(* MATH7502 Prac 1 in Julia with Mathematic for the same. *)

(* For first tutorial. Here are the basic activities to do with Julia:

1. Create a script that approximates the series $\sum_{k=1}^{\infty} \frac{1}{k^2}$. It needs to be close to $\frac{\pi^2}{6}$.

2. Create scripts that plot the distributions,
using histogram of random quantities generated from random matrices:

The random matrices are always of the form
2rand(m,n).-1 (these are always entries between -1 and +1).
Use N = 10^4, 10^5 or 10^6.

For the number of matrices compute:
(a) Determinant of m=2, n=2, m=3, n=3.
(b) sum(inv(A)*A-I) for n = 50, m = 50.
(c) ||u||*||v||- |u^T v|
when u and v are the first two cols of a matrix with m = 4, n =2.

3. Run:https://github.com/h-Klok/StatsWithJuliaBook/blob/master/1_chapter/polyRoots.jl and test it for different polynomials.

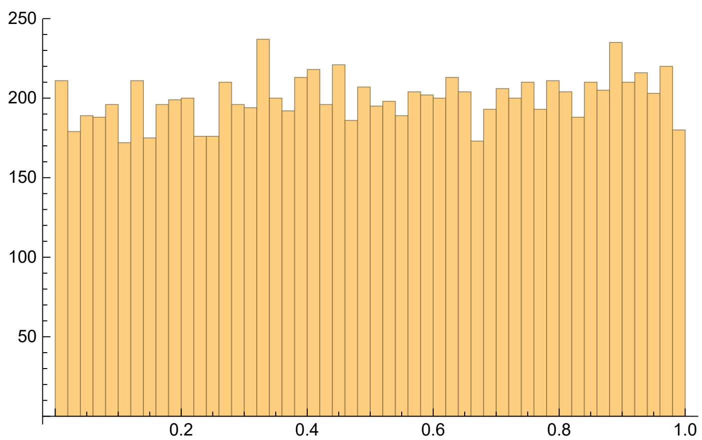
4. Run:https://github.com/h-Klok/StatsWithJuliaBook/blob/master/1_chapter/plotSimple.jl and change the functions plotted *)

In[645]:=
```
(* 1. Create a script that approximates
 the series ∑_{k=1}^∞ 1/k^2. It needs to be close to π^2/6. *)

(* In Mathematica *)

n = 10000;
N[Sum[1/k^2, {k, 1, n}] - Pi^2/6, 20]

(* In Julia: N=10000
sum([1/k^2 for k in 1:N])-π^2/6 #\pi+[TAB] creates π *)
```

Out[646]= -0.000099995000166666666333

In[1218]:=
```
(* 2. Create scripts that plot the distributions,
using histogram of random quantities generated from random matrices:

   The random matrices are always of the form
    2rand(m,n).-1 (these are always entries between -1 and +1).
      Use N = 10^4, 10^5 or 10^6.                                  *)

(* A histogram of 10000 random real numbers in [0, 1] showing 50 rectangles. *)
data = Table[Random[], {j, 1, 10000}];
Histogram[data, 50]

(* In Julia:
  using Plots #for this to work need (first time)
   using Pkg; Pkg.add("Plots")

 histogram(rand(10000),label=false,nbins=50)

*)
```

Out[1219]=



In[1220]:=
```
(* A random n by n matrix with entries in [-1, 1]. *)
q = 2;
A = RandomReal[{-1, 1}, {q, q}];
MatrixForm[A]
(* In Julia:
   m=2
   n=2
   2rand(m,n).-1 #.-subtracts the scalar from the matrix *)
```

Out[1222]//MatrixForm=

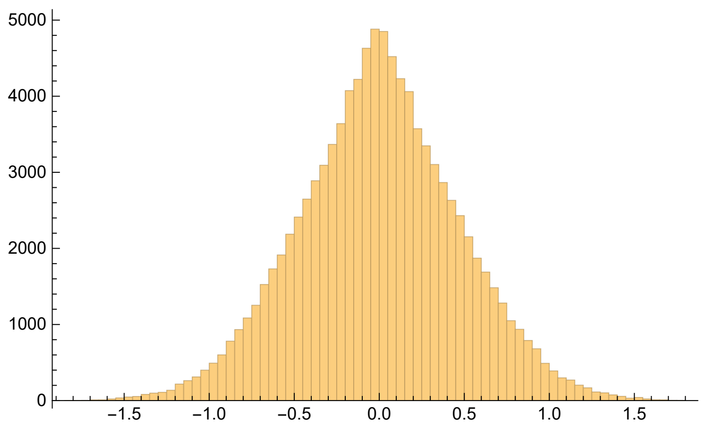
$$\begin{pmatrix} 0.856949 & 0.27695 \\ 0.723718 & 0.658526 \end{pmatrix}$$

