Example 1: ML and LR Confidence Sets

This is the really simple example:

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

Initial settings

Example 1 - one variable and two parameters

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

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

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

```
In[1915]:= likf = PDF[dist, x]
```

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

```
In[1916]:= loglikf = Log[%] // PowerExpand
```

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

Log likelihood is quadratic when errors are Gaussian

We set the seed so that we can replicate experiments

```
In[1917]:= SeedRandom[0];
```

Model specification and data

Number of data points.

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

```
In[1918]:= numdatapoints = 20;
```

Model:

```
In[1919]:= params = {a, b};
           numparams = Length[params];
           vars = {x};
           model[x_] = a + b x ;
```

True parameter values

We choose a degenerate true model

```
In[1923]:= truparams = Thread[params -> 0];
           model[x] /. truparams

Out[ ]= 0
```

Generate data

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

```
In[1925]:= data = Table[RandomReal[{0, 1}, WorkingPrecision → 32], {numdatapoints}];
```

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

```
In[1926]:= truth = model /@ data /. truparams;
```

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

```
In[1927]:= 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[1929]:= noisehat = obs - model /@ data;
```

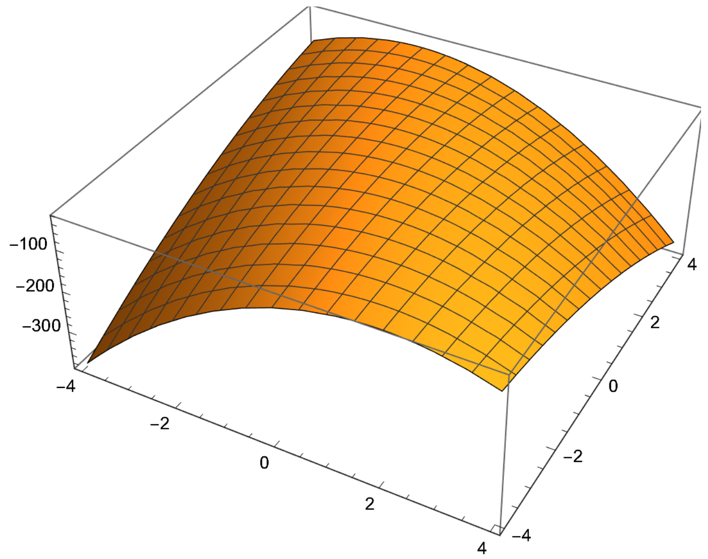
Log likelihood function is

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

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

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

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

Out[•]=


Find and record the maximum likelihood estimate:

```
In[1932]:= estimate = FindMaximum[loglik, {a, b}, WorkingPrecision → 32] // N;
           maxlik = estimate[[1]]
           maxpt = {a, b} /. estimate[[2]]

Out[∘]=    -26.413

Out[∘]=    {0.376012, -0.486851}
```

Plot the estimate for future graphs

```
In[∘]:=    plotmaxpt = ListPlot[{maxpt}, PlotStyle → {Black}];
```

Two-parameter confidence sets

We choose levels for the likelihood function corresponding to the 90%, 95%, and 99% confidence sets

```
In[•]:= levels = {level90, level95, level99} = maxlik - chisquare[[numparams]]
Out[•]= {-31.018, -32.404, -35.623}
```

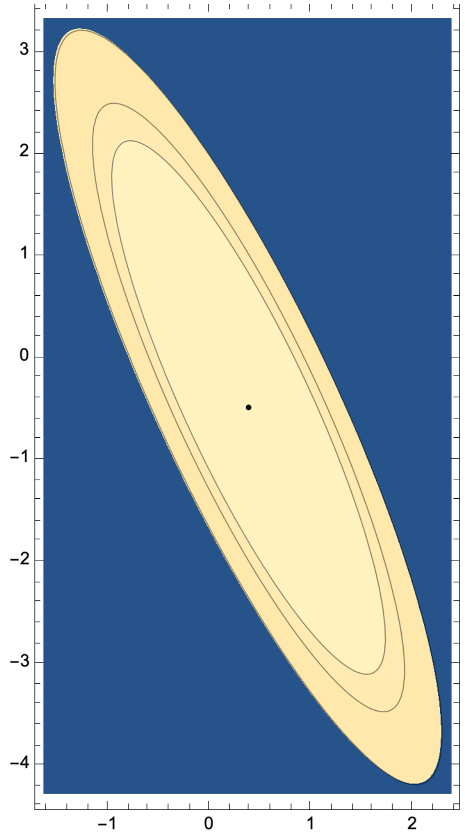
Contour Plots

We first define a box containing all relevant points, that is, all points where the likelihood exceeds the 99% confidence set.

```
In[•]:= amin = FindMinimum[{a, loglik >= level99}, {a, b}][[1]] - 0.1
Out[•]= -1.62817

In[•]:= bmin = FindMinimum[{b, loglik >= level99}, {a, b}][[1]] - 0.1
Out[•]= -4.28656

In[•]:= amax = FindMaximum[{a, loglik >= level99}, {a, b}][[1]] + 0.1
Out[•]= 2.38021

In[•]:= bmax = FindMaximum[{b, loglik >= level99}, {a, b}][[1]] + 0.1
Out[•]= 3.31289
```

We plot contours of the likelihood corresponding to the 90%, 95%, and 99% confidence levels. They are centered on the estimate (maxpt).

