Least Squares Refinement with Error Calculation

Marshall Bradley

The method of least squares refinement with error calculation for finding the best fit to a nonlinear curve in the presence of unknown measurement error is presented. The technique is based on approximating the relationship between the observed data and the model as linear and successively improving the approximation. The results are compared to a Bayesian approach with numerical computations performed by the Metropolis Markov Chain Monte Carlo algorithm. In the numerical example considered here agreement is found to be good between the two approaches.

Least squares refinement

We consider a problem from from Clifford (1973) in which a small number of data observations are fitted to an equation that depends upon three parameters. Additionally it is assumed that the errors with which the observations are are measured are independent, identically distributed but with unknown standard deviation σ. The equation relating the observations (same thing as data) to the parameters is

$$z(c) = \frac{1+sc}{t+uc} \ .$$

The unknown parameters are $\theta = (s,\ t,\ u)$. The observations are $z_i : i = 1,\ 2 \ldots n$. The quantity c is a covariate which is assumed to be measured without error.

In order to to distinguish between z the data and our model for z let us define the function

$$g(s,\ t,\ u,\ c) = \frac{1+sc}{t+uc} \ .$$

Lets make a guess as to the solution $\boldsymbol{\theta}_g = \left(s_g,\ t_g,\ u_g\right)$ which bests fits the data. Then on a linear approximation

$$z_i \ = g\left(s_g,\ t_g,\ u_g,\ c_i\right) + \left\{\frac{\partial g}{\partial s},\ \frac{\partial g}{\partial t},\ \frac{\partial g}{\partial u}\right\}.\left\{s - s_g,\ ,\ t - t_g,\ u - u_g\right\}.$$

If we define $z_{g,i} = g\left(s_g,\ t_g,\ u_g,\ c_i\right)$ then the relationship between our observations and our model on a linear approximation can be written

$$\boldsymbol{z}_{\text{obs}} - \boldsymbol{z}_g \ = \boldsymbol{A}\left(\boldsymbol{\theta} - \boldsymbol{\theta}_g\right)$$

where $\boldsymbol{z}_{\text{obs}}$ is the vector of observations and $\boldsymbol{A}$ is in $n \times 3$ matrix whose rows are $\left\{\frac{\partial g}{\partial s},\ \frac{\partial g}{\partial t},\ \frac{\partial g}{\partial u}\right\}$ evaluated at $\left(s_g,\ t_g,\ u_g,\ c_i\right)$.

Now assume that the covariance matrix of the measurements is the $n \times n$ matrix

$$\boldsymbol{M}_z = \sigma^2 \boldsymbol{I}$$

where $\boldsymbol{I}$ is the $n \times n$ identity matrix and σ^2 is the unknown variance of the measurement error. We weight our observations inversely proportional to the variance associated with the measurement. This leads to

$$(\boldsymbol{M}_z)^{-1} (\boldsymbol{z}_{\text{obs}} - \boldsymbol{z}_g) = (\boldsymbol{M}_z)^{-1} \boldsymbol{A}(\boldsymbol{\theta} - \boldsymbol{\theta}_g).$$

We can make this a square matrix equation (3×3) by multiplying both sides by $\boldsymbol{A}^T$:

$$\boldsymbol{A}^T(\boldsymbol{M}_z)^{-1} (\boldsymbol{z}_{\text{obs}} - \boldsymbol{z}_g) = \boldsymbol{A}^T(\boldsymbol{M}_z)^{-1} \boldsymbol{A}(\boldsymbol{\theta} - \boldsymbol{\theta}_g).$$

The least squares solution to this equation for $\boldsymbol{\theta} - \boldsymbol{\theta}_g$ is

$$(\boldsymbol{\theta} - \boldsymbol{\theta}_g) = \left[\boldsymbol{A}^T(\boldsymbol{M}_z)^{-1} \boldsymbol{A}\right]^{-1} \boldsymbol{A}^T(\boldsymbol{M}_z)^{-1} (\boldsymbol{z}_{\text{obs}} - \boldsymbol{z}_g).$$