In[1223]:=

```
(* A function of n that computes the determinant of a random n by n matrix. *)
detrand[n_] := Module[{X}, X = RandomReal[{-1, 1}, {n, n}]; Det[X]];
detrand[2]
(* In Julia:
   myRandMat()=2rand(m,n).-1

  myRandMat()

  using LinearAlgebra #needed for norm(),det() etc...
det(myRandMat())                                                    *)


(* A list of 100000 determinants of 2 by 2 random matrices. *)
m = 10^5;
data = Table[detrand[2], {j, 1, m}];
Histogram[data, 100]

(* In Julia:
   data=[det(myRandMat()) for _ in 1:N]
histogram(data,label=false,bins=100) *)
```

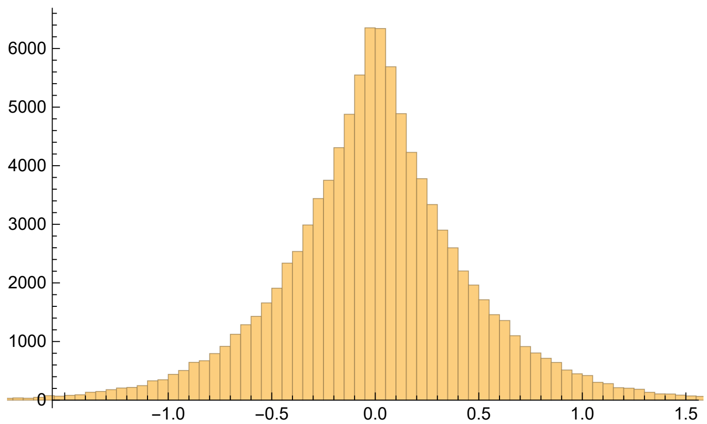
Out[1224]= -0.342177

Out[1227]=



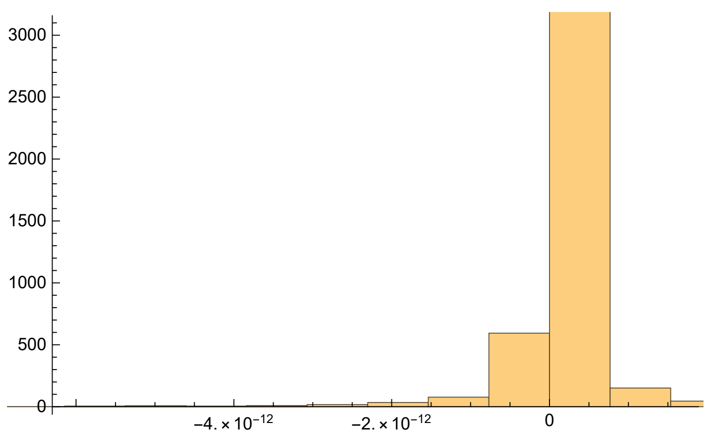
In[1228]:=
```
(* A list of 100000 determinants of 2 by 2 random matrices. *)
m = 10^5;
data = Table[detrand[3], {j, 1, m}];
Histogram[data, 100]

(* In Julia:

   N=10^5
m,n=3,3
data=[det(myRandMat()) for _ in 1:N]
histogram(data,label=false,bins=100)

*)
```

Out[1230]=



In[1231]:=
```
(* A list of 10000 sums of elements of 50 by 50 matrices of the form A.A^(-1)-I,
where the A are random matrices. *)
m = 10^4;
(* A function that performs the task.*)
frobrand[p_] := Module[{},
   A = RandomReal[{-1, 1}, {p, p}];
   B = Inverse[A];
   Z = A.B - IdentityMatrix[p];
   S = Flatten[Z];
   Sum[S[[k]], {k, 1, p^2}]];
(* Construct a list of m such sums. *)
data = Table[frobrand[50], {j, 1, m}];
Histogram[data, PlotRange → 3000]

Max[data]


(* In Julia:

    m,n=50,50
    N=10^4

    data=[]

    for _ in 1:N
        A=myRandMat()
        Ai=inv(A)
        err=sum(A*Ai-I) #Frobenious norm is sum of elements of matrix
        push!(data,err)
    end

     sort(data)

     maximum(data)

 *)
```