In[∘]:=
```
contours = ContourPlot[loglik, {a, amin, amax}, {b, bmin, bmax}, Contours -> levels];
Show[contours, plotmaxpt]
```

Out[∘]=


Grid evaluation

I compute the likelihood function on a grid of points.
I choose those points where the likelihood function exceeds some critical value, and plot that 2D set of points.

Here are some options I set for ListPlot and RegionPlot

```
In[•]:= SetOptions[ListPlot, PlotRange -> {{amin, amax}, {bmin, bmax}}];
        SetOptions[ListPlot, PlotStyle -> Red];
        SetOptions[ListPlot, AspectRatio -> Automatic];
        SetOptions[RegionPlot, AspectRatio -> Automatic];
```

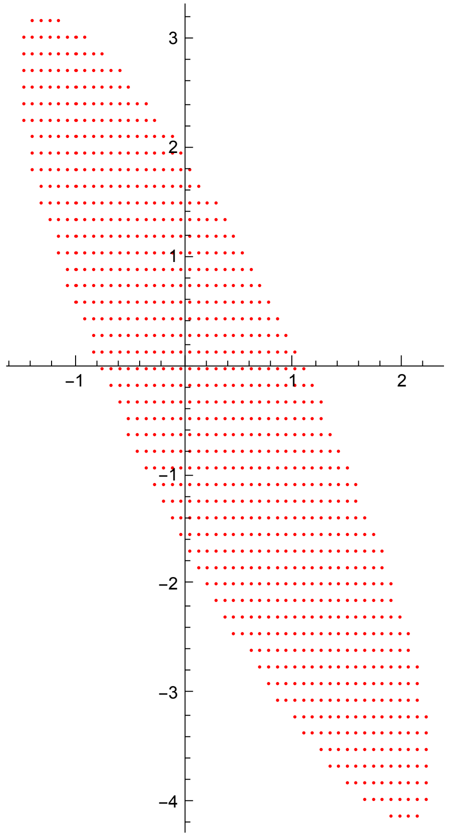
Construct grid:

```
        (* step is the distance between consecutive points *)
        astep = (amax - amin) / 50; bstep = (bmax - bmin) / 50;
        (* define a grid of width 2 x range *)
        grida = Range[amin, amax, astep]; gridb = Range[bmin, bmax, bstep];
        (* Construct the tensor product of the grids *)
        gridpts = Flatten[Outer[List, grida, gridb], 1];
        (* Evaluate the likelihood function at each grid point*)
        gridvals = Map[likfcn @@ # &, gridpts];
        (* Save each grid point along with its likelihood value*)
        griddata = {gridpts, gridvals} // Transpose;
        (* Each entry in griddata is a grid point along with its likelihood value *)
        griddata[[1]]
Out[•]= {{-1.62817, -4.28656}, -172.532}
```

Choose the points in gridpts that have likelihood greater than level90 (level95, level99)

```
In[•]:= data99 = First /@ (Select[griddata, #[[2]] > level99 &]);
        mesh99 = ListPlot[data99, AspectRatio -> Automatic]
```

