Example 3: two variables, three parameters

In[13]:=
```
RV := RandomVariate[NormalDistribution[0, 1], WorkingPrecision → 32]
SetOptions[ListPlot, AspectRatio -> 1];
SetOptions[ListPlot3D, AspectRatio -> 1];
SetOptions[ContourPlot, AspectRatio -> 1];
SetOptions[ContourPlot3D, AspectRatio -> 1];
SeedRandom[Method → "MersenneTwister"];
SetOptions[FindMaximum, AccuracyGoal -> 5, PrecisionGoal -> 5];
```

In[19]:=
```
df1 = {2.706, 3.841, 6.635};
df2 = {4.605, 5.991,  9.210};
df3 = {6.251 , 7.815 , 11.345};
df4 = {7.779, 9.488 , 13.277};
df5 = {9.236 , 11.070 ,  15.086};
chisquare = {df1, df2, df3, df4, df5}
```

Out[24]=
```
{{2.706, 3.841, 6.635}, {4.605, 5.991, 9.21},
 {6.251, 7.815, 11.345}, {7.779, 9.488, 13.277}, {9.236, 11.07, 15.086}}
```

Example 3 - two variables and three parameters

Two variables but three parameters in the nonlinear model

1223 has multiple solutions

In[25]:=
```
SeedRandom[1223];
```

In[26]:=
```
SeedRandom[0];
```

This example has two parameters but also two variables. The functional form is nonlinear.

In[27]:=
```
numdatapoints = 30;
```

Model:

In[28]:=
```
params = {a, b, c};
numparams = Length[params];
truparams = Thread[params -> 0]
vars = {x, y};
Clear[model]
model[x_, y_] = a + b x + b^2 y + c x^2;
```

Out[30]=
$$\{a \to 0, b \to 0, c \to 0\}$$

Generate data (uniform on [0, 1])

In[34]:=
```
data = Table[{RandomReal[], RandomReal[]}, {numdatapoints}];
```

I assume that the true value of each parameter is zero.

In[35]:=
```
truth = model @@@ data /. truparams;
```

Observations

obs is the vector of observations

In[36]:=
```
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[38]:=
```
noisehat = obs - model @@@ data;
```

In[39]:=
```
loglik = -noisehat.noisehat // Expand
```

Out[39]=
$$-28.4574 - 7.58191\,a - 30\,a^2 - 3.73047\,b - 23.8138\,a\,b - 10.0135\,b^2 - 33.1892\,a\,b^2 - 13.079\,b^3 - 11.5886\,b^4 - 2.16039\,c - 13.0999\,a\,c - 8.3222\,b\,c - 7.10431\,b^2\,c - 2.87264\,c^2$$

```
likfcn[a_, b_, c_] = loglik;
```

Out[40]=
$$-28.4574 - 7.58191\,a - 30\,a^2 - 3.73047\,b - 23.8138\,a\,b - 10.0135\,b^2 - 33.1892\,a\,b^2 - 13.079\,b^3 - 11.5886\,b^4 - 2.16039\,c - 13.0999\,a\,c - 8.3222\,b\,c - 7.10431\,b^2\,c - 2.87264\,c^2$$

```
estimate = FindMaximum[loglik, {a, b, c}] // N;
```

Out[41]=
$$\{-27.8303, \{a \to -0.117969, b \to -0.412851, c \to 0.280218\}\}$$

```
maxlik = estimate[[1]]
```

```
maxpt = params /. estimate[[2]]
```

Out[43]=
$$\{-0.117969, -0.412851, 0.280218\}$$

Three-parameter confidence sets

We choose a level for the likelihood function corresponding to the 95% confidence set

