Example 1 of Maximum Likelihood and Tools for LR Confidence Sets

This is the really simple example:

- Linear regression
- Gaussian errors
- Two parameters
- Elliptical confidence sets

Initial settings

In[120]:=

```
SetOptions[ListPlot, AspectRatio -> Automatic];
SetOptions[RegionPlot, AspectRatio -> Automatic];
SetOptions[ListPlot3D, AspectRatio -> 1];
SetOptions[ContourPlot, AspectRatio -> Automatic];
SetOptions[ContourPlot3D, AspectRatio -> 1];
SeedRandom[Method → "MersenneTwister"];
SetOptions[FindMaximum, AccuracyGoal -> 5, PrecisionGoal -> 5];
```

Example 1 - trivial linear regression - one variable and two parameters

Choose the distribution for the errors and let RV be that distribution

In[126]:=
```
dist = NormalDistribution[0, 1];
RV := RandomVariate[dist, WorkingPrecision → 32]
```

Define the likelihood function (likf) (which is the density) and its log (loglikf)

In[128]:=
```
likf = PDF[dist, x]
```

Out[128]=
$$\frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}$$

In[129]:=
```
loglikf = Log[%] // PowerExpand
```

Out[129]=
$$-\frac{x^2}{2} + \frac{1}{2}(-\text{Log}[2] - \text{Log}[\pi])$$

Log likelihood is quadratic when errors are Gaussian

We set the seed so that we can replicate experiments

In[130]:=
```
SeedRandom[0];
```

Model specification and data

Number of data points.

I choose a small number so that one can see each step

In[131]:=
```
numdatapoints = 20;
```

Model:

In[132]:=
```
params = {a, b};
numparams = Length[params];
vars = {x};
model[x_] = a + b x ;
```

True parameter values

We choose a degenerate true model

In[136]:=
```
truparams = Thread[params -> 0];
model[x] /. truparams
```

Out[137]=
```
0
```

Generate data

Choose a random collection of x values, uniform on [0, 1]

In[138]:=
```
data = Table[RandomReal[{0, 1}, WorkingPrecision → 32], {numdatapoints}];
```

Compute true values of model[x] at the data points

In[139]:=
```
truth = model /@ data /. truparams;
```

obs is the vector of observations. Each observation is the truth plus noise.

In[140]:=
```
noise = Table[RV, {numdatapoints}];
obs = truth + noise;
```

Compute maximum likelihood

I now compute the maximum likelihood estimate assuming that the log likelihood function is quadratic in the errors (noise is Gaussian with known mean and variance).

noisehat is the vector of errors given parameters (a, b):

In[142]:=
```
noisehat = obs - model /@ data;
```

Log likelihood function is

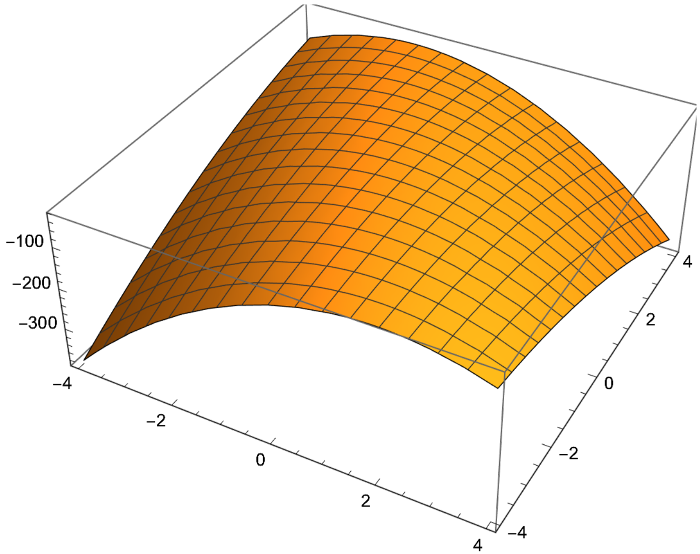
In[143]:=
```
loglik = Sum[loglikf /. x -> pt, {pt, noisehat}] // Expand;  loglik // N
```

Out[143]= $-26.8272 + 3.19175\, a - 10.\, a^2 + 0.763693\, b - 8.89078\, a\, b - 2.64901\, b^2$

Define likfcn, the likelihood of the data give (a, b) parameter values:

```
In[144]:= likfcn[a_, b_] = loglik;  Plot3D[loglik, {a, -4, 4}, {b, -4, 4}]
```

Out[144]=



Find and record the maximum likelihood estimate:

```
In[145]:= estimate = FindMaximum[loglik, {a, b}, WorkingPrecision → 32] // N;
maxlik = estimate[[1]]
maxpt = {a, b} /. estimate[[2]]
```

```
Out[146]= -26.413

Out[147]= {0.376012, -0.486851}
```