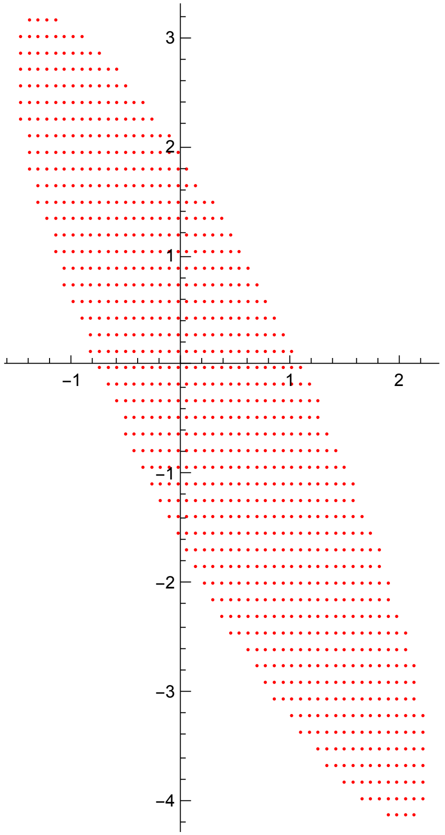
Out[•]=



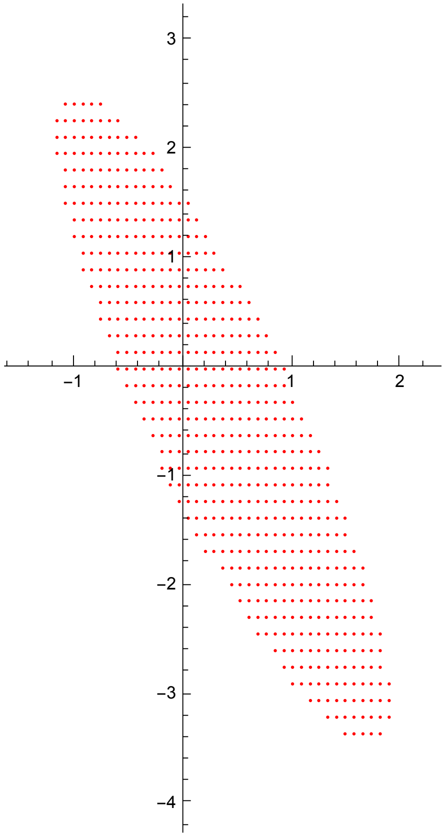
```
In[•]:= data95 = First /@ (Select[griddata, #[[2]] > level95 &]);
        mesh95 = ListPlot[data95];

        data90 = First /@ (Select[griddata, #[[2]] > level90 &]);
```

```
In[•]:= {Show[mesh99, ImageSize -> Medium],
         Show[mesh95, ImageSize -> Medium], Show[mesh90, ImageSize -> Medium]}
```

```
Out[•]= {
```

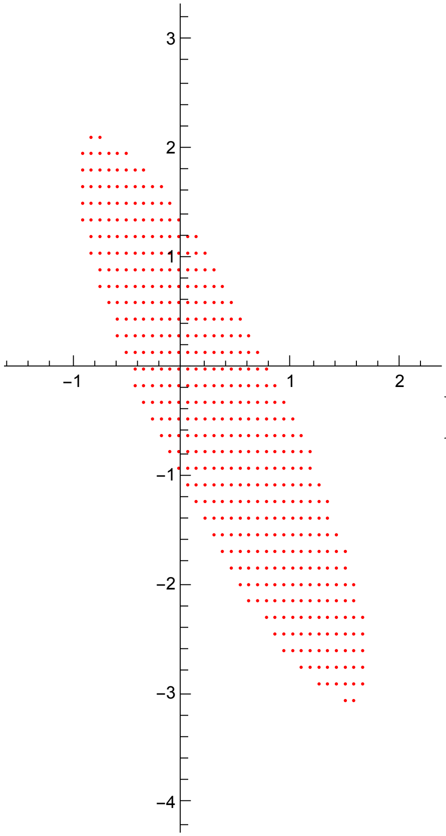


,



3
2
1
–1
–2
–3
–4
–1
1
2

,



3
2
1
–1
–2
–3
–4
–1
1
2

}

Dimension-by-dimension solutions for boundary points

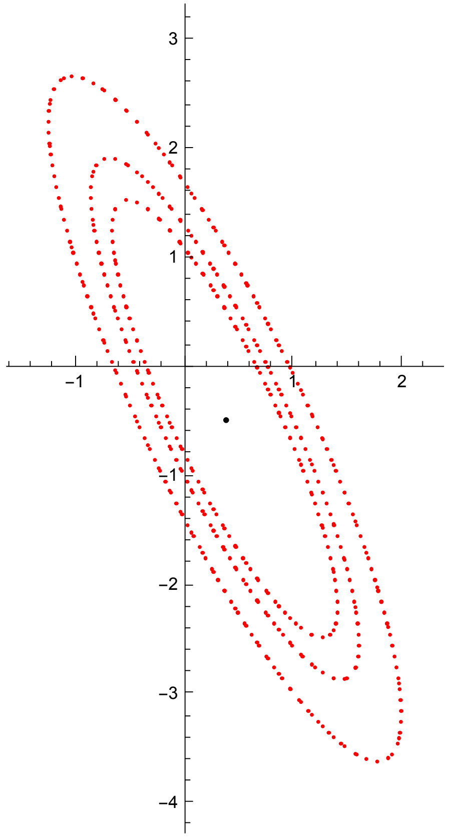
Find values for (a, b) such that likfcn[a, b] = level.
We construct 1D equations: first fix b and solve for a, and second fix a and solve for b.
The command NSolve finds all solutions (possible here because the function is a polynomial), and then we keep only the real solutions. In these examples, we only got two solutions but approach will work even if there are multiple real solutions.
We then merge the two sets of boundary points.

```
In[•]:= step = 10 / 100;
```

```
In[•]:= Bndry :=
  ((* Choose b from {-range, range, step} and solve
    for a such that the loglikelihood is level90 *)tab = Table[
     Outer[List, a /. NSolve[likfcn[a, b] == level, a], {b}], {b, bmin, bmax, step}];
 list1 = Flatten[tab, 2];
 list1 = Select[list1, Element[#[[1]], Reals] && Element[#[[2]], Reals] &];
 plot1 = ListPlot[list1];
 (* Choose a from {-range, range, step} and solve
    for b such that the loglikelihood is level90 *)tab = Table[
     Outer[List, {a}, b /. NSolve[likfcn[a, b] == level, b]], {a, amin, amax, step}];
 list2 = Flatten[tab, 2];
 list2 = Select[list2, Element[#[[1]], Reals] && Element[#[[2]], Reals] &];
 plot2 = ListPlot[list2];
 (* Combine the two lists of points and plot*)
   list = Union[list1, list2];
   ListPlot[list])
```

```
In[•]:= level = level90;  bndry90 = Bndry;
        level = level95;  bndry95 = Bndry;
        level = level99;  bndry99 = Bndry;

In[•]:= Show[bndry90, bndry95, bndry99, plotmaxpt]
```