```
In[•]:= levels = {level90, level95, level99} = maxlik - chisquare[[numparams]]
Out[•]= {-34.0813, -35.6453, -39.1753}
```

Contour evaluation

In[•]:=
```
maxlik
```

Out[•]=
```
-27.8303
```

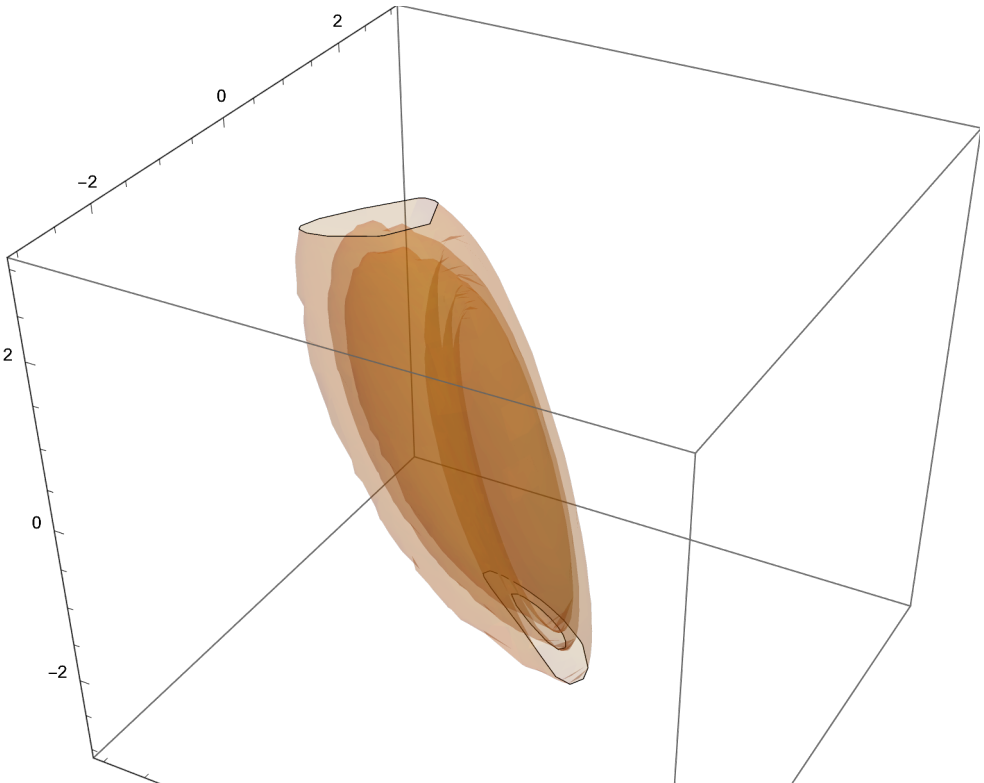
In[•]:=
```
levels = {level90, level95, level99} = maxlik - chisquare[[numparams]]
```

Out[•]=
```
{-34.0813, -35.6453, -39.1753}
```

In[•]:=
```
range = 3;
```

In[•]:=
```
ContourPlot3D[loglik, {a, -range, range}, {b, -range, range}, {c, -range, range},
 Contours → levels, Mesh -> None, ContourStyle → Directive[Orange, Opacity[0.2]]]
```

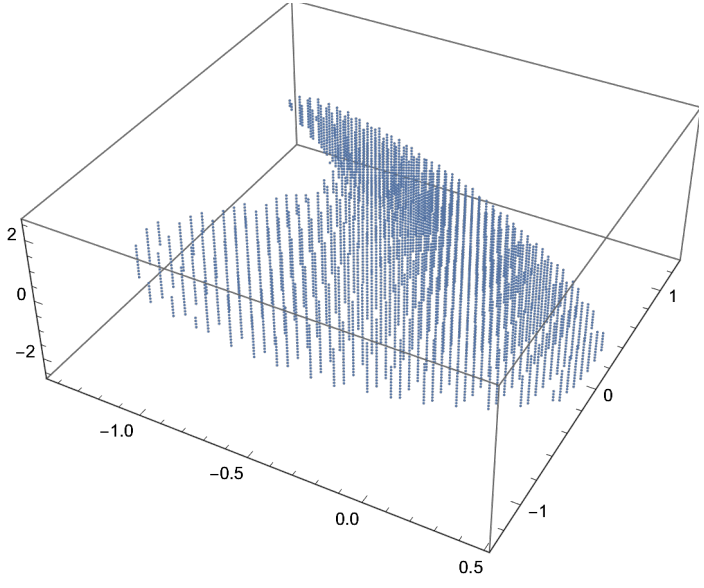
Out[•]=



Compute likelihood on a 3D grid

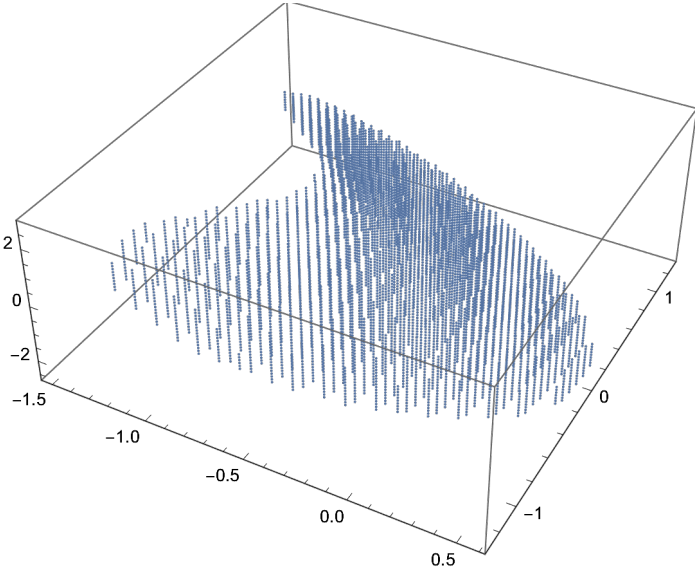
```
In[•]:= step = 0.1;