Out[1234]=

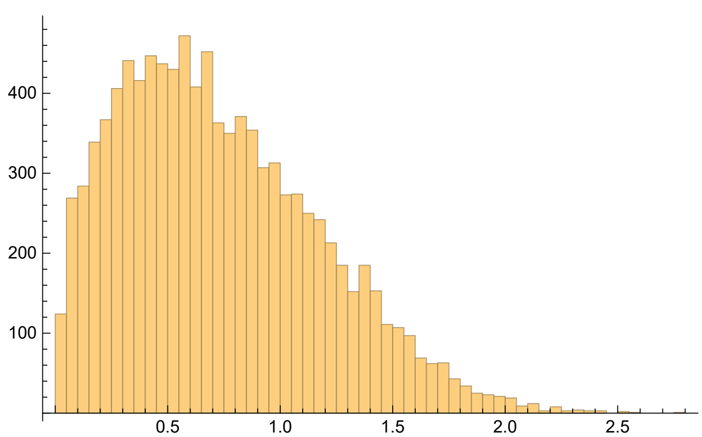


Out[1235]= 5.65832×10^{-11}

In[1236]:=

```
(* 2. (c) Create scripts that plot the distributions,
using a histogram of random quantities generated from:

    ||u||*||v|| - |u^Tv| , should be positive.

   when u and v are the first two cols of a matrix with m = 4, n = 2. *)

cauchy[m_, n_] := Module[{A, u, v},
   A = RandomReal[{-1, 1}, {m, n}];
   u = Partition[A[[All, 1]], 1];
   v = Partition[A[[All, 2]], 1];
   (Norm[u] Norm[v] - Abs[Transpose[u].v])[[1, 1]]];

data = Table[cauchy[4, 2], {j, 1, 10^4}];

Histogram[data, 50]


(* In Julia:

  m,n=4,2
  N=10^4
  data=[]
  for _ in 1:N
      A=myRandMat()
      u=A[:,1]
      v=A[:,2]
      err=norm(u)*norm(v)-abs(u'*v) #Cauchy Schwartz says this is positive
      push!(data,err)
  end

         histogram(data,legend=false)

*)
```

Out[1238]=



In[1239]:=
```
(* 3. Run:https://
 github.com/h-Klok/StatsWithJuliaBook/blob/master/1_chapter/polyRoots.jl
   and test it for different polynomials.   *)

a = Table[RandomReal[{-1, 1}], {i, 1, 20}];
n = Length[a] - 1;
f = Sum[a[[i + 1]] x^i, {i, 0, n}];
NSolve[f == 0, x]

(* In Julia:

  #maybe need using Pkg;Pkg.add("Roots")

   using Roots #around page 20 here:https://
  statisticswithjulia.org/StatisticsWithJuliaDRAFT.pdf

function polynomialGenerator(a...)
    n = length(a)-1
    poly = function(x)
               return sum([a[i+1]*x^i for i in 0:n])
           end
    return poly
end

polynomial = polynomialGenerator(1,3,-10)
zeroVals = find_zeros(polynomial,-10,10)
println("Zeros of the function f(x): ",zeroVals)

*)
```

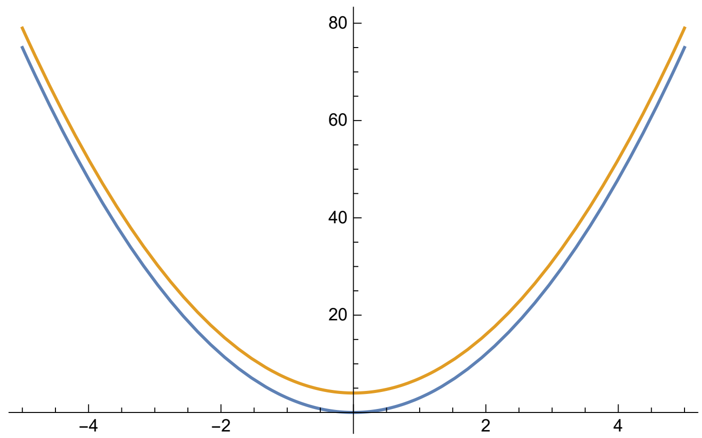
Out[1242]=
```
{{x → -2.02535}, {x → -0.974944}, {x → -0.801705 - 0.562139 i}, {x → -0.801705 + 0.562139 i},
 {x → -0.66721}, {x → -0.577126 - 1.65953 i}, {x → -0.577126 + 1.65953 i},
 {x → -0.343508 - 0.900754 i}, {x → -0.343508 + 0.900754 i}, {x → -0.0576463},
 {x → 0.0642909 - 1.07174 i}, {x → 0.0642909 + 1.07174 i}, {x → 0.465857 - 0.647855 i},
 {x → 0.465857 + 0.647855 i}, {x → 0.721741 - 0.676303 i}, {x → 0.721741 + 0.676303 i},
 {x → 0.871581}, {x → 1.0786 - 0.473521 i}, {x → 1.0786 + 0.473521 i}}
```