Out[•]=



Rays centered at the estimate

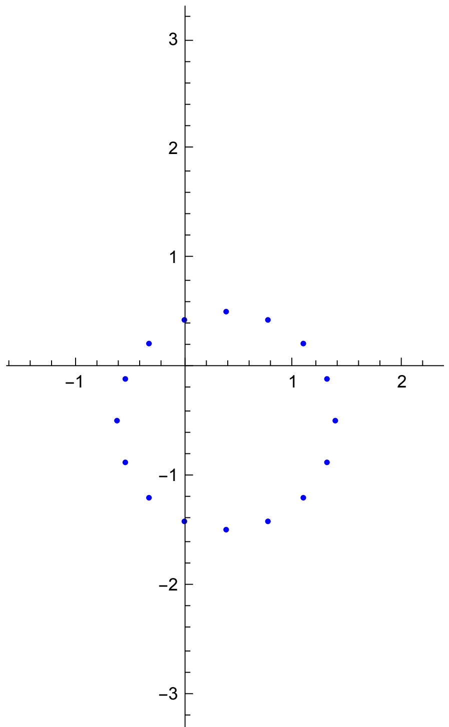
In this case, we find boundary values along rays originating at the ML. In this case, we define the rays to be uniformly distributed on a circle.

Simple example

Here is a simple example with only 16 points

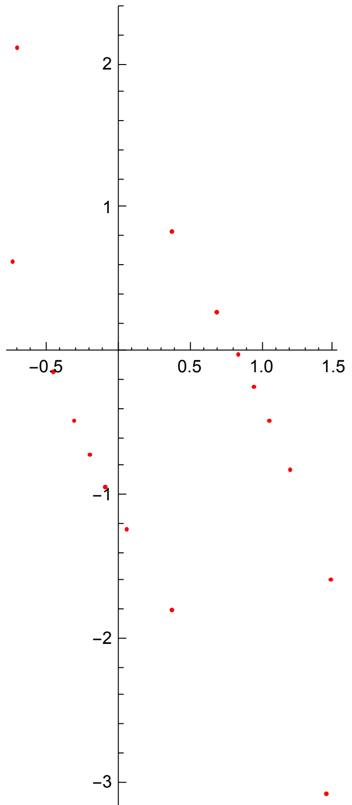
```
In[•]:= rays = Table[maxpt + {Sin[θ], Cos[θ]}, {θ, -Pi , Pi, Pi / 8}] // Union;
listrays = ListPlot[rays, PlotStyle → Blue]
```

Out[•]=



```
In[•]:= ptλ := λ maxpt + (1 - λ) ray

In[•]:= tab = Table[
            λs = λ /. NSolve[likfcn @@ ptλ == level90, λ];
            pts = Table[λs[[j]] maxpt + (1 - λs[[j]]) ray, {j, λs // Length}],
            {ray, rays}];
        list = Flatten[tab, 1];
        list = Cases[list, {__Real}];
        ray90 = ListPlot[list, PlotRange → All]
```

