Feb 22, 2024

Too much talking about random numbers.
ask maple about it

```
> ?random
> ?HowDoI,WorkWithRandomGenerators
```

Most trivial is **rand()**

```
> rand( )
                              395718860534                              (1)
> rand( )
                              193139816415                              (2)
> rand( )
                               22424170465                              (3)
```

Make some dice

```
> rand(1 ..6)
   proc( ) proc( ) option builtin = RandNumberInterface;  end proc(6, 6, 3) + 1  end proc   (4)
> RollDice := rand(1 ..6)
RollDice := proc( )                                                     (5)
   proc( ) option builtin = RandNumberInterface;  end proc(6, 6, 3) + 1
end proc
> RollDice( )
                                    3                                   (6)
> RollDice( )
                                    4                                   (7)
> seq(RollDice( ), i = 1 ..20)
            4, 6, 5, 3, 1, 5, 2, 3, 2, 2, 4, 3, 3, 1, 2, 5, 4, 5, 6, 2   (8)
> randomize(31415)
                                  31415                                 (9)
> RollDice( )
                                    4                                   (10)
>  randomize(31415); RollDice( ); RollDice( )
                                  31415
                                    4
                                    6                                   (11)
> randomize( ); RollDice( ); RollDice( )
                               55005918917
                                    5
                                    4                                   (12)
```

Behind the scenes, rand() is using MersenneTwister   So what.

```
> with(RandomTools[MersenneTwister])
                                                                        (13)
```

```
[GenerateData, GenerateFloat, GenerateFloat64, GenerateInteger, GenerateInteger32,
    GenerateUnsignedInt32, GetState, NewGenerator, SetState]                        (13)
> GenerateFloat( )
                                  0.0200793866                                      (14)
> GenerateFloat( )
                                  0.4317871144                                      (15)
>  GenerateFloat64( )
                               0.969299637098994693                                 (16)
```

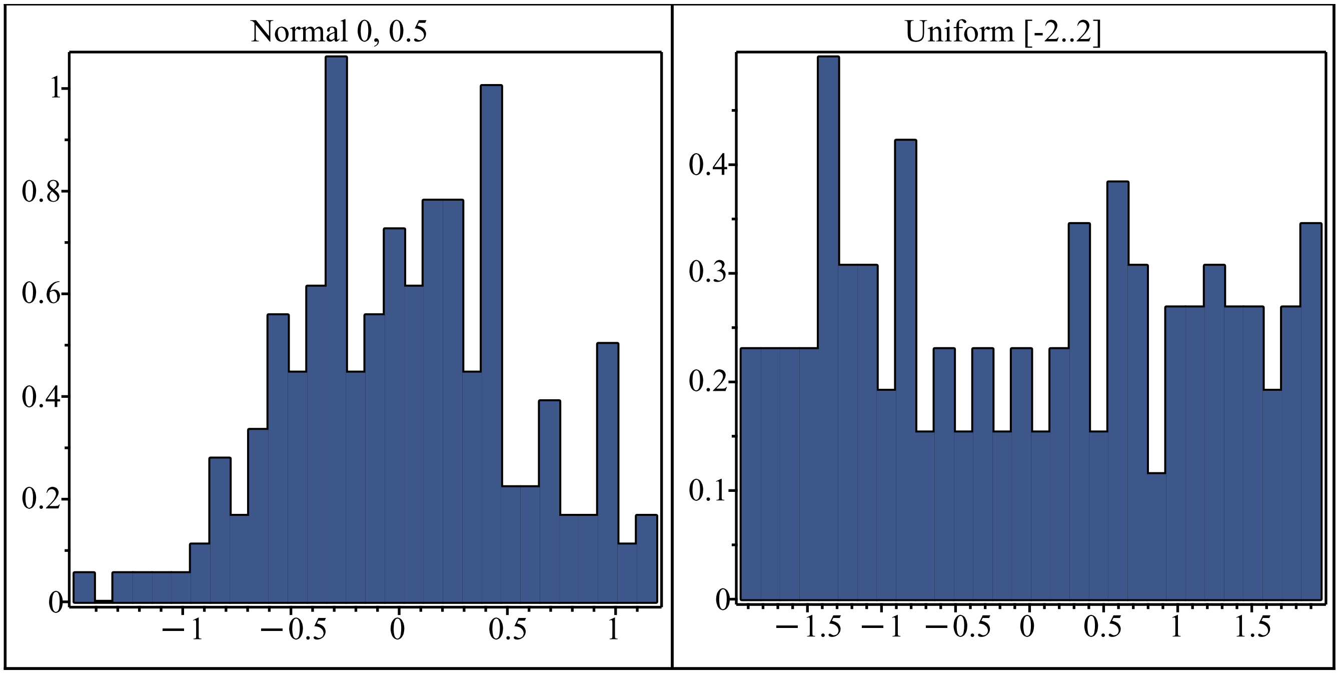
Gives us a real number uniformly from (0,1)

