan kuning membentuk koalisi, kami menyusun ulang elemen-elemennya.

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



Ini adalah jenis plot lainnya.

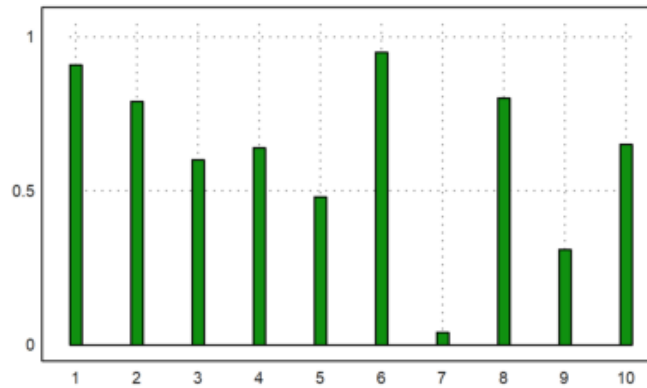
```
>starplot(normal(1,10)+4,lab=1:10,>rays):
```



Beberapa plot di plot2d bagus untuk statika. Berikut adalah plot impuls dari data acak, terdistribusi secara seragam di  $[0,1]$ .

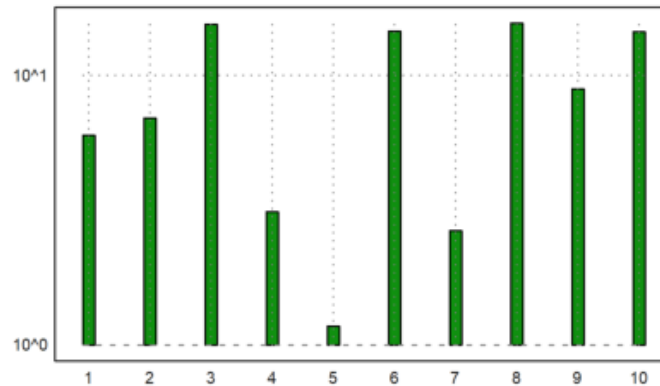
```
>plot2d(makeimpulse(1:10,random(1,10)),>bar):
```





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

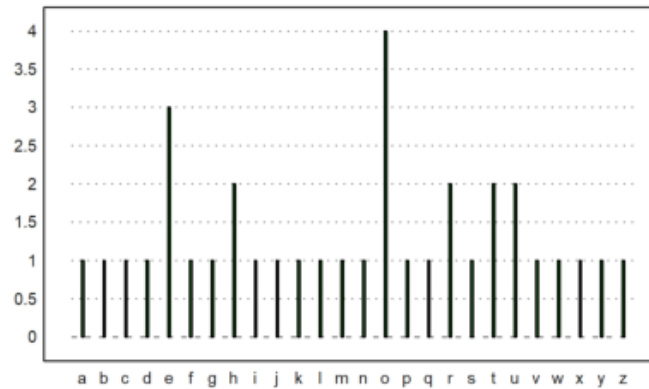
```
>logimpulseplot(1:10,-log(random(1,10))*10):
```



Fungsi Columnplot() lebih mudah digunakan, karena hanya memerlukan vektor nilai. Selain itu, ia dapat mengatur labelnya ke apa pun yang kita inginkan, kami telah mendemonstrasikannya di tutorial ini.

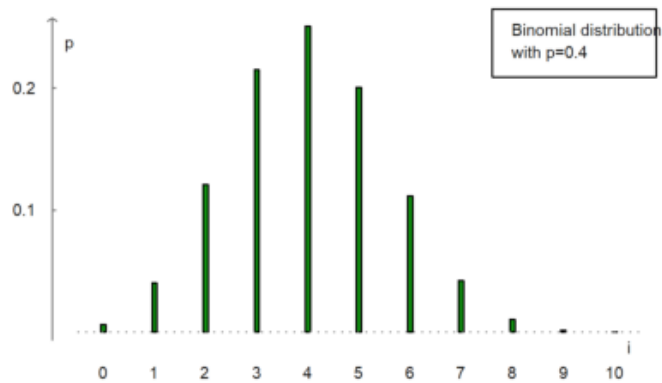
Ini adalah aplikasi lain, di mana kita menghitung karakter dalam sebuah kalimat dan membuat statistik.

```
>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; ...
>columnplot(x,lab=cw,width=0.05):
```



Dimungkinkan juga untuk mengatur sumbu secara manual.

```
>n=10; p=0.4; i=0:n; x=bin(n,i)*p^i*(1-p)^(n-i); ...
>columnsplot(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 cara memplot frekuensi bilangan dalam suatu vektor.  
Kami membuat vektor bilangan acak bilangan bulat 1 hingga 6.

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

```
[8, 5, 8, 8, 6, 8, 8, 3, 5, 5]
```

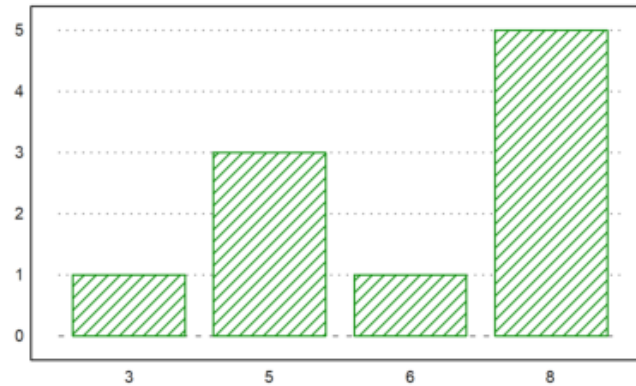
Kemudian ekstrak nomor unik di v.

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