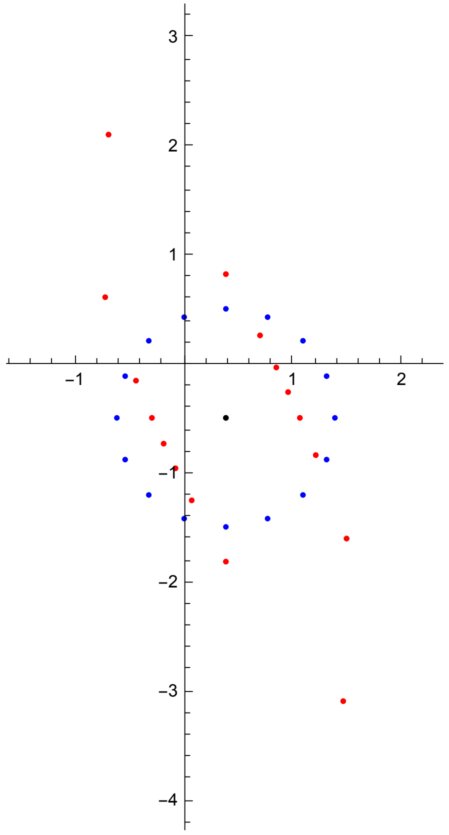
Out[•]=



This graph illustrates that each red dot is on the line connecting the black dot (the ML estiamate) with a blue dot.

In[ ]:=
```
Show[listrays, ray90, plotmaxpt]
```

Out[ ]=

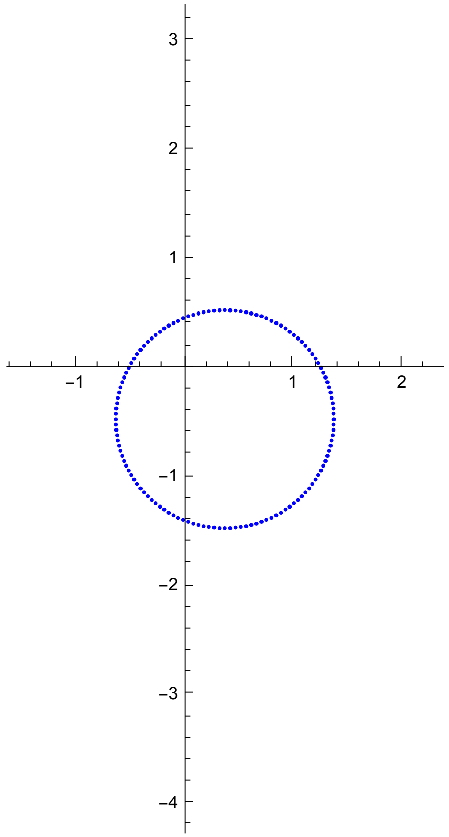


Real computations

We now do this with more rays.

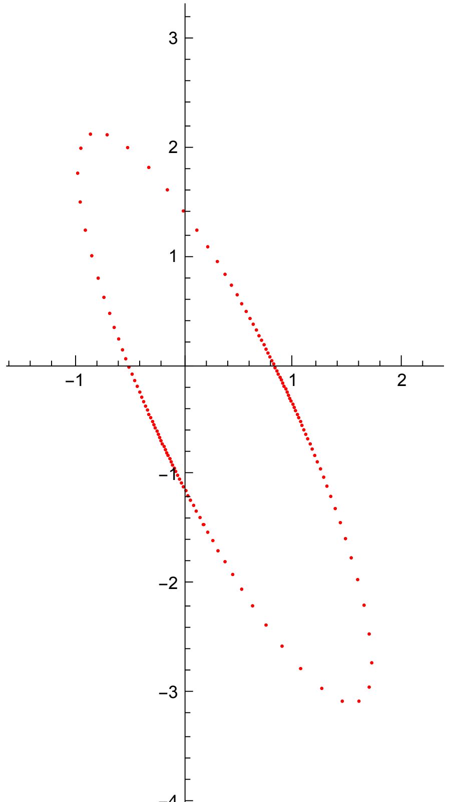
```
In[•]:= rays = Table[maxpt + {Sin[θ], Cos[θ]}, {θ, -Pi , Pi, Pi / 64}] // Union;
listrays = ListPlot[rays, PlotStyle → Blue]
```

Out[•]=

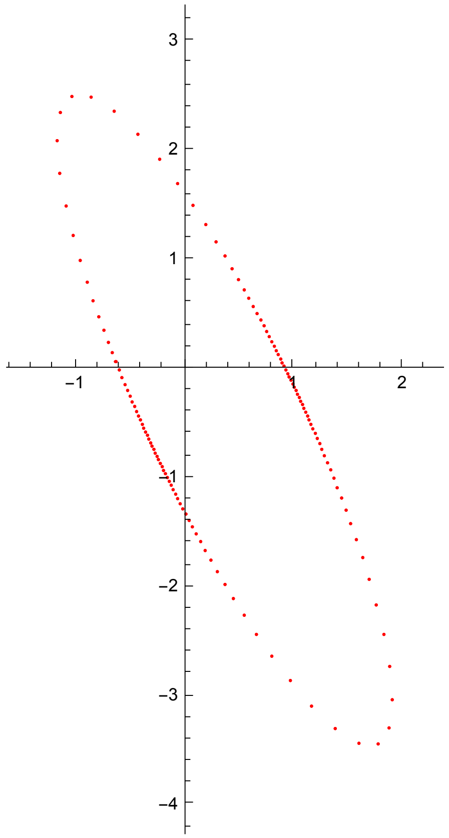


```
In[•]:= ptλ := λ maxpt + (1 - λ) ray

In[•]:= tab = Table[
            λs = λ /. NSolve[likfcn @@ ptλ == level90, λ];
            pts = Table[λs[[j]] maxpt + (1 - λs[[j]]) ray, {j, λs // Length}],
            {ray, rays}];
        list = Flatten[tab, 1];
        list = Cases[list, {__Real}];
        ray90 = ListPlot[list]
```