What if we don't want a uniform distribution?

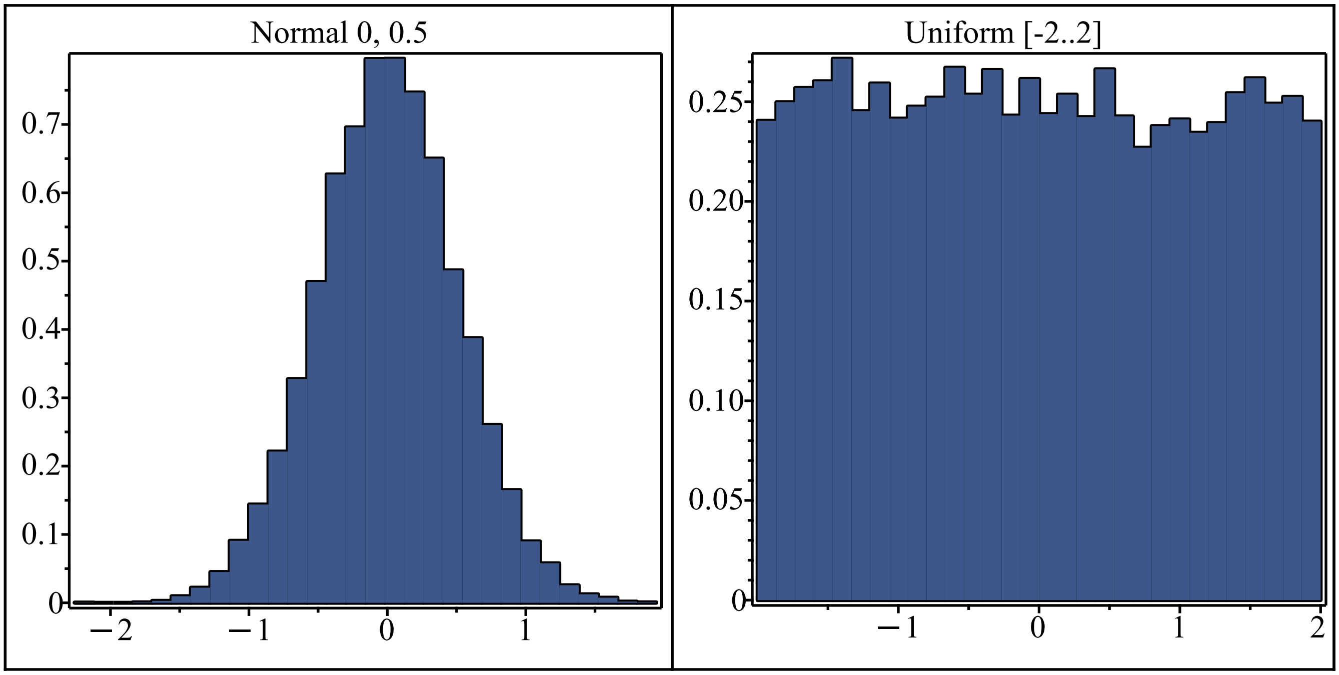
```
> with(Statistics) :
> Sample(Normal(0, 1))(10)
[−0.128256637794691, −0.591149532421989, 0.678191168130654, 0.0188068174304354,     (17)
    −0.866236980876243, 1.57200838858640, −1.38404796272528, −1.15330197268967,
    −0.585735558055269, 0.897078787885014]
> Sample(Normal(5, 10))(10)
[8.58900926672387, 24.8669272182964, 3.25100912265916, 9.67124353162109,           (18)
    2.49067779516532, 6.62855705778330, 14.2450759433564, 18.4730854151050,
    0.212899412716850, −6.88480502946512]
>  Sample(Uniform(5, 10))(10)
[7.47203790270442, 5.31912190007231, 9.92797478825400, 5.49384135917968,           (19)
    7.20505087648033, 7.82909924006966, 9.27453658218099, 5.57554949754972,
    5.70425965682781, 9.83029989756872]
```