```
[3, 5, 6, 8]
```

Dan plot frekuensi dalam plot kolom.

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



Kami ingin mendemonstrasikan fungsi distribusi nilai empiris.

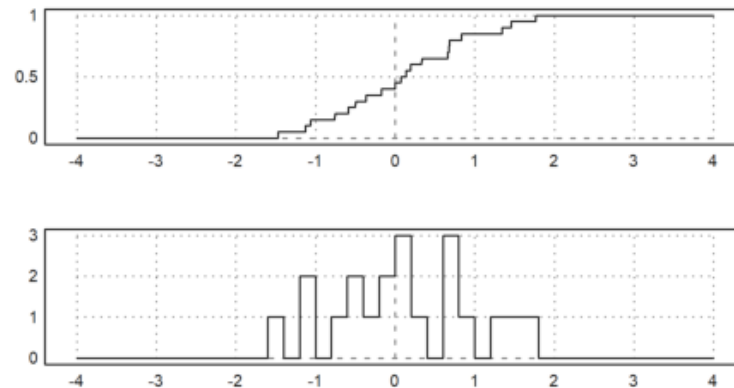
```
>x=normal(1,20);
```

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

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

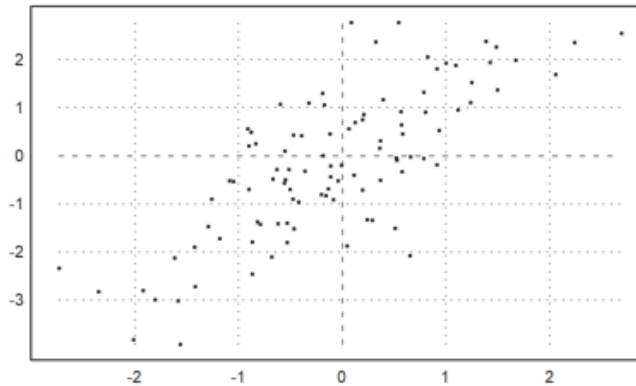
Kemudian kita plot distribusi empiris dan beberapa batang kepadatan ke dalam satu plot. Alih-alih plot batang untuk distribusi kali ini kami menggunakan plot gigi gergaji.

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



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

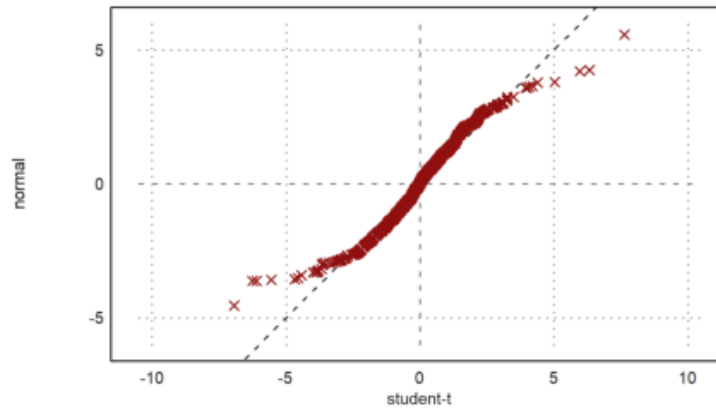
```
>x=normal(1,100); plot2d(x,x+rotright(x),>points,style=".."):
```



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

Untuk pengujiannya, kami mencoba distribusi student-t 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 terdistribusi normal cenderung lebih kecil di ujung ekstrim.

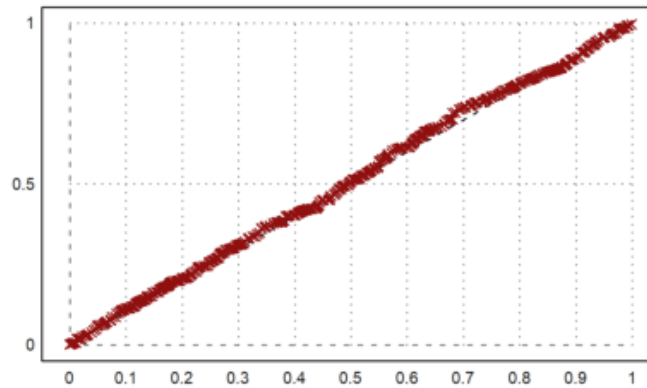
Jika kita mempunyai dua distribusi yang ukurannya berbeda, kita dapat memperluas distribusi yang lebih kecil atau mengecilkan distribusi yang lebih besar. Fungsi berikut ini baik untuk keduanya. Dibutuhkan nilai median dengan persentase antara 0 dan 1.

```
>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 menemukan 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 berbobot dan berbobot. Pertama koefisien kecocokan linier.