In[1243]:=
```
(* 4. Run:https://github.com/h-Klok/StatsWithJuliaBook/blob/master/1
   _chapter/plotSimple.jl and change the functions plotted *)
ClearAll[f]
f[x_, y_] := 3 x^2 + y^2;
f0 = f[x, 0]
f2 = f[x, 2]
Plot[{f0, f2}, {x, -5, 5}]

Plot3D[f[x, y], {x, -5, 5}, {y, -5, 5}]

W = Table[-1 + 0.1 Norm[{6 x, 2 y}], {x, -5, 5, 0.1}, {y, -5, 5, 0.1}];
Image[W, ColorSpace → "Graylevel"]

(* In Julia:

  using Plots,LaTeXStrings,Measures;
pyplot() #around page 26 here:https://
  statisticswithjulia.org/StatisticsWithJuliaDRAFT.pdf

   f(x,y)=3x^2+y^2
   f0(x)=f(x,0)
   f2(x)=f(x,2)

   xVals,yVals=-5:0.1:5,-5:0.1:5
   plot(xVals,[f0.(xVals),f2.(xVals)],
           c=[:blue:red],xlims=(-5,5), legend=:top,
           ylims=(-5,25),ylabel=L"f(x,\cdot)", label=[L"f(x,0)" L"f(x,2)"])
   p1=annotate!(0,-0.2,text("(0,0) The minimum\n of f(x,0)", :left, :top,10))

   z=[f(x,y) for y in yVals,x in xVals]
   p2=surface(xVals,yVals,z,c=cgrad([:blue, :red]), legend=:none,
         ylabel="y",zlabel=L"f(x,y)")

     M = z[1:10,1:10]
   p3 = heatmap(M,c=cgrad([:blue, :red]),yflip=true,
        ylabel="y",xticks=([1:10;],xVals),yticks=([1:10;],yVals))

   plot(p1,p2,p3,layout=(1,3),size=(1200,400),xlabel="x",margin=5mm)

*)
```

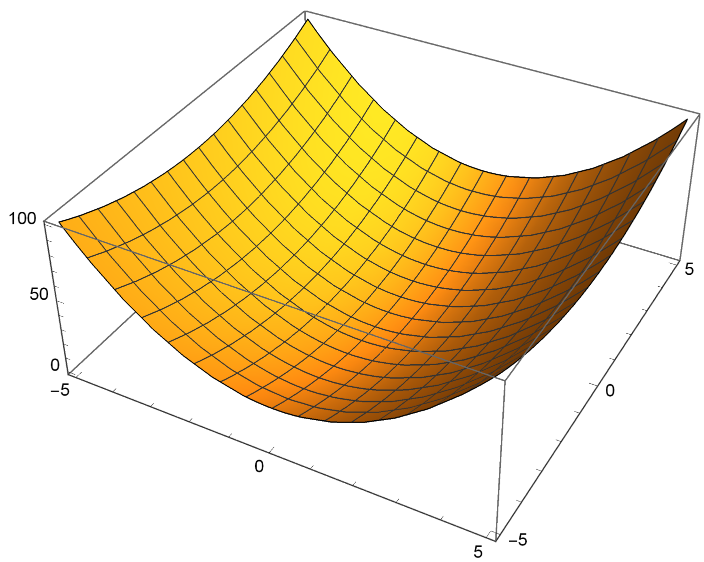
Out[1245]= $3 x^2$

Out[1246]= $4 + 3 x^2$

Out[1247]=



Out[1248]=



Out[1250]=