Find max by search

Choose range

In[148]:=
```
amin = -4; amax = 4; bmin = -4; bmax = 4;
```

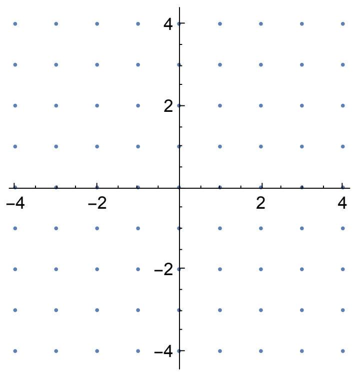
Grid search

In[149]:=
```
range = Range[-4, 4]
```

Out[149]=
```
{-4, -3, -2, -1, 0, 1, 2, 3, 4}
```

In[150]:=
```
pts = Flatten[Outer[List, range, range], 1];
ListPlot[pts]
```

Out[151]=



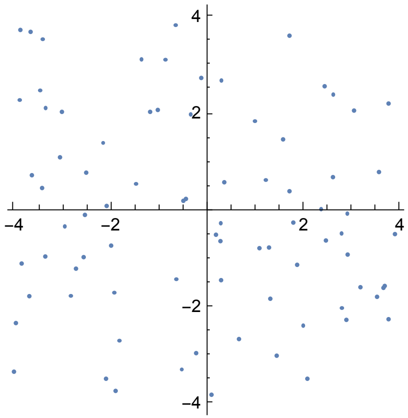
In[152]:=
```
likfcn @@@ pts // Max
```

Out[152]=
```
-26.827202083824753609218272047628
```

MC search

In[153]:=
```
pts = Table[RandomReal[{-4, 4}, 2], {81}];
ListPlot[pts]
likfcn @@@ pts // Max
```

Out[154]=



Out[155]= -26.4401

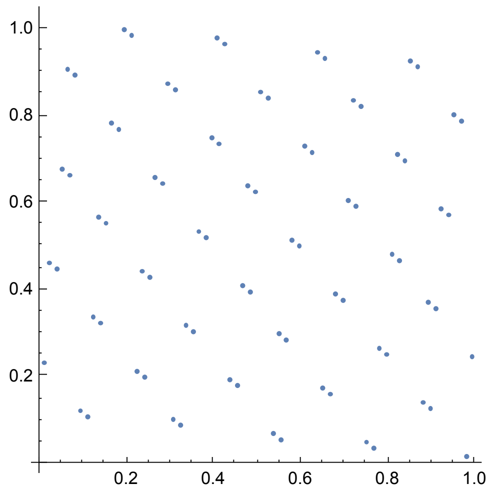
qMC search

Weyl Sequences

In[104]:=
```
num = 81; list = Table[i, {i, 1, num}]; dim1 = list 2^.5;
dim1 = dim1 - Floor[dim1]; dim2 = list 3^.5; dim2 = dim2 - Floor[dim2];
weyl = Table[{dim1[[i]], dim2[[i]]}, {i, 1, num}];
Print[num, " Weyl points"];
ListPlot[weyl, AxesOrigin → {0, 0}]
```

```
81 Weyl points
```

Out[108]=



In[156]:=
```
likfcn @@@ weyl // Max
```

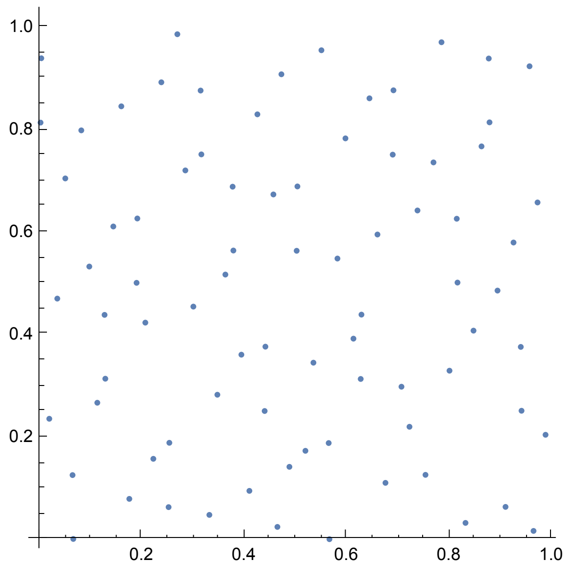
Out[156]=
```
-26.6488
```

Sobol Sequences

```
SeedRandom[Method -> {"MKL", Method -> {"Niederreiter", "Dimension" -> 2}}];
pts = RandomReal[1, {2000, 2}];
```

In[119]:=
```
Do[Print[k, " Sobol points"];
ListPlot[pts[[1 ;; k]]] // Print, {k, {81}}]
```

```
81 Sobol points
```



In[158]:=
```
likfcn @@@ pts[[1 ;; 81]] // Max
```

Out[158]=
```
-26.4401
```

Nelder-Mead (fminsearch)

Not parallelizable except by running multiple versions with different initial guesses

Genetic algorithm

See Goffe-Ferrier-Rogers (1994) for a nice evaluation of simulated annealing versus conventional methods.

Time for a new comparison.