In[•]:= grid = Range[-range, range, step];
        gridpts = Flatten[Outer[List, grid, grid, grid], 2];
        gridpts = Map[maxpt + # &, gridpts];
        gridvals = Map[likfcn @@ # &, gridpts];
        griddata = {gridpts, gridvals} // Transpose;

In[•]:= CS90 = data90 = First /@ (Select[griddata, #[[2]] > level90 &]);
        mesh90 = ListPointPlot3D[data90]
```

Out[•]=



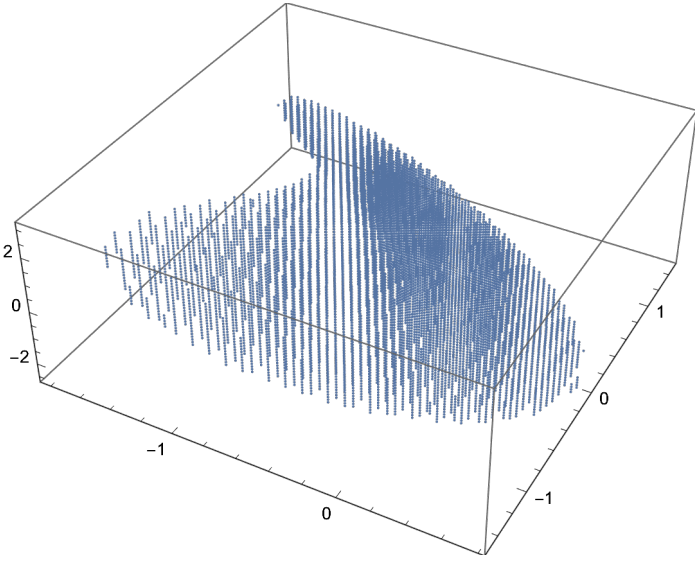
In[•]:=
```
data95 = First /@ (Select[griddata, #[[2]] > level95 &]);
CS95 = mesh95 = ListPointPlot3D[data95]
```

Out[•]=



```
In[•]:= data99 = First /@ (Select[griddata, #[[2]] > level99 &]);
        CS99 = mesh99 = ListPointPlot3D[data99]
```

Out[•]=




```
In[•]:= Length /@ {data90, data95, data99}
```

```
Out[•]= {6310, 8240, 12 650}
```

Two-parameter confidence sets

```
In[•]:= levels = {level90, level95, level99} = maxlik - chisquare[[2]]
Out[•]= {-32.4353, -33.8213, -37.0403}
```

Fix two parameters and compute the boundary value for the third.

