`refitME`: Tutorial for fitting measurement error models using Monte Carlo Expectation Maximization in `R`

Jakub Stoklosa, Hwang W-H., & David Warton

30 November, 2020

This tutorial documents fitting an MCEM algorithm via the `refitME` R-package. For more specific details see: *refitME: Measurement Error Modelling using Monte Carlo Expectation Maximization in R.* Also, see `?refitME` for further details on the fitting function, input arguments and output.

Example 1: A simple GLM example taken from Carroll *et al.* (2006).

We use the Framingham heart study data set. In addition to the naive model, we also fit a `simex` model and compare it with MCEM. Computational times for both models are also reported.

Load data and `R`-packages.

```
suppressWarnings(suppressMessages(library(refitME)))
suppressWarnings(suppressMessages(library(simex)))

set.seed(2020)

B <- 100   # The number of Monte Carlo replication values/SIMEX simulations.

data(Framinghamdata)
```

Setup all variables (the construction below follows the Carroll et al. (2006) monograph).

```
W <- as.matrix(Framinghamdata$w1) # Matrix of error-contaminated covariate.
sigma.sq.u <- 0.01259/2 # ME variance, obtained from Carroll et al. (2006) monograph.
```

Fit the naive model.

The first stored variable `w1` is the error contaminated variable used in the analysis.

```
glm_naiv1 <- glm(Y ~ w1 + z1 + z2 + z3, x = TRUE, family = binomial,
                 data = Framinghamdata)
```

Fit the SIMEX model.

```r
start <- Sys.time()
glm_simex1 <- simex(glm_naiv1, SIMEXvariable = c("w1"),
                    measurement.error = cbind(sqrt(sigma.sq.u)), B = B) # SIMEX.
end <- Sys.time()
t1 <- difftime(end, start, units = "secs")
comp.time <- c(t1)
```

Fit the MCEM model.

```r
start <- Sys.time()
glm_MCEM1 <- refitME(glm_naiv1, sigma.sq.u, W, B)
```

```
## [1] "One specified error-contaminated covariate."
## [1] "convergence :-)"
## [1] 5
```

```r
end <- Sys.time()
t2 <- difftime(end, start, units = "secs")
comp.time <- c(comp.time, t2)
```

Report model estimates and compare computational times.

```r
est.beta <- rbind(coef(glm_naiv1), coef(glm_simex1), coef(glm_MCEM1))
est.beta.se <- rbind(sqrt(diag(vcov(glm_naiv1))),
                 sqrt(diag(glm_simex1$variance.jackknife)), sqrt(diag(vcov(glm_MCEM1))))
row.names(est.beta) = row.names(est.beta.se) <- c("Naive GLM", "SIMEX", "MCEM")
colnames(est.beta) = colnames(est.beta.se) <- c("(Intercept)", "SBP", "chol. level",
                                                "age", "smoke")
round(est.beta, digits = 3)
```

```
##           (Intercept)   SBP chol. level   age smoke
## Naive GLM     -14.951 1.707       0.008 0.055 0.592
## SIMEX         -15.919 1.947       0.008 0.053 0.598
## MCEM          -16.059 1.955       0.008 0.056 0.594
```

```r
round(est.beta.se, digits = 3)   # Standard error estimates.
```

```
##           (Intercept)   SBP chol. level   age smoke
## Naive GLM       1.900 0.418       0.002 0.012 0.250
## SIMEX           2.043 0.456       0.002 0.012 0.251
## MCEM            2.185 0.487       0.002 0.012 0.250
```

```
names(comp.time) <- c("SIMEX", "MCEM")
comp.time  # SIMEX and MCEM.
```

```
## Time differences in secs
##   SIMEX    MCEM
## 7.97810 4.91459
```

Example 2: A GAM example taken from Ganguli *et al.* (2005).

The Milan mortality air pollution data set. Here, we fit GAM models via the `mgcv` package where one covariate is error-contaminated.

