ML & FIML Example

Utrecht University Winter School: Missing Data in R

Utrecht University

Kyle M. Lang

Department of Methodology & Statistics
Utrecht University

Outline

Maximum Likelihood

ML Example

Recall the nth 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(tail(par, -p), p = 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 following helper function:

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

ML Example

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

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

```
## Subset the 'diabetes' data:
dat1 <- readRDS(paste0(dataDir, "diabetes.rds")) %>%
    select(bmi, ldl, glu) %>%
    as.matrix()

## Choose some starting values:
m0   <- rep(0, 3)
s0   <- diag(3) %>% vecChol()
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 <- mle[4:9] %>% as.numeric() %>% vecChol(p = 3, revert = TRUE)
```

ML Example

	bmi	ldl	glu
ML	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

Full Information Maximum Likelihood

FIML Example

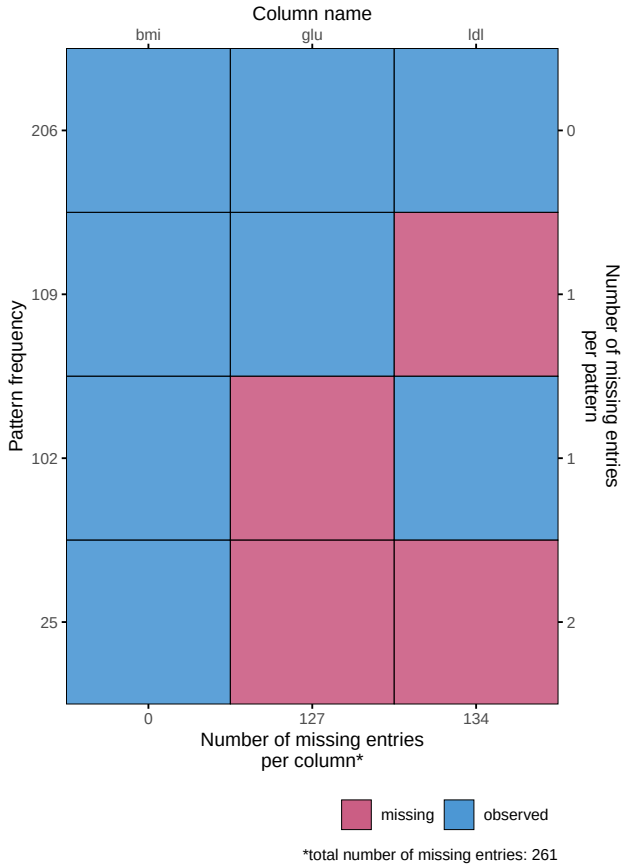
First things first, we need to punch some holes in our example data.

```
## Impose MAR missing:
dat2 <- imposeMissData(data    = dat1,
                       targets = c("ldl", "glu"),
                       preds   = "bmi",
                       pm      = 0.3,
                       types   = c("low", "high"),
                       stdDat  = TRUE)
```

Visualize the Response Patterns

The data contain 4 unique response patterns.

- We'll define 4 different version of μ and Σ.
- We'll calculate each individual loglikelihood contributions using the appropriate flavor of μ and Σ.



FIML Example

```
## Compute the within-pattern contributions to the LL:
ll0 <- function(i, mu, sigma, pats, ind, data) {
    ## Define the current response 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

```
## 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(tail(par, -p), p = 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   <- cov(dat2, use = "pairwise") %>% vecChol()
par0 <- c(m0, s0)

## Use optimx() to numerically optimize the LL function:
mle <- 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:
mle[c("convcode", "kkt1", "kkt2")]

     convcode kkt1 kkt2
BFGS        0 TRUE TRUE

## Get the optimize mean vector and covariance matrix:
muHat1    <- mle[1:3]
sigmaHat1 <- mle[4:9] %>% as.numeric() %>% vecChol(p = 3, 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:

```
## Define the model in lavaan syntax:
mod <- "
bmi ~~ ldl + glu
ldl ~~ glu
"

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

## Extract the estimated parameters:
muHat2    <- inspect(fit, "est")$nu
sigmaHat2 <- inspect(fit, "theta")
```

FIML Example

	bmi	ldl	glu
Manual	26.376	116.634	91.686
Lavaan	26.376	116.636	91.686

Estimated Means

	bmi	ldl	glu
bmi	19.475	24.249	22.835
ldl	24.249	889.032	120.618
glu	22.835	120.618	140.543

Manual FIML Covariance Matrix

	bmi	ldl	glu
bmi	19.476	24.260	22.837
ldl	24.260	889.068	120.633
glu	22.837	120.633	140.544

Lavaan FIML Covariance Matrix