An update for the parameter vector $\boldsymbol{\theta} = (s, t, u)$ is

$$\boldsymbol{\theta}_{\text{update}} = \boldsymbol{\theta}_g + \left[\boldsymbol{A}^T(\boldsymbol{M}_z)^{-1} \boldsymbol{A}\right]^{-1} \boldsymbol{A}^T(\boldsymbol{M}_z)^{-1} (\boldsymbol{z}_{\text{obs}} - \boldsymbol{z}_g).$$

Because of our assumptions about the covariance matrix of the measurements this simplifies to

$$\boldsymbol{\theta}_{\text{update}} = \boldsymbol{\theta}_g + \left[\boldsymbol{A}^T \boldsymbol{A}\right]^{-1} \boldsymbol{A}^T (\boldsymbol{z}_{\text{obs}} - \boldsymbol{z}_g).$$

This can be repeated over and over with the guess from the last step being replaced by the update until (hopefully) convergence occurs. This process is called least squares refinement.

The covariance matrix of the parameters is

$$\boldsymbol{M}_\theta = \left[\boldsymbol{A}^T(\boldsymbol{M}_z)^{-1} \boldsymbol{A}\right]^{-1}$$

Since we have assumed that our measurement errors have a covariance matrix that is proportional to an identity matrix, the covariance matrix of the parameter is in this case the 3×3 matrix

$$\boldsymbol{M}_\theta = \sigma^2 \left[\boldsymbol{A}^T \boldsymbol{A}\right]^{-1}.$$

The diagonal entries in this matrix are our parameter variances $(\sigma_s^2, \sigma_t^2, \sigma_u^2)$. They are proportional to σ^2 which we do not as yet know. An estimate for σ^2 is

$$\sigma^2 = \frac{1}{n-3} \sum_{i=1}^{n} \left[z_i - g(s_g, t_g, u_g, c_i)\right]^2$$

where (s_g, t_g, u_g) is the final estimate of the parameters from the least squares refinement procedure. In the foregoing equation we divide by $n - 3$ because 3 parameters have been estimated from the n observations.

Error calculation using least squares refinement

In the analysis that follows it will be necessary to have the function:

In[•]:=
```
Clear[g, s, t, u, c];
g[s_, t_, u_, c_] := (1 + s * c)/(t + u * c)
```

Since we are going to make a linear approximate we need the partial derivative of this function:

```
In[•]:= {D[g[s, t, u, c], s], D[g[s, t, u, c], t], D[g[s, t, u, c], u]}
```

```
Out[•]=
```

$$\left\{\frac{c}{t+c\,u}, -\frac{1+c\,s}{(t+c\,u)^2}, -\frac{c\,(1+c\,s)}{(t+c\,u)^2}\right\}$$

In functional form they are defined to be:

```
In[•]:= dg[s_, t_, u_, c_] := {c/(t + c u), -(1 + c s)/(t + c u)^2, -(c (1 + c s))/(t + c u)^2}
```

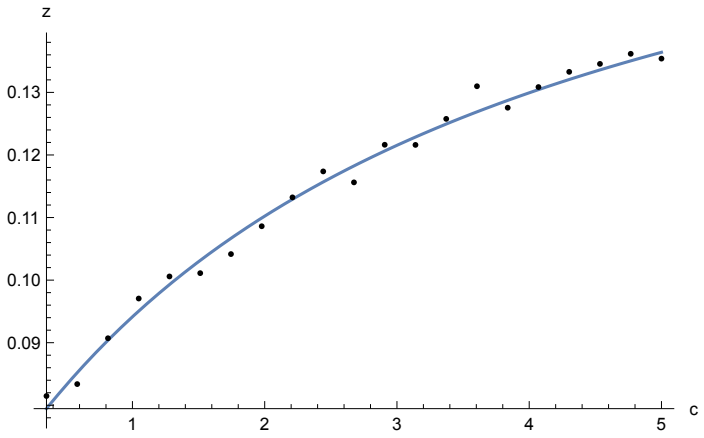
We begin by generating some synthetic data. The are:

```
In[•]:= SeedRandom[12349];
strue = 0.718; ttrue = 14.4; utrue = 3.85; σtrue = 0.002;
cmin = 0.35; cmax = 5.0;
nSample = 21;
cmeas = Table[c, {c, cmin, cmax, (cmax - cmin)/(nSample - 1)}];
zmeas = Table[g[strue, ttrue, utrue, cmeas[[i]]] +
    RandomReal[NormalDistribution[0.0, σtrue]], {i, 1, nSample}];
```