Load data and R-packages.

```
suppressWarnings(suppressMessages(library(refitME)))

set.seed(2020)

epsilon <- 0.00001  # A set convergence threshold.
B <- 3  # The number of Monte Carlo replication values.

data(milan.mort)
dat.air <- milan.mort
```

Setup all variables.

```
Y <- dat.air[, 6]  # Mortality counts.
n <- length(Y)

z1 <- (dat.air[, 1])
z2 <- (dat.air[, 4])
z3 <- (dat.air[, 5])
w1 <- log(dat.air[, 9])
W <- as.matrix(w1)
dat <- data.frame(cbind(Y, z1, z2, z3, w1))

sigma.sq.u <- 0.0915 # This gives a reliability ratio of 0.7.
rel.rat <- round((1 - sigma.sq.u/var(dat$w1))*100, digits = 0)
```

Fit the naive model.

```
gam_naiv1 <- gam(Y ~ s(w1) + s(z1, k = 25) + s(z2) + s(z3), family = "poisson", data = dat)
```

Fit the MCEM model.

Note this will take a fair while to run (consider reducing B if you just want a rough answer).

```
gam_MCEM1 <- refitME(gam_naiv1, sigma.sq.u, W, B)
```

```
## [1] "One specified error-contaminated covariate."
## [1] "convergence :-)"
## [1] 7
```
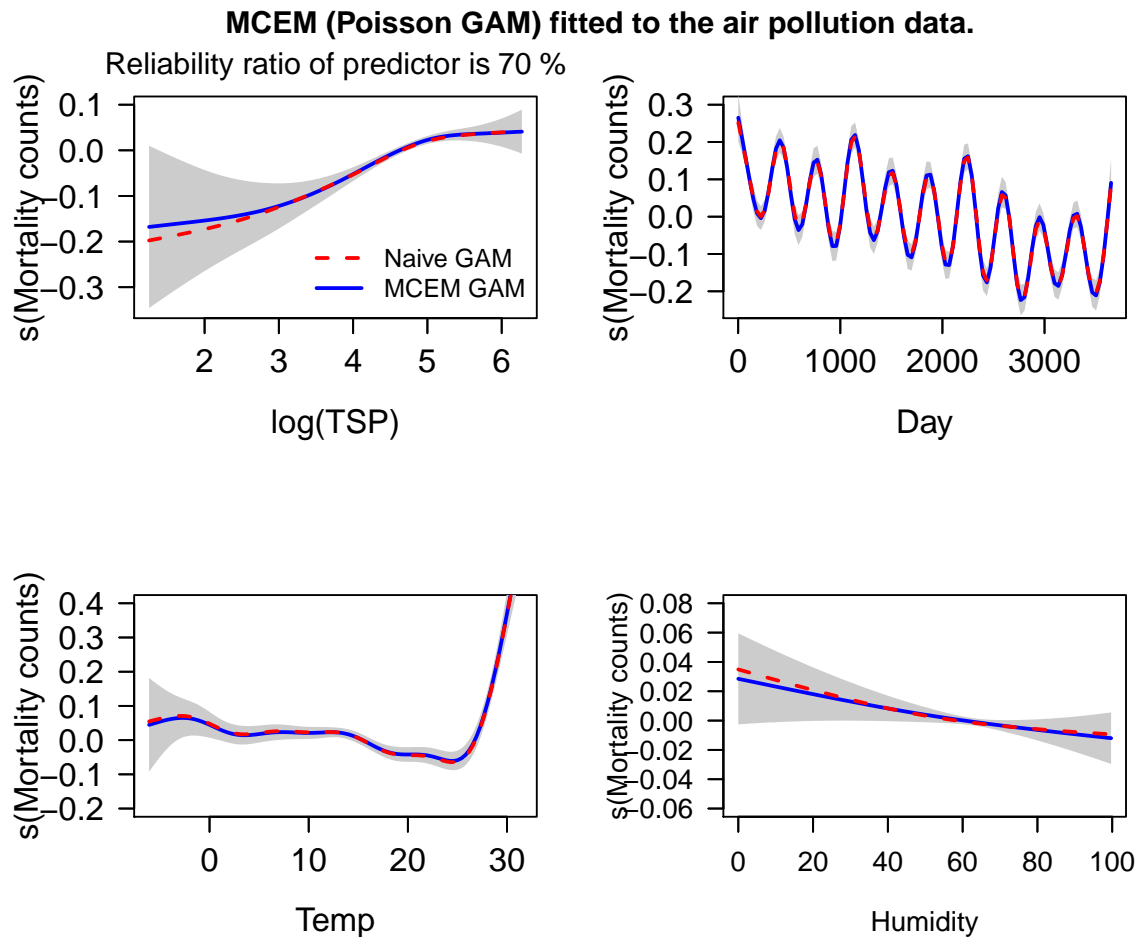

Figure 1: *Plots of smooths against covariate. TSP (top left is the error contaminated variable).*

Example 3: A point-process model using presence-only data

We use the *Corymbia eximia* presence-only data set from Renner and Warton (2013). Here, we fit a naive models point-process model (PPM) and an MCEM PPM.