Out[•]=

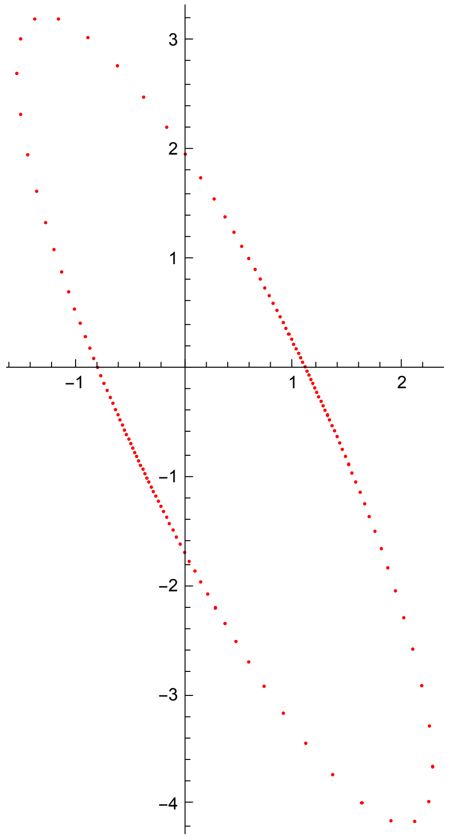


In[•]:=
```
tab = Table[
    λs = λ /. NSolve[likfcn @@ ptλ == level95, λ];
    pts = Table[λs[[j]] maxpt + (1 - λs[[j]]) ray, {j, λs // Length}],
    {ray, rays}];
list = Flatten[tab, 1];
list = Cases[list, {__Real}];
ray95 = ListPlot[list]
```

Out[•]=


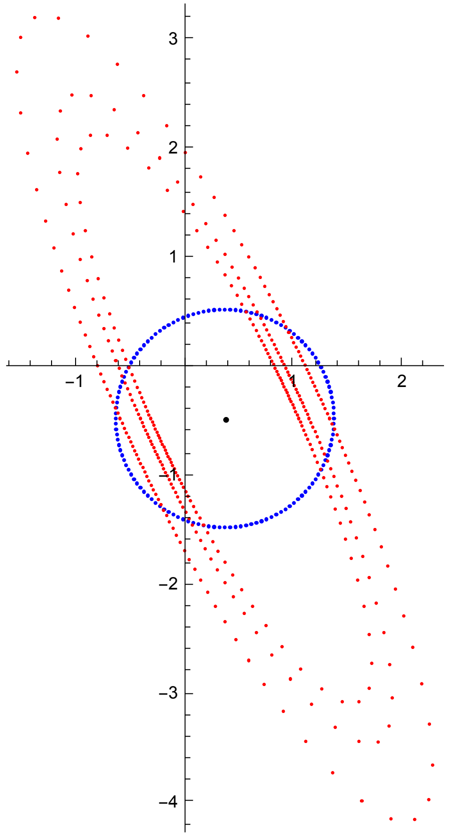
```
In[•]:= tab = Table[
            λs = λ /. NSolve[likfcn @@ ptλ == level99, λ];
            pts = Table[λs[[j]] maxpt + (1 - λs[[j]]) ray, {j, λs // Length}],
            {ray, rays}];
        list = Flatten[tab, 1];
        list = Cases[list, {__Real}];
        ray99 = ListPlot[list]
```

Out[•]=



In[ ]:=
```
Show[listrays, ray99, ray95, ray90, plotmaxpt]
```

Out[ ]=



One-parameter confidence sets

Let’s compute the confidence intervals for a and then b.

