Example 1: ML and LR Confidence Sets

```
In[439]:= x = 0; ClearAll["Global`*"]; DateList[Date[]] // Most
          SetDirectory[NotebookDirectory[]];

Out[439]= {2022, 3, 17, 16, 6}
```

Initial settings

Choose errors

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

```
In[452]:= 𝒟 = MixtureDistribution[{1, 1},
            {NormalDistribution[-1, 1], NormalDistribution[1, 1]}];
          dist = 𝒟;
          RV := RandomVariate[dist, WorkingPrecision → 32]
```

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

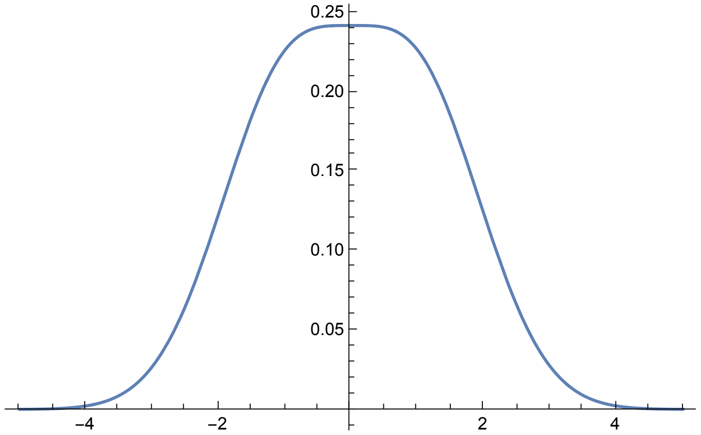
```
In[455]:= likf = PDF[dist, x]
```

Out[455]=

$$\frac{e^{-\frac{1}{2}(-1+x)^2}}{2\sqrt{2\pi}}+\frac{e^{-\frac{1}{2}(1+x)^2}}{2\sqrt{2\pi}}$$

In[456]:=
```
Plot[likf, {x, -5, 5}, PlotRange → All]
```

Out[456]=



In[457]:=
```
loglikf[x_] = Log[likf] // PowerExpand
```

Out[457]=

$$\mathrm{Log}\left[\frac{e^{-\frac{1}{2}(-1+x)^2}}{2\sqrt{2\pi}}+\frac{e^{-\frac{1}{2}(1+x)^2}}{2\sqrt{2\pi}}\right]$$

We set the seed so that we can replicate experiments

In[458]:=
```
SeedRandom[0];
```

Model specification and data

This is the really simple example:

Linear regression

Gaussian errors

Two parameters

Elliptical confidence sets

Number of data points.

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

```
In[459]:= numdatapoints = 200;
```

Model:

```
In[460]:= params = {a, b};
          numparams = Length[params];
          Clear[model]
          vars = {x};
          model[x_] = a + b x ;
```

True parameters

We choose a degenerate true model

```
In[465]:= truparams = Thread[params -> 0];
```

Generate data

Choose a random collection of x values

```
In[466]:= xdata = Table[RandomReal[{0, 1}, WorkingPrecision → 32], {numdatapoints}];
```

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

```
In[467]:= truth = model /@ xdata /. truparams;
```

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

```
In[468]:= noise = RandomVariate[𝒟, numdatapoints];
```

```
In[469]:= noise = Table[RV, {numdatapoints}];
          obs = truth + noise;
```

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 errors assuming that the parameters are (a, b):

```
In[471]:= errors[a_, b_] = obs - model /@ xdata;
```

Compute maximum likelihood

```
In[472]:= loglik[a_, b_] := (
             For[
              i = 1; sum = 0,
              i < numdatapoints,
              i++,
              sum = sum + loglikf[ errors[a, b][[i]] ]
             ];
             sum
           )

In[473]:= loglik[0, 0]

Out[473]= -365.76014200995332513461800863337
```

Find and record the maximum likelihood estimate:

```
In[474]:= Clear[a, b]

In[475]:= estimate = FindMaximum[loglik[a, b], {a, b}, WorkingPrecision → 32] // N;
          maxlik = estimate[[1]]
          maxpt = {a, b} /. estimate[[2]]

Out[476]= -365.452

Out[477]= {-0.155331, 0.218139}
```

Evaluation for random searches

If we have BestSoFar, and we want to avoid unnecessary additions, the Log likelihood function can be computed by loglikS:

In[478]:=
```
loglikS[a_, b_] := (
   For[
    i = 1; sum = 0,
    sum > BestSoFar && i < numdatapoints,
    i++,
    sum = sum + loglikf[ errors[a, b][[i]] ]
   ];
   If[sum > BestSoFar, BestSoFar = sum; θhat = {a, b}, Nothing];
   {i, sum}
  )
```

Choose number of points to evaluate likelihood:

In[479]:=
```
numpoints = 100;
```

Search in a small region close to solution

Define range of parameter values to check

```
In[480]:= amin = bmin = -1; amax = bmax = 1;

In[481]:= BestSoFar = -10 000 000;
          tab = Table[
             loglikS[RandomReal[{amin, amax}], RandomReal[{bmin, bmax}]],
             {numpoints}];
          Print["Number of potential summands:  ", Length[tab] numdatapoints]
          Print["Number of actual summands:  ", Plus @@ tab[[All, 1]]]

          Number of potential summands:  20 000

          Number of actual summands:  18 885

In[485]:= BestSoFar

Out[485]= -365.553

In[486]:= θhat

Out[486]= {-0.0669963, 0.0723332}
```

No gain by using a running check on summation

Search in a large region containing solution

Define range of parameter values to check

```
In[487]:= amin = bmin = -10; amax = bmax = 10;
```

```
In[488]:= BestSoFar = -10 000 000;
          tab = Table[
             logLikS[RandomReal[{amin, amax}], RandomReal[{bmin, bmax}]],
             {numpoints}];
          Print["Number of potential summands:  ", Length[tab] numdatapoints]
          Print["Number of actual summands:  ", Plus @@ tab[[All, 1]]]

          Number of potential summands:  20 000

          Number of actual summands:  5400
```

```
In[492]:= BestSoFar

Out[492]= -375.108
```

```
In[493]:= θhat

Out[493]= {-0.829305, 0.797107}
```

75% reduction in computational cost by using a running check on summation

Search in an even larger region containing solution

Define range of parameter values to check

```
In[494]:= amin = bmin = -50; amax = bmax = 50;

In[513]:= BestSoFar = -1 000 000;
          tab = Table[
             loglikS[RandomReal[{amin, amax}], RandomReal[{bmin, bmax}]] // Quiet,
             {numpoints}];
          Print["Number of potential summands:  ", Length[tab] numdatapoints]
          Print["Number of actual summands:  ", Plus @@ tab[[All, 1]]]

          Number of potential summands:  20 000

          Number of actual summands:  3513

In[499]:= BestSoFar

Out[499]= -375.641

In[500]:= θhat

Out[500]= {-0.780771, 1.7023}
```

85% reduction in computational cost by using a running check on summation

PROJECT: Check this out for

- higher-dimensional regressions
- more interesting likelihood functions