Load data and `R`-packages.

```r
suppressWarnings(suppressMessages(library(refitME)))

data(Corymbiaeximiadata)

suppressMessages(attach(Corymbiaeximiadata))
```

Setup all variables.

```r
Y <- Corymbiaeximiadata$Y.obs

W <- Corymbiaeximiadata$MNT

p.wt <- rep(1.e-6, length(Corymbiaeximiadata$Y.obs))
p.wt[Corymbiaeximiadata$Y.obs == 0] <- 1

X <- cbind(rep(1, length(Y)), poly(W, degree = 2, raw = TRUE),
           poly(Rain, degree = 2, raw = TRUE),
             poly(sqrt(D.Main), degree = 2, raw = TRUE))

colnames(X) <- c("(Intercept)", "X1", "X2", "Z1", "Z2", "Z3", "Z4")

dat <- data.frame(cbind(Y, p.wt, X))
colnames(dat)[1:2] <- c("Y", "p.wt")
```

Fit the naive (PPM) model.

```r
PPM_naiv1 <- glm(Y/p.wt ~ X1 + X2 + Z1 + Z2 + Z3 + Z4, family = "poisson",
 weights = p.wt, data = dat)
```

Fit the MCEM PPM.

```
B <- 5

sigma.sq.u <- 0.25

PPM_MCEM1 <- refitME(PPM_naiv1, sigma.sq.u, W, B)
```

```
## [1] "One specified error-contaminated covariate."
## [1] "convergence :-)"
## [1] 6
```
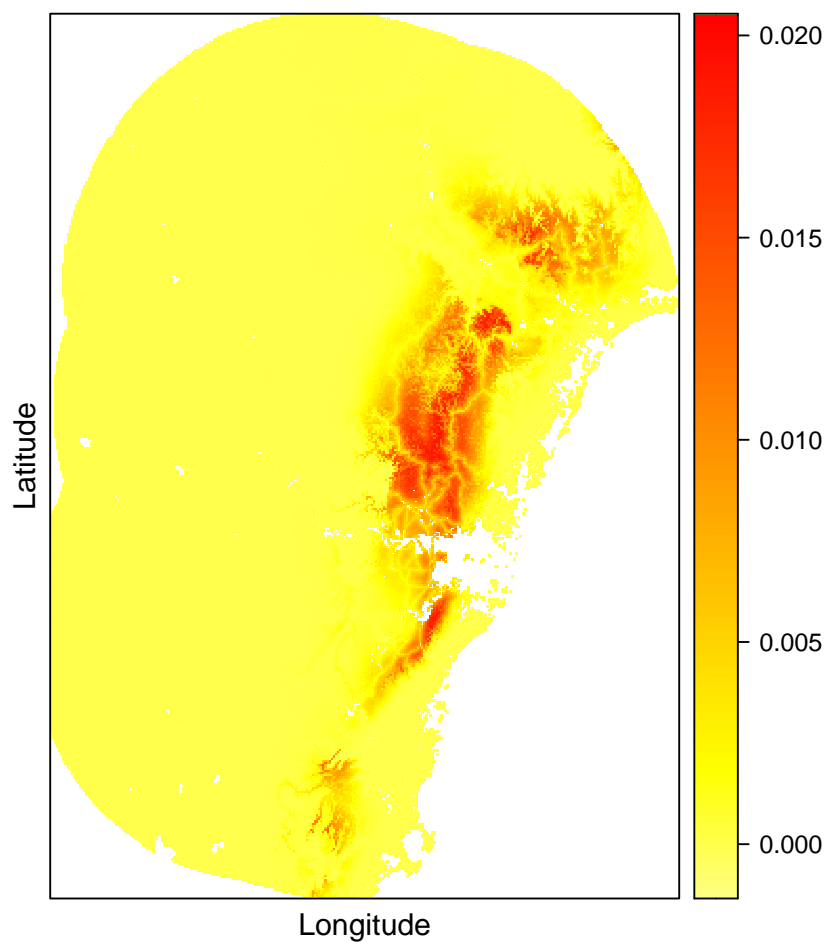

Figure 2: *Plots of predicted presences of* Corymbia eximia *using presence-only data and the max temperature covariate.*