ML estimation: TMB continued

FPA2020 Lecture 08: TMB for Random-effect

Toshihide Kitakado

June 17, 2020

Contents


```
library(ggplot2)
library(TMB)
```

We shall continue exercises for the use of TMB. In the last lecture, we dealt with just some simple likelihood estimation. Today, we shall cover likelihood analyses with random effects.

1 Analysis (1): IID Normal distribution

1.1 Model description and generation of example data

The assumption is like this. A total n IID observations are obtained from each of K strata (say, $k = 1, 2, .., K$) as follows:

$$Y_{k1}, Y_{k2}, \dots, Y_{kn} \sim (iid) N(\mu_k, \sigma^2) \quad (k = 1, 2, ..., K)$$

Parameter of interest here are $\mu_k (k = 1, 2, ..., K)$.

```
set.seed(2020)
Ks <- 20
ns <- 5
# True parameters
# Scenario 1 from U(0,10)
# Scenario 2 from N(5,2^2)
#mu.true <- runif(Ks,0,10)
mu.true <- rnorm(Ks,5,1)
sigma.true <- 2

yobs <- array(NA,c(Ks,ns))
for(k in 1:Ks) yobs[k,] <- rnorm(ns,mu.true[k],sigma.true)

colnames(yobs) <- 1:ns
```

```
cbind(Stratum=1:Ks, round(yobs,2)) %>%
  kable("latex", booktabs=T, caption="Data") %>%
  kable_styling(font_size=9,latex_options=c("hold_position"))
```

Table 1: Data

Stratum	1	2	3	4	5
1	9.73	7.57	6.01	5.23	7.05
2	5.70	7.90	7.17	5.01	5.52
3	2.28	2.41	6.09	8.77	4.68
4	4.45	3.30	4.02	2.75	4.76
5	4.02	1.19	1.60	0.75	-0.16
6	6.23	4.98	5.76	7.04	6.70
7	5.56	7.14	4.59	6.89	6.18
8	5.01	4.40	2.11	3.64	5.93
9	10.58	7.26	3.56	13.16	8.67
10	5.85	6.96	4.71	5.30	5.45
11	5.74	7.44	0.71	3.51	2.34
12	4.50	2.35	4.46	6.00	6.40
13	7.45	6.15	10.82	6.55	4.15
14	9.18	0.85	6.82	8.32	5.67
15	1.81	1.00	5.85	7.37	4.44
16	5.24	7.50	8.16	5.74	5.45
17	3.25	4.72	5.53	7.47	8.20
18	0.10	1.28	5.05	2.49	2.56
19	2.07	5.58	4.72	-0.73	3.31
20	3.41	2.66	2.51	4.81	9.39

The ML estimators for μ's are exactly given by the sample mean in each stratum.

$$\hat{\mu}_k = \bar{y}_{k\cdot} = \frac{1}{n}\sum_{i=1}^{n} y_{ki} \quad (k = 1, 2, ..., K)$$

We shall call this model a "Fixed-effect (FE) model".

Next we define a "random-effect (RE) model". We can assume the following latent structure for random effects μ's as follows:

$$Y_{k1}, Y_{k2}, \dots, Y_{kn} | \mu_k \sim (iid) N(\mu_k, \sigma^2)$$
$$\mu_1, \mu_2, ..., \mu_K \sim (iid) N(\theta, \tau^2).$$

In this RE model, although an explicit expression for the marginal distribution of the observation can be available, we here leave an integral form of the marginal distribution of the observation $Y'_{ik}s$ as follows:

$$L(\theta, \sigma^2, \tau^2) = \int \cdots \int \prod_k \left[\left\{ \prod_i f(y_{ki}|\mu_k, \sigma^2) \right\} f(\mu_k|\theta, \tau^2) \right] d\mu_1 \cdots d\mu_K.$$

So we will optimize the likelihood function with integration using the Laplace approximation (see Appendix). In this model, a set of variables, $\mu_1, \mu_2, ..., \mu_K$, plays a role of latent variable vector u.