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

```
[0.733333, 0.812121]
```

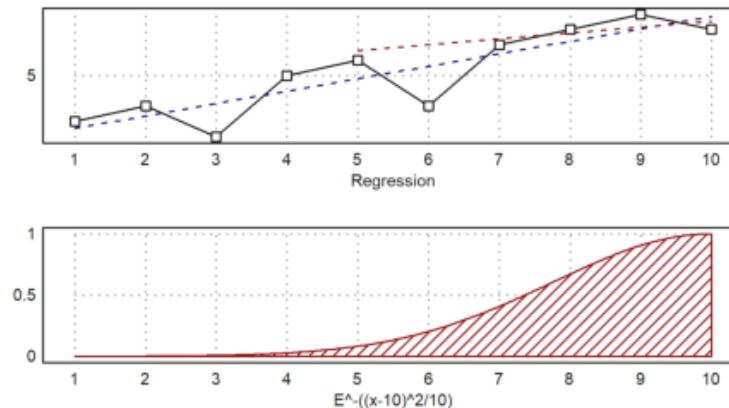
Sekarang koefisien dengan bobot yang menekankan nilai terakhir.

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

```
[4.71566, 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):
```



Contoh lain kita membaca survei siswa, usia mereka, usia orang tua mereka dan jumlah saudara kandung dari sebuah file.

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

```
>{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 bergantung satu sama lain? Kesan pertama muncul dari plot sebar 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 terlihat bahwa usia ayah dan ibu saling bergantung satu sama lain. Mari kita tentukan dan plot 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 merupakan model yang salah. Garis regresinya adalah  $s=17+0,74t$ , dengan  $t$  adalah umur ibu dan  $s$  adalah umur ayah. Perbedaan usia mungkin sedikit bergantung pada usia, tapi tidak terlalu banyak.

Sebaliknya, kami mencurigai fungsi seperti  $s=a+t$ . Maka  $a$  adalah mean dari  $s-t$ . Ini 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 plot ini menjadi satu plot 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 adalah plot kotak dari dua zaman tersebut. Ini hanya menunjukkan, bahwa usianya berbeda-beda.

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

```
Variable or function cs not found.  
Error in:  
boxplot(cs,["mothers","fathers"]): ...  
      ^
```

Menariknya, perbedaan median tidak sebesar perbedaan mean.

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

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

Koefisien korelasi menunjukkan korelasi positif.

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

```
cs is not a variable!  
Error in:  
correl(cs[1],cs[2]) ...  
      ^
```

Korelasi pangkat merupakan ukuran keteraturan yang sama pada kedua vektor. Hal ini juga cukup positif.

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

```
cs is not a variable!  
Error in:  
rankcorrel(cs[1],cs[2]) ...  
      ^
```



## Membuat Fungsi baru

---

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

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

dimana  $m$  adalah mean dari  $x$ .

```
>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))
```

```
-0.198710316203
```

Berikut adalah fungsi lainnya, yang disebut koefisien skewness Pearson.

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

-0.0801873249135

## Simulasi Monte Carlo

---

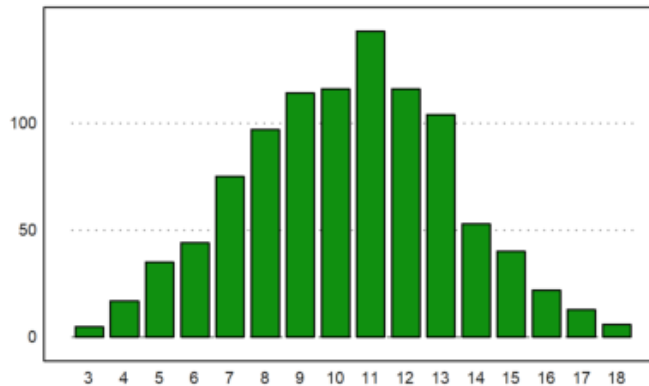
Euler dapat digunakan untuk mensimulasikan kejadian acak. Kita telah melihat contoh sederhana di atas. Ini satu lagi, yang mensimulasikan 1000 kali lemparan 3 dadu, dan menanyakan pembagian jumlahnya.

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

```
[5, 17, 35, 44, 75, 97, 114, 116, 143, 116, 104, 53, 40,  
22, 13, 6]
```

Kita bisa merencanakannya sekarang.

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



Untuk menentukan distribusi yang diharapkan tidaklah mudah. Kami menggunakan rekursi tingkat lanjut untuk ini.

Fungsi berikut menghitung banyaknya cara bilangan  $k$  dapat direpresentasikan sebagai jumlah dari  $n$  bilangan dalam rentang 1 sampai  $m$ . Ia 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 hasil pelemparan dadu sebanyak tiga kali.

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

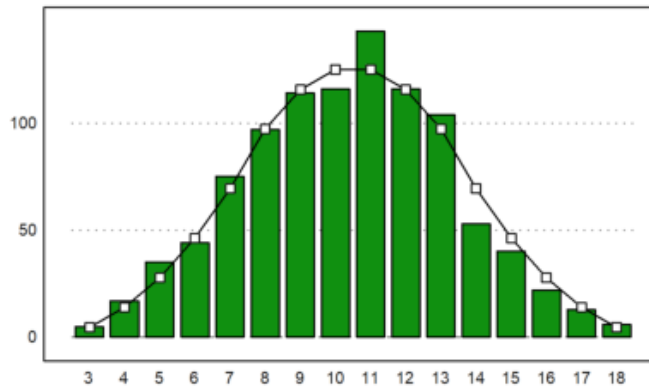
```
[1, 5, 15, 35, 70, 121, 185, 255, 320, 365, 381, 365, 320,  
255, 185, 121, 70, 35, 15, 5, 1]
```

```
>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 n 0-1-variabel acak terdistribusi normal adalah  $1/\sqrt{n}$ .