```
In[•]:= step = 0.2;
```

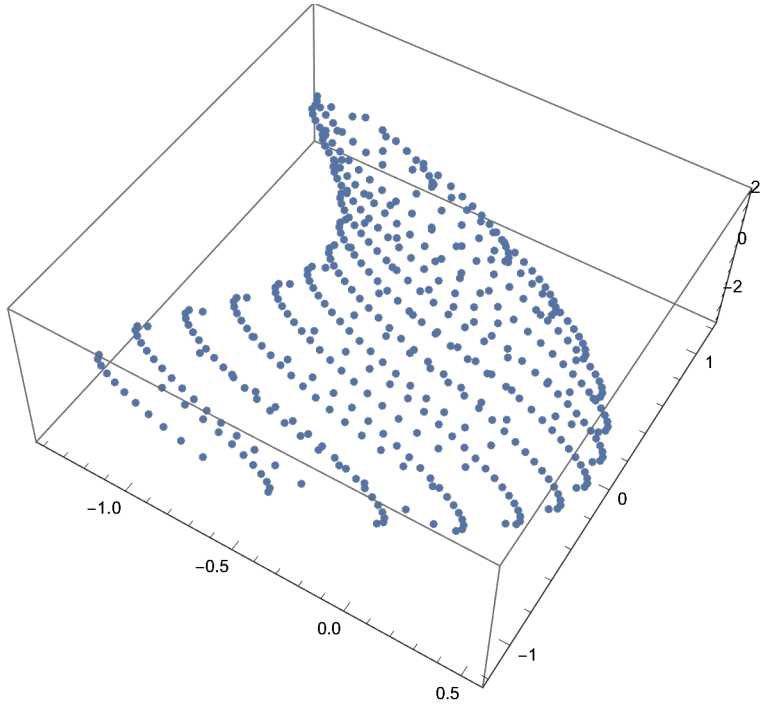
Fix b and c:

In[•]:=
```
Clear[a, b, c]
tab = Table[{a, b, c} /. NSolve[loglik == level95, a],
    {b, -range, range, step}, {c, -range, range, step}]; // AbsoluteTiming
list1 = Flatten[tab, 2];
list1 = Cases[list1, {__Real}];
% // Length
ListPointPlot3D[list1]
```

Out[•]= {0.953064, Null}

Out[•]= 468

Out[•]=

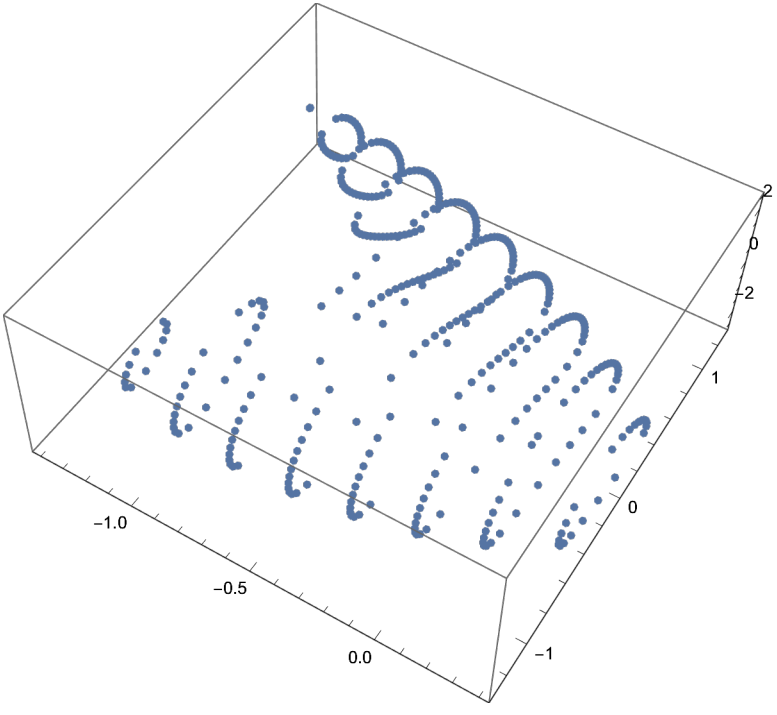


Fix a and c:

In[•]:=
```
Clear[a, b, c]
tab = Table[{a, b, c} /. NSolve[loglik == level95, b],
    {a, -range, range, step}, {c, -range, range, step}];
list2 = Flatten[tab, 2];
list2 = Cases[list2, {__Real}];
% // Length
ListPointPlot3D[list2]