1.2 TMB coding

```
sink(file="normal_re.cpp")
cat("
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator() ()
```

```
{
 DATA_INTEGER(Ks);
 DATA_INTEGER(ns);
 DATA_MATRIX(yobs);
 PARAMETER_VECTOR(mu);
 PARAMETER(logSigma);
 PARAMETER(theta);
 PARAMETER(logTau);
 Type sigma = exp(logSigma);
 Type tau = exp(logTau);
 ADREPORT(sigma);
 ADREPORT(tau);

 Type f=0.0;

 //likelihood from observation yobs (let me use simple coding)
 for(int k=0;k<Ks;k++){
  for(int i=0;i<ns;i++){
   f += (-1.0)*dnorm(yobs(k,i),mu(k),sigma,true);
  }
 }

 //likelihood of random effects
 f += (-1.0)*sum(dnorm(mu,theta,tau,true));

 return f;
}
", fill=TRUE)
sink()
```

1.3 Compiling "cpp" file and call "Dynamic link"

```
compile("normal_re.cpp")
```

```
[1] 0
```

```
dyn.load(dynlib("normal_re"))
```

1.4 Minimization etc.

- "MakeADFun": Constructing objective functions with derivatives based on a compiled C++ template
- "nlminb": Minimization
- "sdreport": extract the results

```
data <- list(Ks=Ks, ns=ns, yobs=yobs)
parameters <- list(mu=rep(0,Ks),logSigma=0,theta=0,logTau=0)
model <- MakeADFun(data, parameters, random="mu", DLL="normal_re")
fit <- nlminb(model$par, model$fn, model$gr)
```

```
Optimizing tape... Done
iter: 1  value: 531.5883 mgc: 43.23235 ustep: 1
iter: 2  mgc: 7.993606e-15
iter: 1  mgc: 7.993606e-15
Matching hessian patterns... Done
outer mgc:  387.4819
iter: 1  value: 293.1095 mgc: 0.2544483 ustep: 1
iter: 2  mgc: 7.21645e-16
iter: 1  mgc: 7.21645e-16
outer mgc:  75.56846
iter: 1  value: 305.9279 mgc: 4.733696 ustep: 1
iter: 2  mgc: 3.330669e-15
iter: 1  value: 274.8912 mgc: 1.012086 ustep: 1
iter: 2  mgc: 1.554312e-15
iter: 1  mgc: 1.554312e-15
outer mgc:  36.48937
iter: 1  value: 270.4047 mgc: 0.0777322 ustep: 1
iter: 2  mgc: 6.661338e-16
iter: 1  mgc: 6.661338e-16
outer mgc:  13.26888
iter: 1  value: 265.7147 mgc: 0.4770983 ustep: 1
iter: 2  mgc: 7.771561e-16
iter: 1  mgc: 7.771561e-16
outer mgc:  20.79594
iter: 1  value: 265.0355 mgc: 0.08340726 ustep: 1
iter: 2  mgc: 5.551115e-16
iter: 1  value: 263.0416 mgc: 0.02001583 ustep: 1
iter: 2  mgc: 7.21645e-16
iter: 1  value: 252.0726 mgc: 0.269541 ustep: 1
iter: 2  mgc: 7.910339e-16
iter: 1  mgc: 7.910339e-16
outer mgc:  5.46507
iter: 1  value: 936.2442 mgc: 187.3016 ustep: 1
iter: 2  mgc: 7.21645e-14
iter: 1  value: 246.7648 mgc: 0.4231157 ustep: 1
iter: 2  mgc: 5.551115e-16
iter: 1  mgc: 5.551115e-16
outer mgc:  6.476475
iter: 1  value: 243.6653 mgc: 0.5868904 ustep: 1
iter: 2  mgc: 9.436896e-16
iter: 1  mgc: 9.436896e-16
outer mgc:  6.326165
iter: 1  value: 244.3696 mgc: 0.4258873 ustep: 1
iter: 2  mgc: 5.967449e-16
iter: 1  mgc: 5.967449e-16
outer mgc:  10.85891
iter: 1  value: 243.4602 mgc: 0.2518584 ustep: 1
iter: 2  mgc: 7.077672e-16
iter: 1  mgc: 7.077672e-16
outer mgc:  3.356424
```

```
iter: 1  value: 242.7531 mgc: 0.1629781 ustep: 1
iter: 2  mgc: 1.110223e-15
iter: 1  mgc: 1.110223e-15
outer mgc:  1.90133
iter: 1  value: 242.1932 mgc: 0.2029594 ustep: 1
iter: 2  mgc: 7.771561e-16
iter: 1  mgc: 7.771561e-16
outer mgc:  0.3536151
iter: 1  value: 242.102 mgc: 0.07166562 ustep: 1
iter: 2  mgc: 1.026956e-15
iter: 1  mgc: 1.026956e-15
outer mgc:  0.04056468
iter: 1  value: 242.106 mgc: 0.002495816 ustep: 1
iter: 2  mgc: 7.771561e-16
iter: 1  mgc: 7.771561e-16
outer mgc:  0.005654033
iter: 1  value: 242.1055 mgc: 0.0001412438 ustep: 1
iter: 2  mgc: 6.661338e-16
iter: 1  mgc: 6.661338e-16
outer mgc:  0.0001003698
iter: 1  value: 242.1055 mgc: 4.588619e-06 ustep: 1
iter: 2  mgc: 7.771561e-16
iter: 1  mgc: 7.771561e-16
outer mgc:  1.485505e-06
iter: 1  mgc: 7.771561e-16
```

```
rep <- sdreport(model)
```

```
iter: 1  mgc: 7.771561e-16
outer mgc:  1.485505e-06
iter: 1  value: 242.1194 mgc: 0.002506295 ustep: 1
iter: 2  mgc: 7.979728e-16
outer mgc:  0.1636533
iter: 1  value: 242.0917 mgc: 0.002511313 ustep: 1
iter: 2  mgc: 7.771561e-16
outer mgc:  0.1639569
iter: 1  value: 242.1055 mgc: 0.0004987753 ustep: 1
iter: 2  mgc: 8.049117e-16
outer mgc:  0.006898426
iter: 1  value: 242.1055 mgc: 0.0004987753 ustep: 1
iter: 2  mgc: 7.771561e-16
outer mgc:  0.006899227
iter: 1  value: 242.1116 mgc: 0.002506295 ustep: 1
iter: 2  mgc: 6.661338e-16
outer mgc:  0.01913545
iter: 1  value: 242.0993 mgc: 0.002511313 ustep: 1
iter: 2  mgc: 6.661338e-16
outer mgc:  0.01912678
outer mgc:  2.114395
```