A comparison of the data to the true curve $z_{\text{true}}(c) = g(s_{\text{true}}, t_{\text{true}}, u_{\text{true}}, c)$ is:

```
In[•]:= data = Transpose[{cmeas, zmeas}];
Plot[g[strue, ttrue, utrue, c], {c, cmin, cmax}, Epilog → {…}, AxesLabel → {…}]
```

```
Out[•]=
```



It will also be convenient to have the sum of squared differences:

```
In[•]:= chiSquare[s_, t_, u_] := Sum[(zmeas[[i]] - g[s, t, u, cmeas[[i]]])^2, {i, 1, Length[zmeas]}]
```

Just for fun we can as for the Wolfram Language to minimize this quantity and give us the maximum likelihood solution:

In[ ]:=
```
sol = FindMinimum[chiSquare[s, t, u], {{s, 3}, {t, 3}, {u, 2}}];
{sML, tML, uML} = {s, t, u} /. Last[sol]
```

Out[ ]=
```
{0.611129, 14.0234, 3.10013}
```

The least squares refinement procedure is:

In[ ]:=
```
{sguess, tguess, uguess} = {3, 3, 2};
lsi = Table[
   Clear[c];
   gc = {c/(t + c u), -(1 + c s)/(t + c u)^2, -(c (1 + c s))/(t + c u)^2} /. {s → sguess, t → tguess, u → uguess};
   Aforward = Table[gc /. {c → cmeas[[i]]}, {i, 1, Length[cmeas]}];
   Δz = zmeas - Table[g[sguess, tguess, uguess, cmeas[[i]]], {i, 1, Length[cmeas]}];
   {sguess, tguess, uguess} = {sguess, tguess, uguess} +
     Inverse[Transpose[Aforward].Aforward].Transpose[Aforward].Δz;
   SSD =
    Sum[(zmeas[[i]] - g[sguess, tguess, uguess, cmeas[[i]]])^2, {i, 1, Length[zmeas]}];
   {SSD, {sguess, tguess, uguess}}, {count, 1, 10}];
Last@lsi
```

Out[ ]=
```
{0.0000569885, {0.611129, 14.0234, 3.10013}}
```

Our estimate of (*s*, *t*, *u*) is :

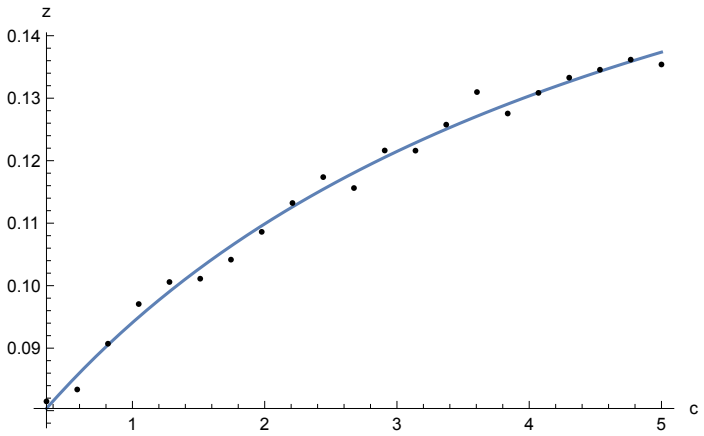
In[ ]:=
```
{sLS, tLS, uLS} = Last@Last[lsi]
```