```

Out[•]= 390

Out[•]=



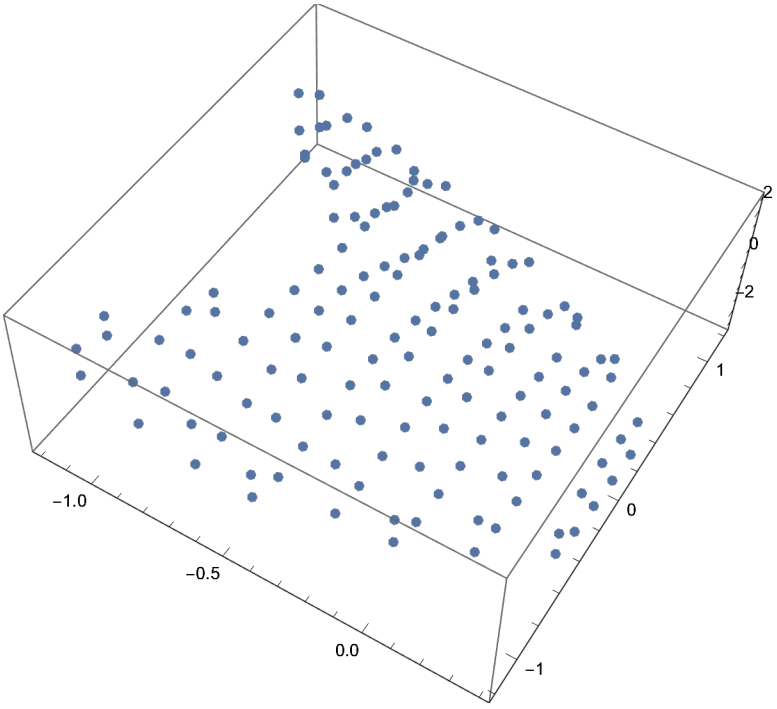
Fix b and c:

In[•]:=
```
Clear[a, b, c]
tab = Table[{a, b, c} /. NSolve[loglik == level95, c],
    {b, -range, range, step}, {a, -range, range, step}]; // AbsoluteTiming
list3 = Flatten[tab, 2];
list3 = Cases[list3, {__Real}];
% // Length
ListPointPlot3D[list3]
```

Out[•]= {0.97418, Null}

Out[•]= 148

Out[•]=

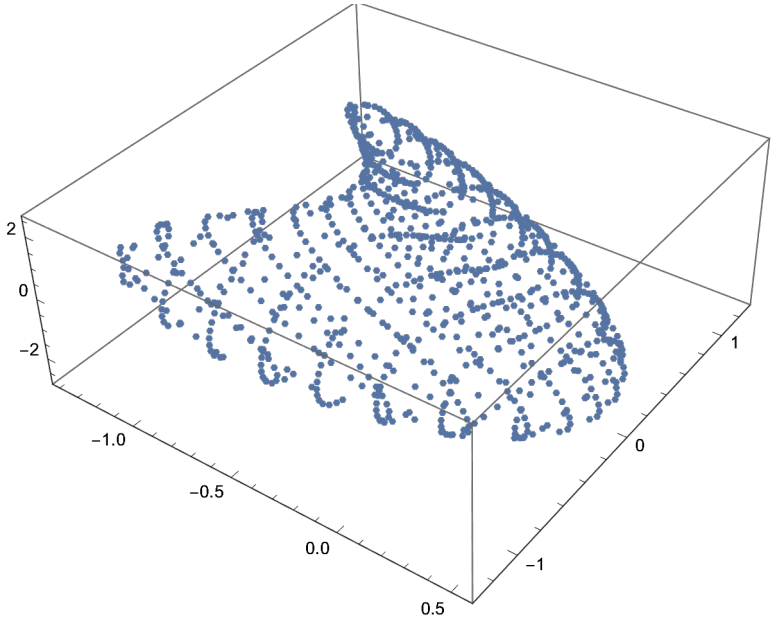


Merge the different point sets

In[•]:=
```
CS95 = list = Union[list1, list2, list3];
% // Length
ListPointPlot3D[list]
```

Out[•]=
```
1006
```

Out[•]=



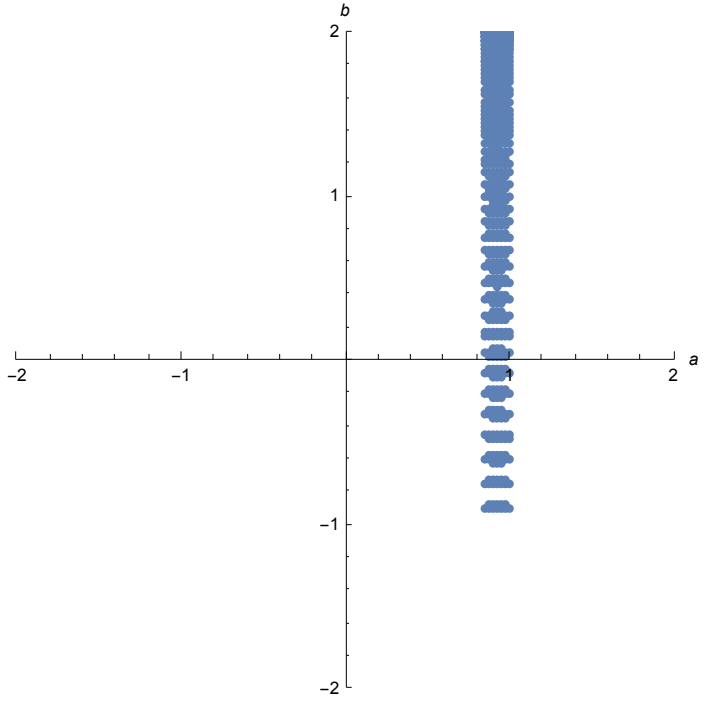