1.5 Results

You can see a shrinkage effect in the RE model; the estimates under the FE model are shrunken toward the grand mean. This causes a slight bias in the estimation, but it provides better precision as known as "Stein effect", where a concept of "borrowing the strength" is implemented.

```
mu.est.RE <- as.list(rep, "Est")$mu; round(mu.est.RE,3)
```

```
 [1] 6.501 5.908 4.931 4.246 2.604 5.826 5.778 4.496 7.558 5.490 4.309 4.859
[13] 6.437 5.844 4.410 6.016 5.613 3.169 3.647 4.729
```

```
mu.sd.RE <- as.list(rep, "Std")$mu
mu.est.FE <- apply(yobs,1,mean); round(mu.est.FE,3)
```

```
 [1] 7.118 6.260 4.847 3.857 1.482 6.142 6.073 4.218 8.646 5.656 3.948 4.743
[13] 7.025 6.167 4.094 6.417 5.834 2.299 2.990 4.555
```

```
grand.mean <- mean(yobs)

plot(mu.true, pch="*", ylim=c(0,10), xlab="Stratum", ylab=TeX("Value of $\\mu$"))
points(mu.est.RE, col="red", pch=19)
points(mu.est.FE, col="blue", pch=19)
abline(grand.mean,0,lty=2)
for(k in 1:Ks)arrows(k, mu.est.FE[k], k, mu.est.RE[k],length=0.1)
legend(1,3, pch=19, col=c("black","red","blue"),
       legend=c("True","RE","FE"),bty="n")
```