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

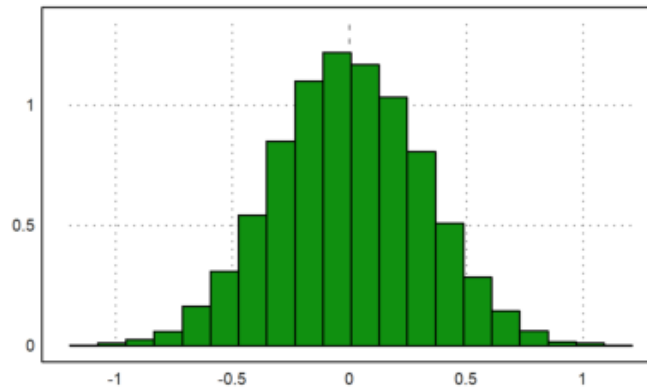
0.316227766017

Mari kita periksa ini dengan simulasi. Kami menghasilkan 10.000 kali 10 vektor acak.

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

0.319493614817

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



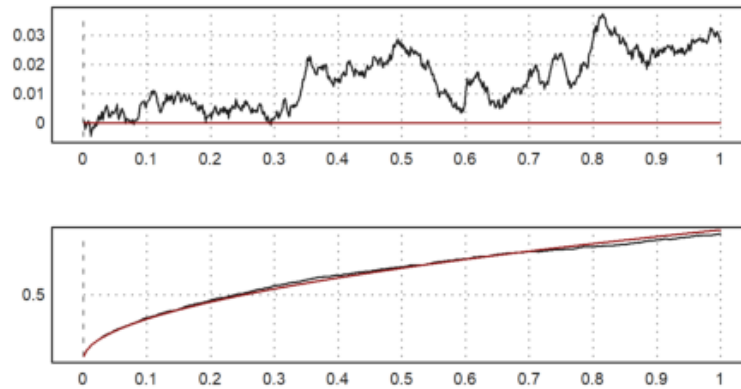
Median dari 10 bilangan acak berdistribusi normal 0-1 mempunyai deviasi yang lebih besar.

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

0.374460271535

Karena kita dapat dengan mudah menghasilkan jalan acak, kita dapat mensimulasikan proses Wiener. Kami mengambil 1000 langkah dari 1000 proses. Kami kemudian memplot deviasi standar dan rata-rata langkah ke-n dari proses ini bersama dengan nilai yang diharapkan berwarna 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):
```





Tes adalah alat penting dalam statistik. Di Euler, banyak tes yang diterapkan. Semua pengujian ini mengembalikan kesalahan yang kita terima jika kita menolak hipotesis nol.

Misalnya, kami menguji lemparan dadu untuk distribusi yang seragam. Pada 600 kali lemparan, kami mendapatkan nilai berikut, yang kami masukkan ke dalam uji chi-kuadrat.

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

```
0.498830517952
```

Uji chi-kuadrat juga memiliki mode yang menggunakan simulasi Monte Carlo untuk menguji statistiknya. Hasilnya seharusnya hampir sama. Parameter `>p` menafsirkan vektor `y` sebagai vektor probabilitas.

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

```
0.526
```

Kesalahan ini terlalu besar. Jadi kita tidak bisa menolak pemerataan. Ini tidak membuktikan bahwa dadu kita adil. Tapi kita tidak bisa menolak hipotesis kita.

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

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

0.528028118442

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

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

0.0218365848476

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

Sekarang mari kita periksa dua pengukuran untuk mean yang sama. Kami menolak hipotesis bahwa keduanya mempunyai mean yang sama, jika hasilnya  $< 0,05$ .

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

0.38722000942

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)
```

```
5.60009101758e-07
```

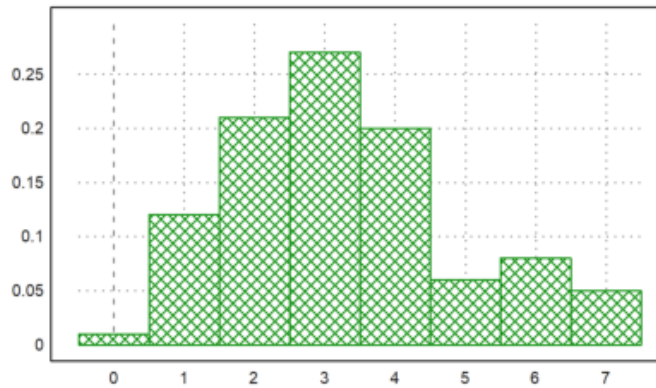
Pada contoh berikutnya, kita membuat 20 lemparan dadu acak sebanyak 100 kali dan menghitung yang ada di dalamnya. Rata-rata harus ada  $20/6=3,3$ .

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

```
3.28
```

Sekarang kita bandingkan jumlah satuan dengan distribusi binomial. Pertama kita plot distribusinya.

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



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

Kemudian kami menghitung 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-square menolak hipotesis bahwa distribusi kita merupakan distribusi binomial, jika hasilnya  $< 0,05$ .