Lets look at a picture of what this means

```
> makesomepics := nsamp→plots[display](⟨
  Histogram(Sample(Normal(0, 0.5))(nsamp), title = "Normal 0, 0.5") |
  Histogram(Sample(Uniform(−2, 2))(nsamp), title = "Uniform [-2..2]")⟩)
makesomepics := nsamp ↦ plots[display](⟨Histogram(Sample(Normal(0, 0.5))(nsamp), title    (20)
    = "Normal 0, 0.5")|Histogram(Sample(Uniform(−2, 2))(nsamp), title
    = "Uniform [-2..2]")⟩)
> makesomepics(200)
```



```
> makesomepics(20000)
```



Enuf about that.

Grab some data from class web page   at daily/extras/bad_line.txt

```
> #
  #  Maple procedure to execute maple code stored on the web
  #
  ExecuteFromWeb :=proc(URL :: string, {printfile :: truefalse := false})
```

```
    local n, m, status, webfile, headers;
    # try to get the URL
    status, webfile, headers := HTTP[Get](URL) :
    if ( HTTP[Code](status) ≠ "OK") then
       error(HTTP[Code](status), URL);
    fi;
    # now interpret the maple on the web page
    n := 0 :
    while (n < length(webfile)) do
      m := n;
      parse(webfile, statement, lastread='n', offset=n);
      if (printfile) then printf("%s", webfile[m + 1 ..n]); fi;
    od:
  end:
```

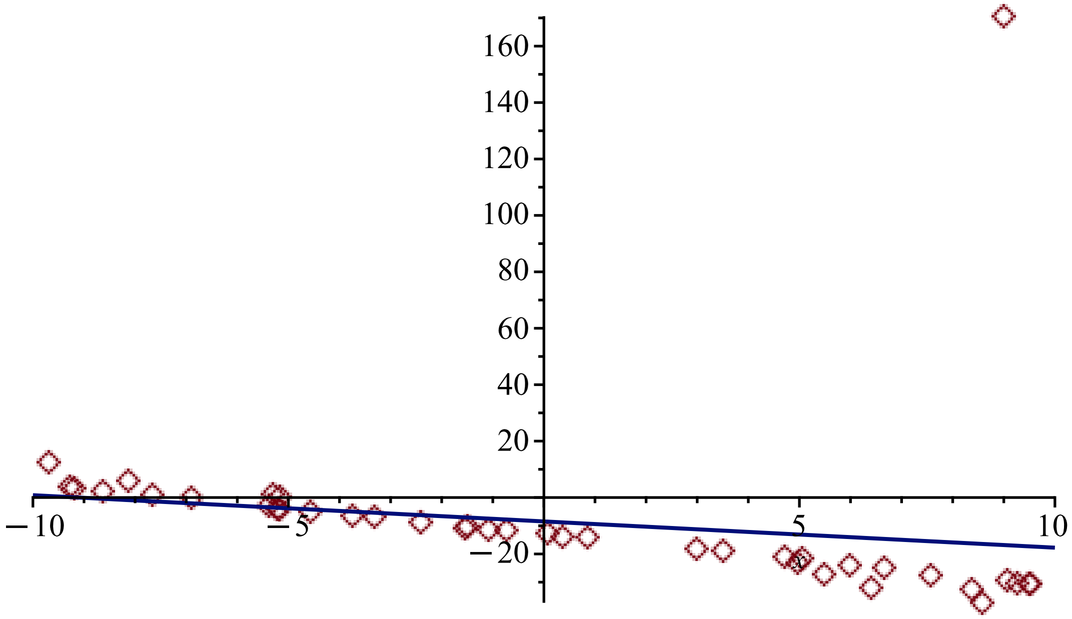
```
> ExecuteFromWeb("https://www.math.stonybrook.edu/~scott/mat331.spr24/daily/extras/bad_line.
      txt");
loaded 40 points into lpts.
```

```
> with(CurveFitting) :
```

```
> bline := LeastSquares(lpts, x);
```

$$bline := -8.47217789980814 - 0.930101594702111\, x \tag{21}$$

```
> plot([lpts, bline], x=-10..10, style=[point, line], symbolsize=20, size=[.8, .6])
```