We first take the projections of the 2D sets to get rough bounds on the intervals

```
In[•]:= listT = list // Transpose;
        {amin, amax} = {Min[listT[[1]]], Max[listT[[1]]]}
        {bmin, bmax} = {Min[listT[[2]]], Max[listT[[2]]]}

Out[•]= {-1.52808, 2.2801}

Out[•]= {-4.16454, 3.19084}
```

Choose levels corresponding to one free parameter

```
In[•]:= levels = {level90, level95, level99} = maxlik - chisquare[[1]]

Out[•]= {-29.119, -30.254, -33.048}

In[•]:= step = 1. / 100;
```

Optimization approach

We choose a grid of a values in the interval [amin, amax]. For each a, we find the b which maximizes the likelihood function.
We keep those points which produce likelihoods in excess of the Chi-square level.
This produces the confidence level for a.

```
In[•]:= tab = Table[
            {a, FindMaximum[loglik, {b}, Method -> "Newton"][[1]]}, {a, amin, amax, step}];
        Select[tab, #[[2]] > level90 &] // Transpose // First;
        {aleft, aright} = {Min[%], Max[%]}

Out[•]= {-0.64808, 1.40192}
```

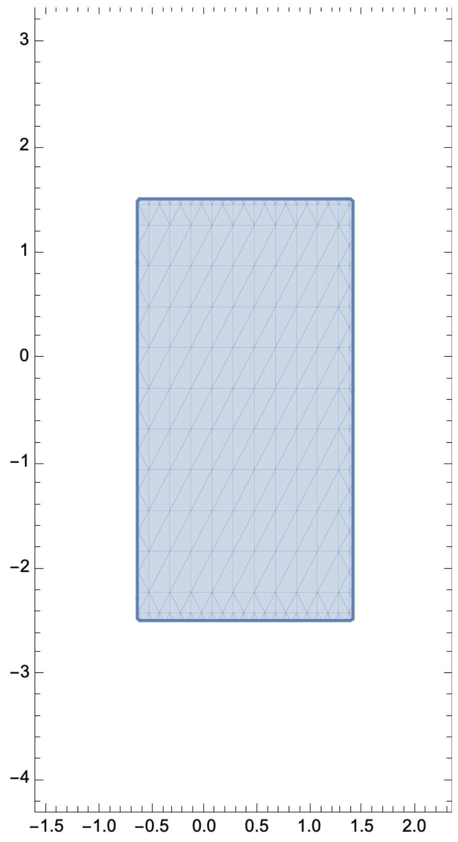
Repeat for b:

```
In[•]:= tab = Table[
            {b, FindMaximum[loglik, {a}, Method -> "Newton"][[1]]}, {b, bmin, bmax, step}];
        Select[tab, #[[2]] > level90 &] // Transpose // First;
        {bleft, bright} = {Min[%], Max[%]}

Out[•]= {-2.48454, 1.51546}
```

We now plot the rectangle defined by the two confidence intervals

```
In[•]:= CS1D90 = RegionPlot[
          aleft <= a <= aright && bleft <= b <= bright, {a, amin, amax}, {b, bmin, bmax}]
```

Out[•]=



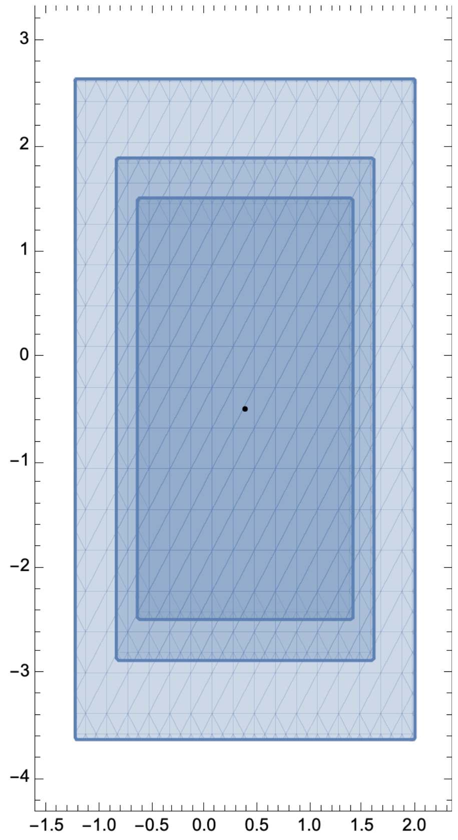
Repeat for 95% and 99% confidence levels:

```
In[•]:= tab = Table[
           {a, FindMaximum[loglik, {b}, Method -> "Newton"][[1]]}, {a, amin, amax, step}];
        Select[tab, #[[2]] > level95 &] // Transpose // First;
        {aleft, aright} = {Min[%], Max[%]};
        tab = Table[
           {b, FindMaximum[loglik, {a}, Method → "Newton"][[1]]}, {b, bmin, bmax, step}];
        Select[tab, #[[2]] > level95 &] // Transpose // First;
        {bleft, bright} = {Min[%], Max[%]};
        CS1D95 = RegionPlot[
           aleft <= a <= aright && bleft <= b <= bright, {a, amin, amax}, {b, bmin, bmax}];
```

```
In[•]:= tab = Table[
           {a, FindMaximum[loglik, {b}, Method -> "Newton"][[1]]}, {a, amin, amax, 0.01}];
        Select[tab, #[[2]] > level99 &] // Transpose // First;
        {aleft, aright} = {Min[%], Max[%]};
        tab = Table[
           {b, FindMaximum[loglik, {a}, Method → "Newton"][[1]]}, {b, bmin, bmax, 0.01}];
        Select[tab, #[[2]] > level99 &] // Transpose // First;
        {bleft, bright} = {Min[%], Max[%]};
        CS1D99 = RegionPlot[
           aleft <= a <= aright && bleft <= b <= bright, {a, amin, amax}, {b, bmin, bmax}];
```

