Full Information Maximum Likelihood

Stats Camp 2018: Missing Data Analysis

Kyle M. Lang

Department of Methodology & Statistics
Tilburg University

19–21 October 2018

Outline

- Look at FIML in more depth
  - Review of maximum likelihood (ML)
  - Show how to do ML and FIML manually in R.

FIML Conceptual Refresher

FIML is an ML estimation method that is robust to ignorable nonresponse.

- FIML partitions the missing information out of the likelihood function so that the model is only estimated from the observed parts of the data.

After a minor alteration to the likelihood function, FIML reduces to simple ML estimation.

- So, let’s review ML estimation before moving forward.

Maximum Likelihood Estimation Refresher

ML estimation simply finds the parameter values that are “most likely” to have given rise to the observed data.

- The *likelihood* function is just a probability density (or mass) function with the data treated as fixed and the parameters treated as random variables.
- Having such a framework allows us to ask: “Given that I’ve observed these data values, what parameter values most probably describe these data?”

Maximum Likelihood Estimation Refresher

ML estimation is usually employed when there is no closed form solution for the parameters we seek.

- This is why you don't usually see ML used to fit general linear models.

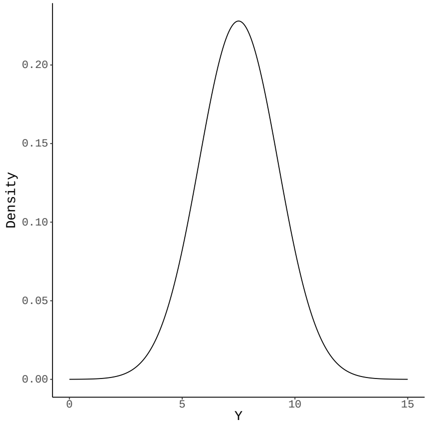
After choosing a likelihood function, we iteratively optimize the function to produce the ML estimated parameters.

- In practice, we nearly always work with the natural logarithm of the likelihood function (i.e., the *loglikelihood*).

Likelihoods

Suppose we have the following model:

$$Y \sim \mathrm{N}\left(\mu, \sigma^2\right).$$



Likelihoods

For a given Y_n, we have:

$$P\left(Y_n|\mu,\sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(Y_n-\mu)^2}{2\sigma^2}}. \tag{1}$$

If we plug estimated parameters into Equation 1, we get the probability of observing Y_n given $\hat{\mu}$ and $\hat{\sigma}^2$:

$$P\left(Y_n|\hat{\mu},\hat{\sigma}^2\right) = \frac{1}{\sqrt{2\pi\hat{\sigma}^2}}e^{-\frac{(Y_n-\hat{\mu})^2}{2\hat{\sigma}^2}}. \tag{2}$$

Applying Equation 2 to all N observations and multiplying the results produces a *likelihood*:

$$\hat{L}\left(\hat{\mu},\hat{\sigma}^2\right) = \prod_{n=1}^{N} P\left(Y_n|\hat{\mu},\hat{\sigma}^2\right).$$

Likelihoods

We generally want to work with the natural logarithm of Equation 2. Doing so gives the *loglikelihood*:

$$\begin{aligned}\hat{\mathcal{L}}\left(\hat{\mu}, \hat{\sigma}^2\right) &= \ln \prod_{n=1}^{N} P\left(Y_n | \hat{\mu}, \hat{\sigma}^2\right) \\ &= -\frac{N}{2}\ln 2\pi - N \ln \hat{\sigma} - \frac{1}{2\hat{\sigma}^2}\sum_{n=1}^{N}\left(Y_n - \hat{\mu}\right)^2\end{aligned}$$

ML tries to find the values of $\hat{\mu}$ and $\hat{\sigma}^2$ that maximize $\hat{\mathcal{L}}\left(\hat{\mu}, \hat{\sigma}^2\right)$.

- Find the values of $\hat{\mu}$ and $\hat{\sigma}^2$ that are *most likely*, given the observed values of Y.