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

0.53921579764

Contoh berikut berisi hasil dua kelompok orang (misalnya laki-laki dan perempuan) 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  $\chi^2$  melakukan hal ini. Dampaknya terlalu besar untuk menolak kemerdekaan. Jadi kita tidak bisa bilang, kalau voting tergantung jenis kelamin dari data tersebut.

```
>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, kami menyimpulkan bahwa pemungutan suara tidak bergantung pada jenis kelamin.

```
>contingency(A)
```

0.0427225484717

## Beberapa Tes Lagi

---

Selanjutnya kita menggunakan analisis varians (uji F) untuk menguji tiga sampel data yang berdistribusi normal untuk nilai mean yang sama. Metode tersebut disebut ANOVA (analisis varians). Di 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, kami menolak hipotesis nilai mean yang sama. Kami melakukan ini dengan probabilitas kesalahan 1,3%.

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

```
>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
```

Tes kesetaraan lainnya adalah tes peringkat. Ini jauh lebih tajam daripada tes median.

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

```
0.00199969612469
```

Pada contoh berikut, kedua distribusi mempunyai mean yang sama.

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

```
0.129608141484
```



Sekarang mari kita coba mensimulasikan dua perlakuan 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];
```

Tes signum memutuskan, apakah a lebih baik dari b.

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

```
0.0546875
```

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

Uji Wilcoxon lebih tajam dibandingkan uji ini, namun mengandalkan nilai kuantitatif perbedaannya.

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

```
0.0296680599405
```

Mari kita coba dua tes lagi menggunakan rangkaian yang dihasilkan.

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

0.0068706451766

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

0.275145971064

## Angka Acak

---

Berikut ini adalah pengujian pembangkit bilangan acak. Euler menggunakan generator yang sangat bagus, jadi kita tidak perlu mengharapkan adanya masalah.

Pertama kita menghasilkan sepuluh juta angka acak di  $[0,1]$ .

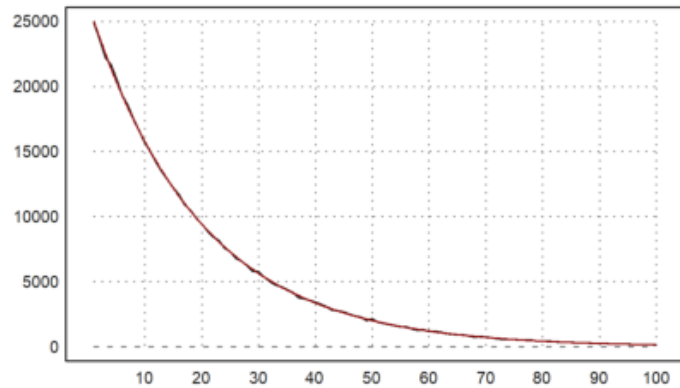
```
>n:=10000000; r:=random(1,n);
```

Selanjutnya kita hitung jarak antara dua angka yang 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 datanya.

```
>remvalue n;
```

## Pengantar untuk Pengguna Proyek R

---

Jelasnya, EMT tidak bersaing dengan R sebagai paket statistik. Namun, ada banyak prosedur dan fungsi statistik yang tersedia di EMT juga. Jadi EMT dapat memenuhi kebutuhan dasar. Bagaimanapun, EMT hadir dengan paket numerik dan sistem aljabar komputer.

Notebook ini cocok untuk Anda yang sudah familiar dengan R, namun perlu mengetahui perbedaan sintaksis EMT dan R. Kami mencoba memberikan gambaran umum tentang hal-hal yang sudah jelas dan kurang jelas yang perlu Anda ketahui.

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

Perhatikan bahwa ini masih dalam proses.

**Sintaks Dasar**

---

Hal pertama yang Anda pelajari di R adalah membuat vektor. Dalam EMT, perbedaan utamanya adalah operator : dapat mengambil ukuran langkah. Selain itu, ia mempunyai daya ikat 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 untuk menggabungkan sesuatu.

Contoh berikut, seperti banyak contoh lainnya, berasal dari "Interoduksi ke R" yang disertakan dengan proyek R. Jika Anda membaca PDF ini, Anda akan menemukan bahwa saya mengikuti jalurnya 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 di 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 di EMT. Untuk masukan vektor dapat dituliskan 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 tugas. Operator "->" digunakan untuk satuan dalam EMT.

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

```
77.6713990297 miles
```

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

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

```
0
```

Di R, "a<-4<3" berfungsi, tetapi "a<-4<-3" tidak. Saya juga memiliki ambiguitas serupa di EMT, tetapi saya mencoba menghilangkannya sedikit demi sedikit.

EMT dan R memiliki vektor bertipe boolean. Namun dalam EMT, angka 0 dan 1 digunakan untuk mewakili salah dan benar. Di R, nilai benar dan salah tetap bisa digunakan dalam aritmatika biasa seperti di 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
```



Stringnya 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 pengkodean lokal dan sebaliknya. Selain itu, u"..." dapat berisi entitas HTML.