Out[ ]=
```
{0.611129, 14.0234, 3.10013}
```

A comparison of the data to the least squares solution is shown in the following plot:

In[ ]:=
```
data = Transpose[{cmeas, zmeas}];
Plot[g[sLS, tLS, uLS, c], {c, cmin, cmax}, Epilog → {…} + , AxesLabel → {…} + ]
```

Out[ ]=



A comparison of the true parameter values to the maximum likelihood and least squares refinement estimates is:

```
In[ ]:= TableForm[{{strue, ttrue, utrue}, {sML, tML, uML}, {sLS, tLS, uLS}}, ... → ... ]
```

Out[ ]//TableForm=

	s	t	u
True	0.718	14.4	3.85
Max Likelihood	0.611129	14.0234	3.10013
Least Squares	0.611129	14.0234	3.10013

An estimate of σ is (we estimate 3 parameters form 8 data points so there are 5 degrees of freedom):

```
In[ ]:= σ = (SSD/5)^(1/2)
```

Out[ ]=

```
0.00337605
```

The posterior covariance matrix is:

```
In[ ]:= postCov = σ^2 Inverse[Transpose[Aforward].Aforward];
        MatrixForm[%]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0.0343026 & 0.136635 & 0.230406 \\ 0.136635 & 0.632327 & 0.890966 \\ 0.230406 & 0.890966 & 1.55815 \end{pmatrix}$$

The standard deviations of our estimates of (s, t, u) are:

```
In[ ]:= {σs, σt, σu} = Sqrt[Diagonal[postCov]]
```

Out[ ]=

```
{0.18521, 0.79519, 1.24826}
```

A table of useful quantities is:

```
In[ ]:= Table[{cmeas[[i]], zmeas[[i]],
    D[g[s, t, u, c], s] /. {s → sguess, t → tguess, u → uguess, c → cmeas[[i]]},
    D[g[s, t, u, c], t] /. {s → sguess, t → tguess, u → uguess, c → cmeas[[i]]},
    D[g[s, t, u, c], u] /. {s → sguess, t → tguess, u → uguess, c → cmeas[[i]]},
    g[s, t, u, c] /. {s → sguess, t → tguess, u → uguess, c → cmeas[[i]]}},
  {i, 1, Length[zmeas]}];
TableForm[%, ... → ...]
```

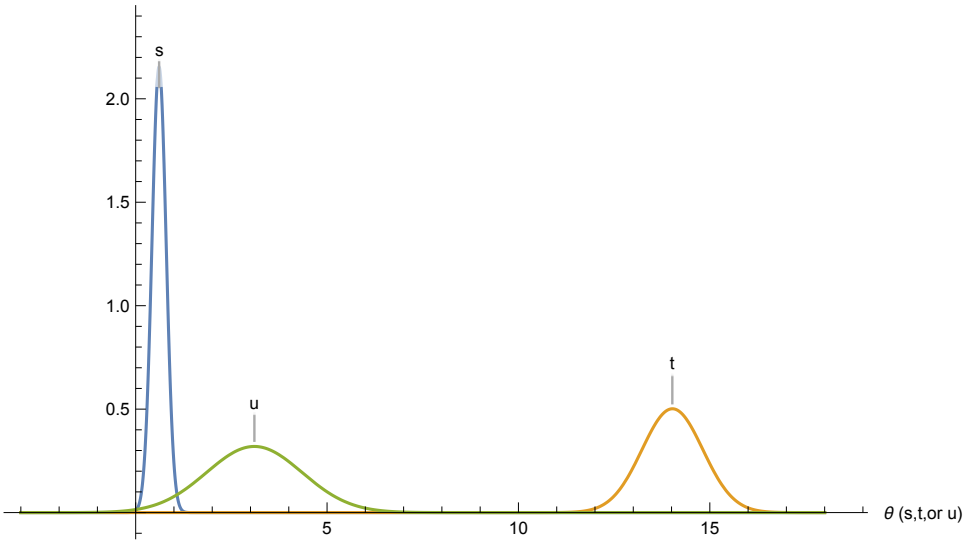
Out[ ]//TableForm=

c_i	z_i	$\partial_s\ g_i$	$\partial_t\ g_i$	$\partial_u\ g_i$	$z_{LS,i}$
0.35	0.0814806	0.0231659	−0.00531795	−0.00186128	0.0803457
0.5825	0.0833852	0.0367991	−0.00541175	−0.00315234	0.0856635
0.815	0.090711	0.0492448	−0.00546939	−0.00445755	0.0905181
1.0475	0.0970586	0.0606517	−0.00549875	−0.00575994	0.0949675
1.28	0.100587	0.0711447	−0.00550595	−0.00704761	0.0990604
1.5125	0.101114	0.0808292	−0.00549574	−0.00831231	0.102838
1.745	0.104162	0.0897954	−0.00547187	−0.00954842	0.106335
1.9775	0.10861	0.0981202	−0.0054373	−0.0107523	0.109582
2.21	0.113225	0.10587	−0.00539437	−0.0119215	0.112605
2.4425	0.117372	0.113103	−0.00534496	−0.0130551	0.115427
2.675	0.115621	0.119868	−0.00529058	−0.0141523	0.118065
2.9075	0.12163	0.12621	−0.00523243	−0.0152133	0.120539
3.14	0.121608	0.132167	−0.00517149	−0.0162385	0.122863
3.3725	0.125767	0.137774	−0.00510855	−0.0172286	0.12505
3.605	0.130982	0.14306	−0.00504425	−0.0181845	0.127112
3.8375	0.127549	0.148051	−0.0049791	−0.0191073	0.129059
4.07	0.130852	0.152773	−0.00491352	−0.019998	0.1309
4.3025	0.133284	0.157246	−0.00484784	−0.0208578	0.132645
4.535	0.134554	0.161489	−0.00478235	−0.021688	0.1343
4.7675	0.136167	0.16552	−0.00471727	−0.0224896	0.135872
5.	0.135397	0.169354	−0.00465276	−0.0232638	0.137368

If we assume Gaussian statistics then our error distributions for the least squares refinement solution look like this:

In[ ]:=
```
gLS = Plot[{Callout[PDF[NormalDistribution[sLS, σs], theta], "s", Above],
    Callout[PDF[NormalDistribution[tLS, σt], theta], "t", Above], Callout[
     PDF[NormalDistribution[uLS, σu], theta], "u", Above]}, {theta, -3, 18}, ···]
```

Out[ ]=



Metropolis MCMC Solution

In this section we check our results using a Metropolis Markov Chain Monte Carlo (MCMC) technique. The actual algorithm we use is adapted from Brewer (). To use the algorithm we need an expression for the log likelihood of the data including appropriate modifications for prior probability distributions.

The likelihood of the data is

$$L(s, t, u, \sigma) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left\{-\frac{1}{2\sigma^2}\,[z_i - g(s, t, , u, c_i)]^2\right\}.$$

This can be written

$$L(s, t, u, \sigma) = \frac{1}{(2\pi)^{n/2}\,\sigma^n} \exp\left\{-\frac{1}{2\sigma^2}\,\sum_{i=1}^{n}[z_i - g(s, t, , u, c_i)]^2\right\}.$$

The log likelihood is

$$\text{logL}(s, t, u, \sigma) = \log\left(\frac{1}{(2\pi)^{n/2}\,\sigma^n}\right) - \frac{1}{2\sigma^2}\,\sum_{i=1}^{n}[z_i - g(s, t, , u, c_i)]^2.$$

The parameter σ is a scale parameter. The appropriate choice for a prior probability distribution for this parameter is uniform in log. This is known as a Jeffreys prior. We assume flat priors on the other parameters. The modified log likelihood is

$$\text{logL}(s, t, u, \sigma) = \log\left(\frac{1}{\sigma}\right) + \log\left(\frac{1}{(2\pi)^{n/2}\,\sigma^n}\right) - \frac{1}{2\sigma^2}\,\sum_{i=1}^{n}[z_i - g(s, t, , u, c_i)]^2.$$

The log likelihood of the data including a Jeffreys prior is given in the following block of code:

In[ ]:=
```
Clear[logLikelihood];
logLikelihood = Compile[{{θ, _Real, 1}},
   Module[{s, t, u, σ, Nd, logL},
    {s, t, u, σ} = θ;
    Nd = Length[zmeas];
    logL = -Log[σ] + Nd/2.0 Log[1.0/(2.0 N[Pi] σ^2)] -
       1.0/(2.0 σ^2) Sum[(zmeas[[i]] - g[s, t, u, cmeas[[i]]])^2, {i, 1, Nd}];
    logL
   ], CompilationOptions → {"InlineExternalDefinitions" → True},
   Parallelization → True, CompilationTarget → "WVM"];
```

We work in terms of particles. A particle is the combination of parameter values and the log likelihood of those parameter values in the form (logLikelihood, *s*, *t*, *u*, *σ*).

An example is:

In[ ]:=
```
θM = {0.1, 11.0, 3.0, 0.005};
particleM = Join[{logLikelihood[θM]}, θM]
```

Out[ ]=
```
{-1074.22, 0.1, 11., 3., 0.005}
```

We assume flat priors on the (*s*, *t*, *u*). Since we have modified the log likelihood function with a Jeffreys prior in *σ*, we can treat this parameter as flat in terms of defining a finite input range. For (*s*, *t*, *u*) we choose to work on a plus or minus 5 sigma range centered on the least squares refinement estimates. We limit *σ* to the range (00 001, 0.01):

In[ ]:=
```
nσ = 5;
pdfFList = {UniformDistribution[{sLS - nσ * σs, sLS + nσ * σs}],
   UniformDistribution[{tLS - nσ * σt, tLS + nσ * σt}], UniformDistribution[
    {uLS - nσ * σu, uLS + nσ * σu}], UniformDistribution[{0.00001, 0.01}]};
priorSupport = Map[First, pdfFList]
```

Out[ ]=
```
{{-0.314919, 1.53718}, {10.0474, 17.9993}, {-3.14116, 9.34142}, {0.00001, 0.01}}
```

Our MCMM algorithm is defined here:

```
In[ ]:= mcmcfun = Compile[{{particleM, _Real, 1}, {priorSupport, _Real, 2}},
    Module[{logLM, θM, σP, θP, priorRelativeP0,
      priorRelativeP, localMin, localMax, logLP, test, output},
     (* Strip off particle log likelihood and particle parameters. *)
     logLM = First[particleM];
     θM = Take[particleM, {2, Length[particleM]}];
     (* Heavy tailed proposal distribution. *)
     σP = Table[10^(1.5-6.0 Random[]), {Length[particleM] - 1}];
     θP = Table[RandomReal[NormalDistribution[θM[[i]], σP[[i]]]], {i, 1, Length[θM]}];
     (* Compute prior support of the proposal. *)
     priorRelativeP0 = Table[
       {localMin, localMax} = priorSupport[[i]];
       If[localMin < θP[[i]] < localMax, 1.0, 0.0], {i, 1, Length[θP]}];
     priorRelativeP = Apply[Times, priorRelativeP0];
     (* Metropolis step. *)
     If[priorRelativeP < 1.0, output = Join[{logLM}, θM],
      logLP = logLikelihood[θP];
      test = logLP - logLM;
      output = If[test ≥ Log[RandomReal[]], Join[{logLP}, θP], Join[{logLM}, θM]]];
     output], CompilationOptions → {"InlineExternalDefinitions" → True},
    Parallelization → True, CompilationTarget → "WVM"];
```

The input to the algorithm is a particle. The output is a particle as well. When combined with the NestList command the algorithm will produce a chain of samples from the posterior distribution $P(s, t, u, \sigma \mid D)$ where D is the data. This samples can be processed using simple techniques (histograms) and elementary statistics to make inferences about the parameters (s, t, u, σ).

An example single output from the algorithm is:

```
In[ ]:= mcmcfun[particleM, priorSupport]
Out[ ]=
{-1074.22, 0.1, 11., 3., 0.005}
```

Here we build a chain of length 600,000 and discard the first 100,000 to insure convergence of the chain to a stable solution:

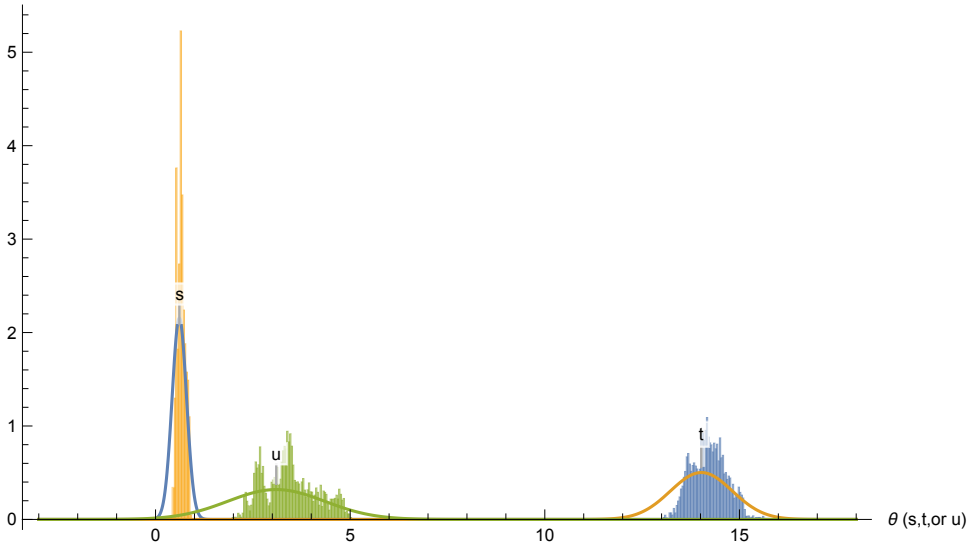
```
In[ ]:= SeedRandom[12349];
θM = Map[RandomReal, pdfFList];
particleM = Join[{logLikelihood[θM]}, θM];
sim = NestList[mcmcfun[#, priorSupport] &, particleM, 600000];
simGood = Take[sim, {100000, Length[sim]}];
{log, smcm, tmcm, umcm, σmcm} = Transpose[simGood];
```