Likelihoods

Suppose we have a linear regression model:

$$Y = \beta_0 + \beta_1 X + \varepsilon,$$

$$\varepsilon \sim \mathrm{N}\left(0, \sigma^2\right) .$$

This model can be equivalently written as:

$$Y \sim \mathrm{N}\left(\beta_0 + \beta_1 X, \sigma^2\right)$$

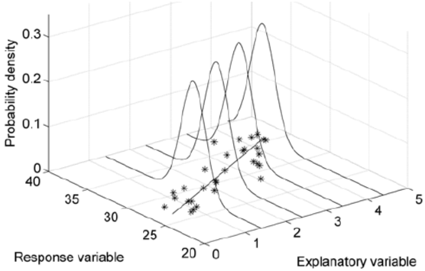


Image retrieved from: `http://www.seaturtle.org/mtn/archives/mtn122/mtn122p1.shtml`

Likelihoods

For a given $\{Y_n, X_n\}$, we have:

$$P\left(Y_n | X_n, \beta_0, \beta_1, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(Y_n - \beta_0 - \beta_1 X_n)^2}{2\sigma^2}}. \tag{3}$$

If we plug our estimated parameters into Equation 3, we get the probability of observing Y_n given $\hat{Y}_n = \hat{\beta}_0 + \hat{\beta}_1 X_n$ and $\hat{\sigma}^2$.

$$P\left(Y_n | X_n, \hat{\beta}_0, \hat{\beta}_1, \hat{\sigma}^2\right) = \frac{1}{\sqrt{2\pi\hat{\sigma}^2}} e^{-\frac{\left(Y_n - \hat{\beta}_0 - \hat{\beta}_1 X_n\right)^2}{2\hat{\sigma}^2}} \tag{4}$$

Likelihoods

So, our final loglikelihood function would be the following:

$$\begin{aligned}\hat{\mathcal{L}}\left(\hat{\beta}_0, \hat{\beta}_1, \hat{\sigma}^2\right) &= \ln \prod_{n=1}^{N} P\left(Y_n | X_n, \hat{\beta}_0, \hat{\beta}_1, \hat{\sigma}^2\right) \\ &= -\frac{N}{2} \ln 2\pi - N \ln \hat{\sigma} - \frac{1}{2\hat{\sigma}^2} \sum_{n=1}^{N} \left(Y_n - \hat{\beta}_0 - \hat{\beta}_1 X_n\right)^2 .\end{aligned}$$

Example

```r
## Fit a model:
out1 <- lm(ldl ~ bp + glu + bmi, data = diabetes)

## Extract the predicted values and estimated residual
## standard error:
yHat <- predict(out1)
s    <- summary(out1)$sigma

## Compute the row-wise probabilities:
pY <- dnorm(diabetes$ldl, mean = yHat, sd = s)

## Compute the loglikelihood, and compare to R's version:
sum(log(pY)); logLik(out1)[1]

## [1] -2109.939
## [1] -2109.93
```

Multivariate Normal Distribution

The PDF for the multivariate normal distribution is:

$$P(\mathbf{Y}|\boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^P |\Sigma|}} e^{-\frac{1}{2}(\mathbf{Y}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{Y}-\boldsymbol{\mu})}.$$

So, the multivariate normal loglikelihood is:

$$\mathcal{L}(\boldsymbol{\mu}, \Sigma) = -\left[\frac{P}{2}\ln(2\pi) + \frac{1}{2}\ln|\Sigma| + \frac{1}{2}\right] \sum_{n=1}^{N} (\mathbf{Y}_n - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{Y}_n - \boldsymbol{\mu}).$$

Which can be further simplified if we multiply through by -2:

$$-2\mathcal{L}(\boldsymbol{\mu}, \Sigma) = [P\ln(2\pi) + \ln|\Sigma|] \sum_{n=1}^{N} (\mathbf{Y}_n - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{Y}_n - \boldsymbol{\mu}).$$

Steps of ML

1. Choose a probability distribution, $f(Y|\theta)$, to describe the distribution of the data, Y, given the parameters, θ.
2. Choose some estimate of θ, $\hat{\theta}^{(i)}$.
3. Compute each row's contribution to the loglikelihood function by evaluating: $\ln\left[f\left(Y_n|\hat{\theta}^{(i)}\right)\right]$.
4. Sum the individual loglikelihood contributions from Step 3 to find the loglikelihood value, $\hat{\mathcal{L}}$.
5. Choose a "better" estimate of the parameters, $\hat{\theta}^{(i+1)}$, and repeat Steps 3 and 4.
6. Repeat Steps 3 – 5 until the change between $LL^{(i-1)}$ and $LL^{(i)}$ falls below some trivially small threshold.
7. Take $\hat{\theta}^{(i)}$ as your estimated parameters.