```
>u"&#169; Ren&eacute; Grothmann"
```

© René Grothmann

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

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

Penggabungan string dilakukan dengan "+" atau "|". Ini bisa berisi angka, yang akan dicetak dalam format saat ini.

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

pi = 3.14159265359

## Pengindeksan

---

Seringkali, ini akan berfungsi seperti di R.

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

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

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

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

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

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

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

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

```
[10.4,  5.6,  3.1,  6.4,  21.7]  
[1,  1,  0,  1,  1]  
[10.4,  5.6,  6.4,  21.7]
```

Sama seperti di R, vektor indeks dapat berisi pengulangan.

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

```
[10.4,  5.6,  5.6,  10.4]
```

Namun penamaan indeks tidak dimungkinkan di EMT. Untuk paket statistik, hal ini sering kali diperlukan untuk memudahkan akses ke elemen vektor.

Untuk meniru perilaku ini, kita dapat mendefinisikan suatu 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)]; ... ...  
      ^
```

```
[10.4,  3.1]
```

## Tipe Data

---

EMT memiliki lebih banyak tipe data tetap daripada R. Jelasnya, di R terdapat vektor yang berkembang. Anda dapat mengatur vektor numerik kosong `v` dan memberikan 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` ditambahkan pada tumpukan dan menyalin vektor tersebut kembali ke variabel global `v`.

Semakin efisien vektor telah ditentukan sebelumnya.

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

Untuk mengubah tipe tanggal di EMT, Anda dapat menggunakan fungsi seperti `kompleks()`.

```
>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. Tapi 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 ada contoh yang disebut faktor.

Berikut ini adalah daftar wilayah 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, kami ingin menghitung rata-rata pendapatan di suatu wilayah. Menjadi program statistik, R memiliki faktor() dan tapply() untuk ini.

EMT dapat melakukan hal ini dengan menemukan indeks wilayah dalam daftar wilayah 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 itu, kita dapat menulis fungsi perulangan kita sendiri untuk melakukan sesuatu hanya untuk satu faktor.

Atau kita bisa 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
```

Ini 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.

Sehingga kita bisa mencetak rata-rata pendapatan negara di daerah secara bersahabat.

```
>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 bisa mencoba meniru perilaku R sepenuhnya.

Faktor-faktor tersebut harus disimpan dengan jelas dalam kumpulan beserta jenis dan kategorinya (negara bagian dan teritori dalam contoh kita). Untuk EMT, kami menambahkan 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 bisa meniru `tapply()` dengan cara berikut. 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 pengecekan tipe di sini. Satu-satunya tindakan pencegahan menyangkut kategori (faktor) yang tidak memiliki data. Tetapi kita harus memeriksa panjang `t` yang benar dan kebenaran pengumpulan `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 datanya disebut matriks. Namun, akan mudah untuk menulis fungsi untuk dimensi yang lebih tinggi atau perpustakaan C untuk ini.

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

Dalam EMT, vektor adalah matriks dengan satu baris. Itu 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, mirip dengan R.

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

2	3
7	8
12	13
17	18

Namun, di R dimungkinkan untuk menyetel daftar indeks vektor tertentu ke suatu nilai. Hal yang sama mungkin terjadi di EMT hanya dengan satu putaran.

```
>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 di EMT. Jika Anda tidak ingin mengubah matriks M asli, Anda perlu menyalinnya ke dalam fungsi.

```
>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

Perkalian luar dalam EMT hanya dapat dilakukan antar vektor. Ini otomatis karena bahasa matriks. Satu vektor harus berupa vektor kolom dan vektor lainnya harus berupa vektor baris.

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

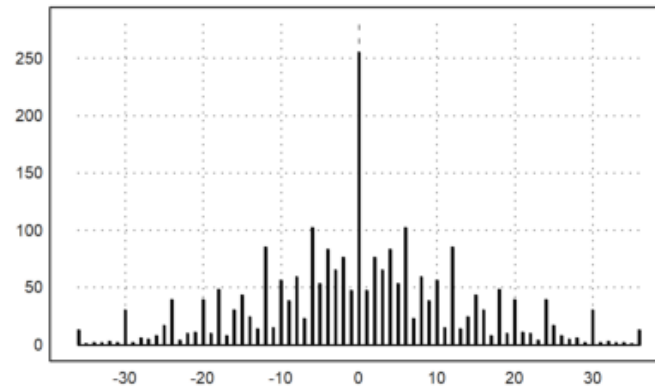
1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

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

Tentu saja, hal ini dapat dicapai dengan satu putaran. Tapi loop tidak efektif di EMT atau R. Di EMT, kita bisa menulis loop di C dan itu akan menjadi solusi tercepat.

Namun kita ingin meniru perilaku R. Untuk melakukannya, 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 eksak, EMT dapat menghitung frekuensi dalam vektor.

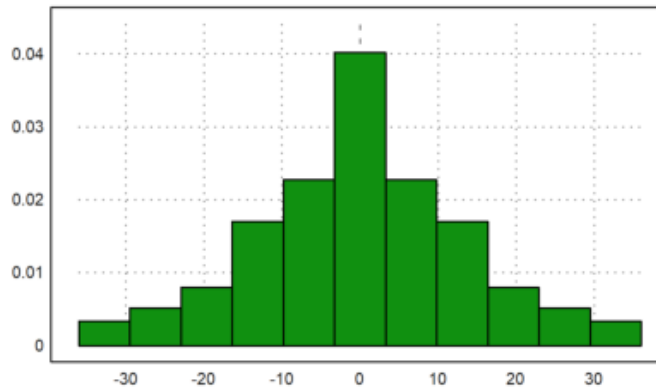
```
>getfrequencies(q,-50:10:50)
```

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



Cara paling mudah untuk memplotnya sebagai distribusi adalah sebagai berikut.

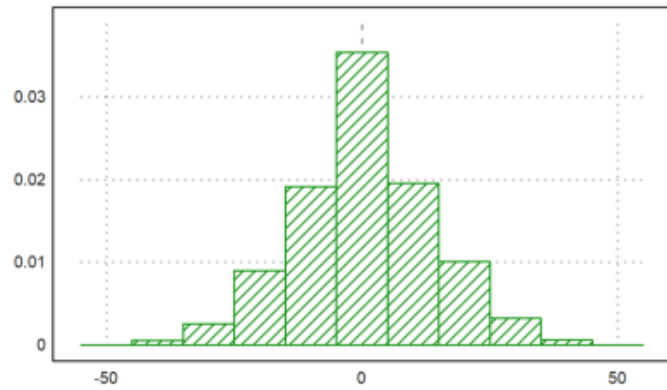
```
>plot2d(q,distribution=11):
```



Namun dimungkinkan juga untuk menghitung terlebih dahulu penghitungan dalam interval yang dipilih sebelumnya. 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. Salah satunya adalah daftar global yang bisa berubah, dan yang lainnya adalah tipe daftar yang tidak bisa diubah. Kami tidak peduli dengan daftar global di sini.

Tipe daftar yang tidak dapat diubah disebut koleksi di EMT. Ini berperilaku seperti struktur di 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. Mereka diakses dengan nomor.

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