```
In[ ]:= {log, smcm, tmcm, umcm, σmcm} = Transpose[simGood];
```

The MCMC histograms for (s, t, u) are compared to the least squares refinement solution here:

In[ ]:=
```
gmcm = Histogram[{smcm, tmcm, umcm}, {-3, 18, σs / 5}, ···];
Show[gmcm, gLS, ImageSize → 500]
```

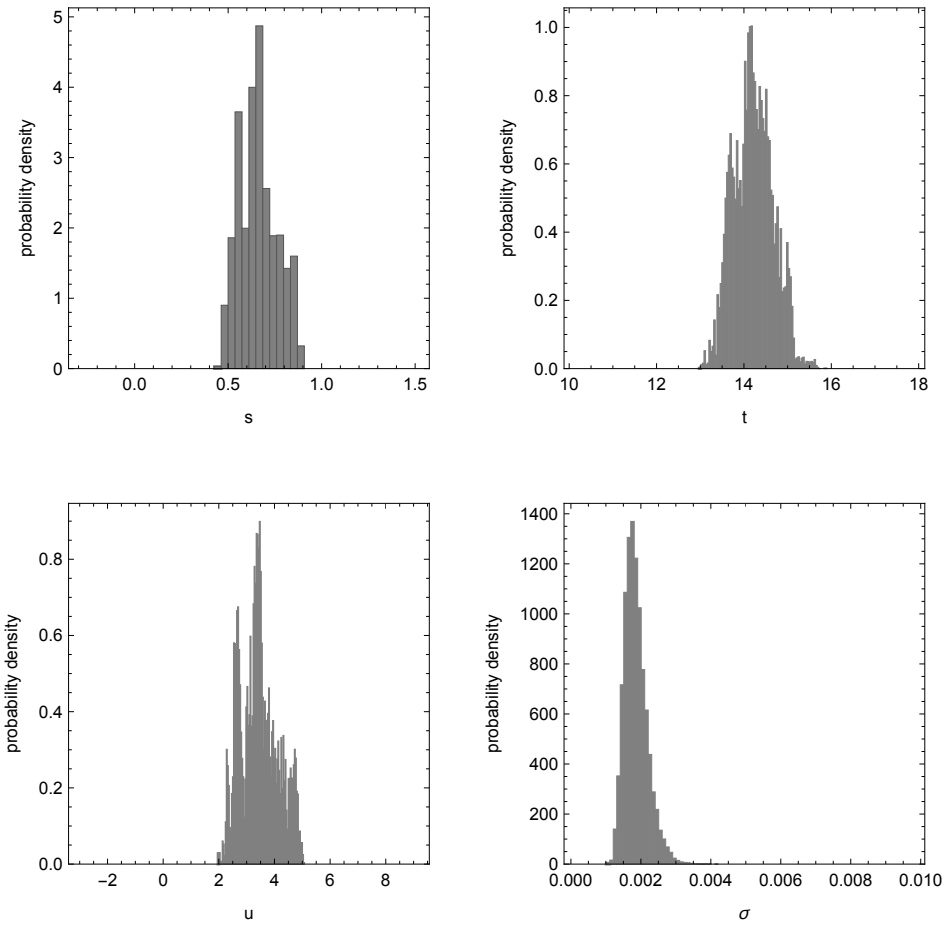
Out[ ]=



A more detailed look at the histograms including the σ histogram is shown here:

In[ ]:=
```
gs = Histogram[smcm, {sLS - 5 σs, sLS + 5 σs, σs / 5}, ···];
gt = Histogram[tmcm, {tLS - 5 σt, tLS + 5 σt, σs / 5}, ···];
gu = Histogram[umcm, {uLS - 5 σu, uLS + 5 σu, σs / 5}, ···];
gσ = Histogram[σmcm, {0.00001, 0.01, 0.0001}, ···];
GraphicsGrid[{{gs, gt}, {gu, gσ}}, ImageSize → 500]
```

Out[ ]=



As summation of outputs is:

In[ ]:=
```
{sMCMC, tMCMC, uMCMC, σMCMC} = Map[Mean, {smcm, tmcm, umcm, σmcm}];
TableForm[{{strue, ttrue, utrue, σtrue}, {sML, tML, uML, "NA"},
  {sLS, tLS, uLS, σ}, {sMCMC, tMCMC, uMCMC, σMCMC}}, ··· → ···]
```

Out[ ]//TableForm=

	s	t	u	σ
True	0.718	14.4	3.85	0.002
Max Likelihood	0.611129	14.0234	3.10013	NA
Least Squares	0.611129	14.0234	3.10013	0.00337605
Metropolis MCMC	0.662024	14.2141	3.44713	0.00185835

References

Brewer, Brendon J. (2018), "Bayesian Inference and Computation: A Beginner's Guide", published in *Bayesian Astrophysics*, Volume XXVI, Cambridge University Press.

Clifford, A. A. (1973), Multivariate Error Analysis, *Applied Science Publishers*, London.