ML Example

Recall the *n*th observation's contribution to the multivariate normal loglikelihood function:

$$\mathcal{L}\left(\boldsymbol{\mu}, \Sigma\right)_n = -\frac{P}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma| - \frac{1}{2}(\mathbf{Y}_n - \boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{Y}_n - \boldsymbol{\mu}).$$

It turns out that this function is readily available in R via the **mvtnorm** package:

```
## Vector of row-wise contributions to the overall LL:
ll0 <- dmvnorm(y, mean = mu, sigma = sigma, log = TRUE)
```

ML Example

We can wrap the preceding code in a nice R function:

```r
## Complete data loglikelihood function:
ll <- function(par, data) {
    ## Extract the parameter matrices:
    p  <- ncol(data)
    mu <- par[1 : p]

    ## Populate sigma from its Cholesky factor:
    sigma <-vecChol(par[-c(1 : p)], revert = TRUE)

    ## Compute the row-wise contributions to the LL:
    ll0 <-
        dmvnorm(data, mean = mu, sigma = sigma, log = TRUE)

    sum(ll0)# return the overall LL value
}
```

ML Example

We'll also need the helper functions:

```r
## Convert from covariance matrix to vectorized Cholesky
## factor and back:
vecChol <- function(x, revert = FALSE) {
    if(revert) {
        tmp                <- matrix(0, nV(x), nV(x))
        tmp[!lower.tri(tmp)] <- x
        crossprod(tmp)
    }
    else
        chol(x)[!lower.tri(x)]
}

## Find the number of variables given a vector of unique
## variances/covariances:
nV <- function(x) (-1 + sqrt(1 + 8 * length(x))) / 2
```

ML Example

The **optimx** package can numerically optimize arbitrary functions.

- We can use it to (semi)manually implement ML.

```r
## Subset the 'diabetes' data:
dat1 <- as.matrix(diabetes[ , c("bmi", "ldl", "glu")])

## Choose some starting values:
m0   <- rep(0, 3)
s0   <- vecChol(diag(3))
par0 <- c(m0, s0)

## Use optimx() to numerically optimize the LL function:
mle <- optimx(par     = par0,
              fn      = ll,
              data    = dat1,
              method  = "BFGS",
              control = list(maximize = TRUE, maxit = 1000)
              )

## Maximizing -- use negfn and neggr
```

ML Example

Finally, let's check convergence and extract the optimized parameters:

```
## Check convergence:
mle[c("convcode", "kkt1", "kkt2")]

##      convcode kkt1 kkt2
## BFGS        0 TRUE TRUE

## Get the optimize mean vector and covariance matrix:
muHat    <- mle[1 : 3]
sigmaHat <- vecChol(as.numeric(mle[4 : 9]), revert = TRUE)
```

ML Example

	bmi	ldl	glu
Max. Like.	26.376	115.437	91.260
Closed Form	26.376	115.439	91.260

Estimated Means

	bmi	ldl	glu
bmi	19.476	35.013	19.697
ldl	35.013	922.820	101.373
glu	19.697	101.373	131.864

ML Covariance Matrix

	bmi	ldl	glu
bmi	19.520	35.093	19.742
ldl	35.093	924.955	101.605
glu	19.742	101.605	132.166

Closed Form Covariance Matrix

From ML to FIML

The *n*th observation's contribution to the multivariate normal loglikelihood function would be the following:

$$\mathcal{L}\,(\boldsymbol{\mu}, \Sigma)_n = -\frac{P}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma| - \frac{1}{2}(\mathbf{Y}_n - \boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{Y}_n - \boldsymbol{\mu}). \qquad (5)$$

From ML to FIML

The *n*th observation's contribution to the multivariate normal loglikelihood function would be the following:

$$\mathcal{L}\left(\boldsymbol{\mu}, \Sigma\right)_n = -\frac{P}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma| - \frac{1}{2}(\mathbf{Y}_n - \boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{Y}_n - \boldsymbol{\mu}). \tag{5}$$

FIML just tweaks Equation 5 a tiny bit:

$$\mathcal{L}\left(\boldsymbol{\mu}, \Sigma\right)_{fiml,n} = -\frac{P}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma_q| - \frac{1}{2}(\mathbf{Y}_n - \boldsymbol{\mu}_q)^T\Sigma_q^{-1}(\mathbf{Y}_n - \boldsymbol{\mu}_q).$$

Where $q = 1, 2, \ldots, Q$ indexes response patterns.