```
1992
```

## File Input dan Output (Membaca dan Menulis Data)

---

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

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

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

```
0.49815
```

```
0.28037
```

Untuk menulis data ke file, kita menggunakan fungsi `writematrix()`.

Karena pengenalan ini kemungkinan besar ada di direktori, di mana pengguna tidak memiliki akses tulis, kami menulis data ke direktori home pengguna. Untuk buku catatan sendiri, hal ini tidak diperlukan, karena file data akan ditulis ke dalam direktori yang sama.

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

Sekarang kita menulis vektor kolom a' ke file. Ini menghasilkan satu nomor di setiap baris file.

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

Untuk membaca data, kami menggunakan readmatrix().

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

Dan hapus file tersebut.

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

```
0.49815  
0.28037
```

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

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

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

Anda sekarang dapat membuka file ini dengan Excel bahasa Indonesia secara langsung.

```
>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 melakukan tokenisasi ini, kami mendefinisikan vektor token.

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

```
f  
m
```

Kemudian 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 header 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();
```

Filenya terlihat seperti ini.

```
>printfile(file)
```

```
A,B,C  
0.7003664386138074,0.1875530821001213,0.3262339279660414  
0.5926249243193858,0.1522927283984059,0.368140583062521  
0.8065535209872989,0.7265910840408142,0.7332619844597152
```

Fungsi `readtable()` dalam bentuknya yang paling sederhana dapat membaca ini dan mengembalikan kumpulan nilai dan baris judul.

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

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

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

A	B	C
0.70037	0.18755	0.32623
0.59262	0.15229	0.36814
0.80655	0.72659	0.73326



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

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

```
0.40472
```

```
0.37102
```

```
0.75547
```

## File CSV

---

Pertama, mari kita menulis matriks ke dalam file. Untuk outputnya, kami membuat file di direktori kerja saat ini.

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

Berikut isi file ini.

```
>printfile(file)
```

```
0.8221197733097619,0.821531098722547,0.7771240608094004  
0.8482947121863489,0.3237767724883862,0.6501422353377985  
0.1482301827518109,0.3297459716109594,0.6261901074210923
```

CVS ini dapat dibuka pada sistem berbahasa Inggris ke Excel dengan klik dua kali. Jika Anda mendapatkan file seperti itu di sistem 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 file dengan `readmatrix()`.

```
>readmatrix(file)
```

0.82212	0.82153	0.77712
0.84829	0.32378	0.65014
0.14823	0.32975	0.62619

Dimungkinkan untuk menulis beberapa matriks ke satu file. Perintah `open()` dapat membuka file untuk ditulis dengan parameter "w". Standarnya adalah "r" untuk membaca.

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

Matriks dipisahkan oleh garis kosong. Untuk membaca matriks, buka file 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 spreadsheet serupa, Anda dapat mengekspor matriks sebagai CSV (nilai yang dipisahkan koma). Di Excel 2007, gunakan "save as" dan "other format", lalu pilih "CSV". Pastikan tabel saat ini hanya berisi data yang ingin Anda ekspor.

Ini sebuah contoh.

```
>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 bahasa Jerman saya menggunakan titik koma sebagai pemisah dan koma desimal. Anda dapat mengubahnya di pengaturan sistem atau di Excel, tetapi hal ini tidak diperlukan untuk membaca matriks menjadi EMT.

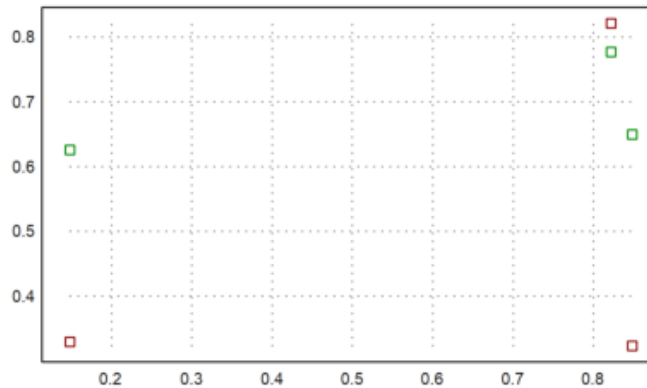
Cara termudah untuk membaca ini ke dalam Euler adalah readmatrix(). Semua koma diganti dengan titik dengan parameter >koma. Untuk CSV bahasa Inggris, hilangkan saja parameter ini.

```
>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 rencanakan ini.

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



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

Fungsi berikut adalah contohnya. Itu akan berhenti di akhir file 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)
```

```

0.82212      0  0.82153      0  0.77712
0.84829      0  0.32378      0  0.65014
0.14823      0  0.32975      0  0.62619

```

Dimungkinkan juga untuk membaca semua angka dalam file itu dengan `getvector()`.