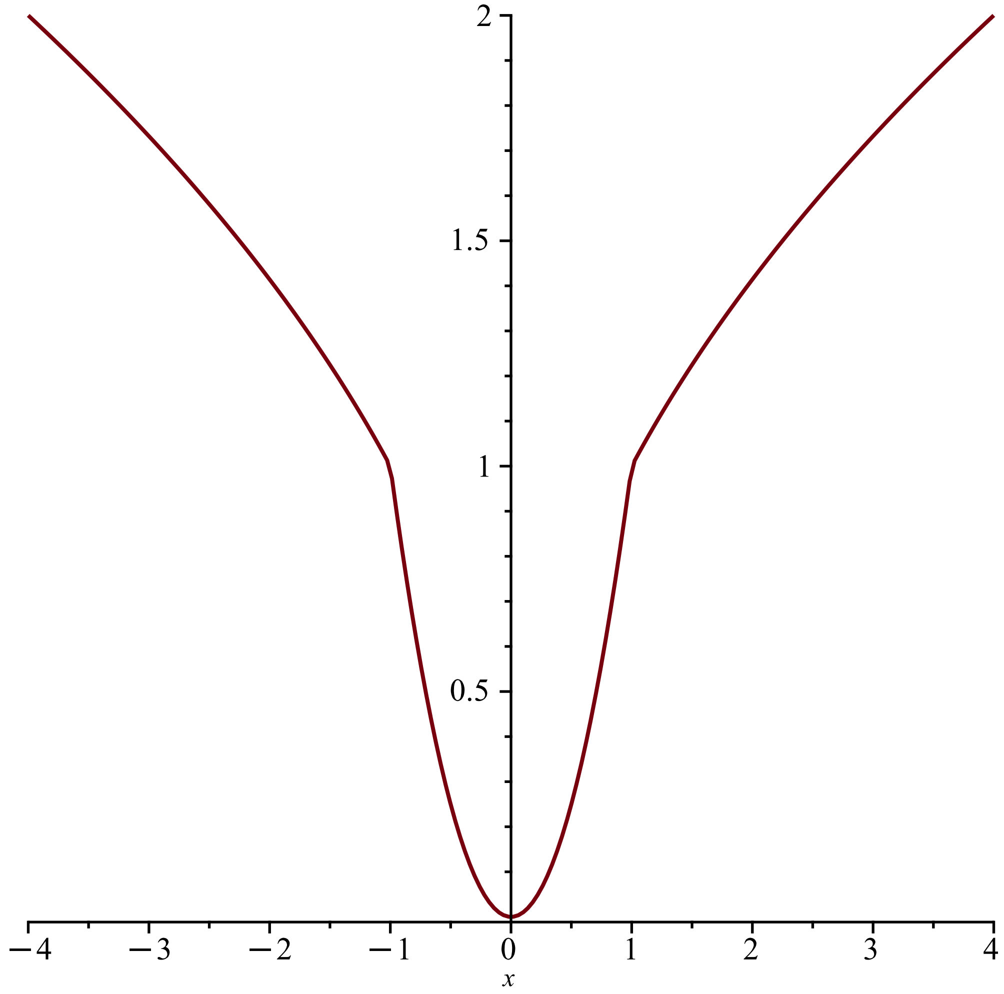


A bad idea, but not too bad

```
> obj := x→piecewise(x < -1, sqrt(-x), x < 1 , x^2, sqrt(x))
```

$$obj := x \mapsto \begin{cases} \sqrt{-x} & x < -1 \\ x^2 & x < 1 \\ \sqrt{x} & otherwise \end{cases} \tag{22}$$

```
> plot(obj(x), x=-4..4)
```


2
1.5
1
0.5
−4
−3
−2
−1
0
1
2
3
4
x

>