FIML Example

First things first, we need to punch some holes in our example data.

```r
## Impose MAR missing data:
dat2 <-
    imposeMissData(dat       = dat1,
                   targets = list(mar = c("ldl", "glu")),
                   preds   = "bmi",
                   pm      = 0.3,
                   snr     = 2,
                   pattern = "low")$data
```

FIML Example

```r
## Compute the within-pattern contributions to the LL:
ll0 <- function(i, mu, sigma, pats, ind, data) {
    ## Find the current pattern:
    p1 <- pats[i, ]

    if(sum(p1) > 1) # More than one observed variable?
        dmvnorm(x     = data[ind == i, p1],
                mean  = mu[p1],
                sigma = sigma[p1, p1],
                log   = TRUE)
    else
        dnorm(x    = data[ind == i, p1],
              mean = mu[p1],
              sd   = sqrt(sigma[p1, p1]),
              log  = TRUE)
}
```

FIML Example

```r
## FIML loglikelihood function:
llm <- function(par, data, pats, ind) {
    ## Extract the parameter matrices:
    p  <- ncol(data)
    mu <- par[1 : p]

    ## Populate sigma from its cholesky factor:
    sigma <-vecChol(par[-c(1 : p)], revert = TRUE)

    ## Compute the pattern-wise contributions to the LL:
    ll1 <- sapply(X     = 1 : nrow(pats),
                  FUN   = ll0,
                  mu    = mu,
                  sigma = sigma,
                  pats  = pats,
                  ind   = ind,
                  data  = data)

    sum(unlist(ll1))
}
```

FIML Example

```
## Summarize response patterns:
pats <- uniquecombs(!is.na(dat2))
ind  <- attr(pats, "index")

## Choose some starting values:
m0   <- colMeans(dat2, na.rm = TRUE)
s0   <- vecChol(cov(dat2, use = "pairwise"))
par0 <- c(m0, s0)

## Use optimx() to numerically optimize the LL function:
mle2 <- optimx(par     = par0,
               fn      = llm,
               data    = dat2,
               pats    = pats,
               ind     = ind,
               method  = "BFGS",
               control = list(maximize = TRUE, maxit = 1000)
               )

## Maximizing -- use negfn and neggr
```

FIML Example

Check convergence and extract the optimized parameters:

```
## Check convergence:
mle2[c("convcode", "kkt1", "kkt2")]

##      convcode kkt1 kkt2
## BFGS        0 TRUE TRUE

## Get the optimize mean vector and covariance matrix:
muHat2.1    <- mle2[1 : 3]
sigmaHat2.1 <-
    vecChol(as.numeric(mle2[4 : 9]), revert = TRUE)
```

FIML Example

Just to make sure our results are plausible, we can do the same analysis using the `cfa` function from the **lavaan** package:

```
## Confirm the manual approach by using lavaan::cfa() to
## get the FIML estimates:
mod1 <- "
bmi ~~ ldl + glu
ldl ~~ glu
"

## Fit the model with lavaan::cfa():
out2 <- cfa(mod1, data = dat2, missing = "fiml")

muHat2.2     <- inspect(out2, "est")$nu
sigmaHat2.2 <- inspect(out2, "theta")
```

FIML Example

	bmi	ldl	glu
Manual	26.376	120.225	91.335
Lavaan	26.376	120.228	91.335

Estimated Means

	bmi	ldl	glu
bmi	19.477	11.172	18.792
ldl	11.172	873.123	78.640
glu	18.792	78.640	132.201

Manual FIML Covariance Matrix

	bmi	ldl	glu
bmi	19.476	11.193	18.794
ldl	11.193	873.145	78.690
glu	18.794	78.690	132.213

Lavaan FIML Covariance Matrix