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

```

0.82212      0  0.82153
      0  0.77712  0.84829
      0  0.32378      0

```

Oleh karena itu sangat mudah untuk menyimpan suatu vektor nilai, satu nilai di setiap baris dan membaca kembali vektor ini.

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

0.50303

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

0.50303

## Menggunakan Tabel

---

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

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

one	two	three
0.09,	0.39,	0.86
0.39,	0.86,	0.71
0.2,	0.02,	0.83

Ini dapat diimpor ke Excel.

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

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

one	two	three
0.09	0.39	0.86
0.39	0.86	0.71
0.2	0.02	0.83



## Menganalisis Garis

---

Anda bahkan dapat mengevaluasi setiap baris dengan tangan. Misalkan, kita memiliki baris dengan format berikut.

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

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

Pertama, kita dapat memberi token pada garis tersebut.

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

```
2020-11-03
```

```
Tue
```

```
1'114.05
```

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

```
>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 sebuah 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-pencocokan "(...)",

- braket pembuka dan penutup menggunakan solusi terpendek,
- sekali 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
```

## Membaca dari Web

---

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

Dalam contoh, kita membaca versi terkini dari situs EMT. Kami menggunakan ekspresi reguler untuk memindai "Versi ..." dalam sebuah 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 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 = [ ..  
0.5991820585590205, 0.7960280262224293;  
0.5167243983231363, 0.2996684599070898];
```

Sekarang kita dapat memuat file tersebut. Ini akan mendefinisikan matriks M.

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

```
M =  
0.59918  0.79603  
0.51672  0.29967
```

Omong-omong, jika `writevar()` digunakan pada suatu variabel, definisi variabel dengan nama variabel tersebut akan dicetak.

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

```
M = [ ..  
0.5991820585590205, 0.7960280262224293;  
0.5167243983231363, 0.2996684599070898];  
inch$ = 0.0254;
```

Kita juga bisa membuka file baru atau menambahkan file yang sudah ada. Dalam contoh kita menambahkan file yang dibuat sebelumnya.

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

```
M1 =  
  0.30287  0.15372  
  0.7504   0.75401  
M2 =  
  0.27213  
  0.053211  
  0.70249
```

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

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

Vektor baris dalam suatu file tidak memerlukan koma, jika setiap angka berada pada baris baru. Mari kita buat file 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 = [  
0.344851384551  
0.0807510017715  
0.876519562911  
0.754157709472  
0.688392638934  
];
```

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

```
[0.34485, 0.080751, 0.87652, 0.75416, 0.68839]
```

## Latihan Soal

---

Nomor 1

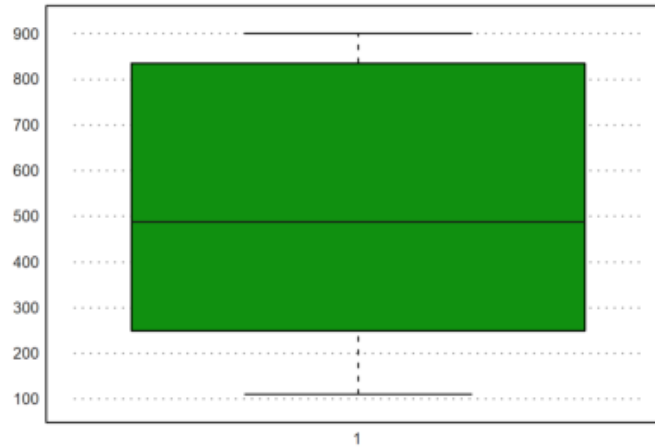
Carilah rata-rata dan standar deviasi beserta plot dari data berikut

$X = 890, 555, 420, 110, 150, 350, 780, 900$

```
>X=[890,555,420,110,150,350,780,900]; ...  
>mean(X), dev(X),
```

```
519.38  
314.87
```

```
>aspect(1.5); boxplot(X):
```



Nomor 2

Misalkan diberikan data skor hasil statistika dari 20 orang mahasiswa sebagai berikut :

89, 80, 67, 11, 98, 55, 85, 90, 79, 75, 76, 95, 87, 91, 43, 89, 22, 93, 89, 55.

Tentukan rata-rata, median, dan standar deviasi beserta plot dari data tersebut!

```
>F=[89,80,67,11,98,55,85,90,79,75,76,95,87,91,43,89,22,93,89,55]
```

```
[89, 80, 67, 11, 98, 55, 85, 90, 79, 75, 76, 95, 87, 91,
43, 89, 22, 93, 89, 55]
```

```
>mean(F)
```

73.45

```
>median(F)
```

82.5

```
>dev(F)
```

24.444

```
>aspect(1.25); boxplot(F):
```