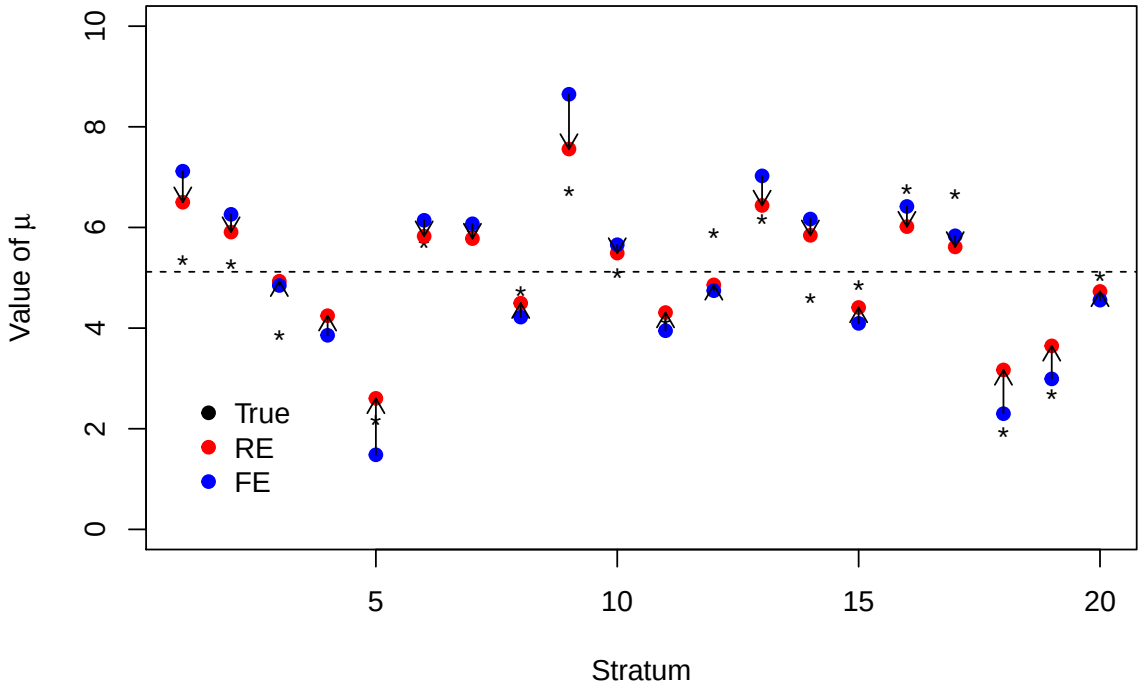

Figure 1: Comparison of estimates between FE and RE models (shrinkage effects).

Appendix: Laplace approximation

Here in general y, u and θ are vectors of the observation, latent variables (random effects) and other parameters, respectively, and let $f(y,u|\theta)$ be a joint distribution as $f(y|u,\theta)$ and $f(u|\theta)$

$$L(\theta) = \int f(y|u,\theta) f(u|\theta) d\mu = \int f(y,u|\theta) du = \int e^{-g(u,\theta)} du$$

where

$$g(u,\theta) = -\log f(y,u|\theta).$$

The Laplace approximation is often used in statistics. The approximation depends on a second-order Taylor expansion of $g(u,\theta)$. Let $\hat{u}(\theta)$ be the vector of u that minimizes $g(u,\theta)$ for fixed θ, that is

$$\hat{u}(\theta) = \arg\min_u g(u,\theta)$$

Then, $L(\theta)$ can be approximated (ignoring factors not depending on θ) as follows:

$$\begin{aligned} L(\theta) &= \int e^{-g(u,\theta)} du \\ &\approx \int \exp\left\{ -g(\hat{u}(\theta),\theta) - \frac{1}{2}(u-\hat{u}(\theta))' H(\theta)(u-\hat{u}(\theta)) \right\} du \\ &= (2\pi)^{n/2} \det\{H(\theta)\}^{-1/2} \exp\{-g(\hat{u}(\theta),\theta)\} \\ &= L^*(\theta), \end{aligned} ,$$

where

$$H(\theta) = \frac{\partial^2}{\partial u \partial u'} g(u,\theta)|_{u=\hat{u}(\theta)},$$

and $\det\{H(\theta)\}$ denotes the determinant of $H(\theta)$. Taking the logarithm of $L^*(\theta)$, we get

$$\log L^*(\theta) = Const + \frac{1}{2}\log\det\{H(\theta)\} - g(\hat{u}(\theta),\theta).$$

Once you get

$$\hat{\theta} = \arg\max_u \log L^*(\theta)$$

then you can also get esetimates for the latent variables as

$$\hat{u} = \hat{u}(\hat{\theta}).$$

Note that the third term on the right-hand side is the profile log-likelihood of θ. In fact, the equation above can be regarded as an approximate modified profile log-likelihood of θ to reduce the estimation bias (Lindsey 1996, Pawitan 2001).