Confidence sets for different pairs using this boundary approximation

In[ ]:=
```
range = 2;
```

In[ ]:=
```
CScolumns = CS // Transpose;
```

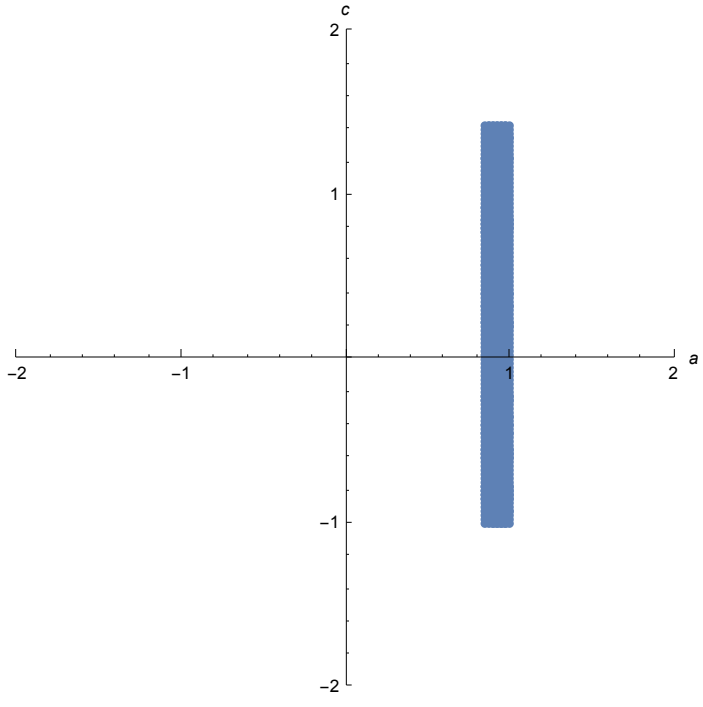
In[ ]:=
```
ListPlot[CScolumns[[{1, 2}]] // Transpose, PlotStyle → PointSize[Medium],
 PlotRange → {{-range, range}, {-range, range}}, AxesLabel → {a, b}]
```

Out[ ]=



In[•]:=
```
ListPlot[CScolumns[[{1, 3}]] // Transpose, PlotStyle → PointSize[Medium],
 PlotRange → {{-range, range}, {-range, range}}, AxesLabel → {a, c}]
```

Out[•]=



In[•]:=
```
ListPlot[CScolumns[[{2, 3}]] // Transpose, PlotStyle → PointSize[Medium],
 PlotRange → {{-range, range}, {-range, range}}, AxesLabel → {b, c}]
```

Out[•]=