In[•]:=
```
Show[CS1D90, CS1D95, CS1D99, plotmaxpt]
```

Out[•]=

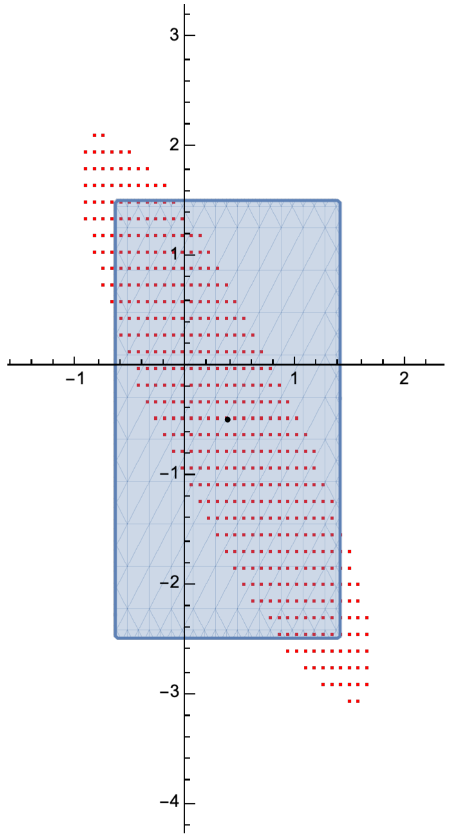


Comparison

Display the 90% (95%, 99%) 2D confidence sets and the boxes defined by the 1D confidence intervals.

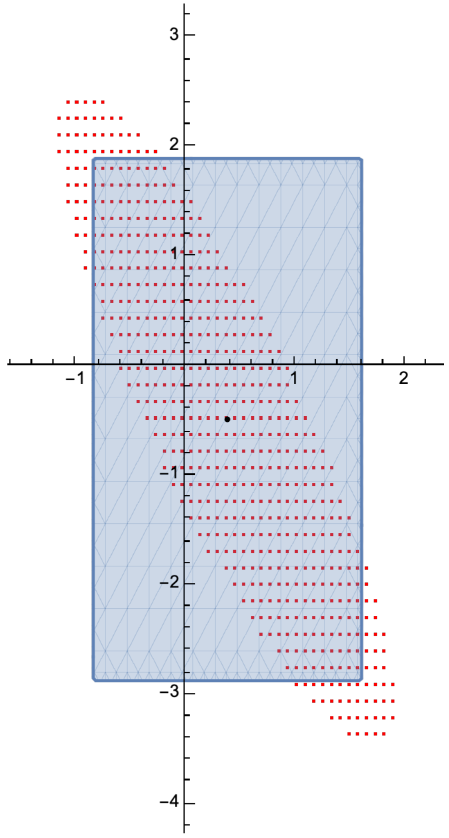
In[∘]:=
```
Show[mesh90, CS1D90, plotmaxpt]
```

Out[∘]=



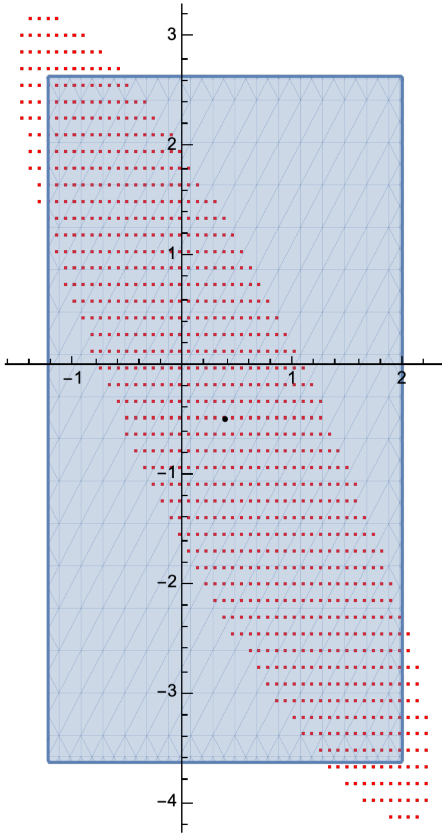
In[∘]:=
```
Show[mesh95, CS1D95, plotmaxpt]
```

Out[∘]=



In[•]:=
```
Show[mesh99, CS1D99, plotmaxpt]
```

Out[•]=


3
2
1
−1
1
2
−1
−2
−3
−4