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

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

23 April, 2021

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.

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

set.seed(2020)

data(Framinghamdata)
```

Fit the naive model.

The first stored variable `w1` is the error contaminated predictor as used in the original Carroll *et al.* (2006) analysis.

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

Setup measurement error (ME) variance and all tuning parameters.

```r
sigma.sq.u <- 0.01259/2 # ME variance, as obtained from Carroll et al. (2006).
B <- 100  # The number of Monte Carlo replication values/SIMEX simulations.
```

Fit the SIMEX model.

```
start <- Sys.time()
glm_simex <- simex(glm_naiv, 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.

```
start <- Sys.time()
glm_MCEM <- refitME(glm_naiv, sigma.sq.u, B)
```

```
## One specified error-contaminated predictor/covariate.

## convergence :-)

## [1] 5
```

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

Report model estimates and compare computational times.

```
est.beta <- rbind(coef(glm_naiv), coef(glm_simex), coef(glm_MCEM))
est.beta.se <- rbind(sqrt(diag(vcov(glm_naiv))),
                   sqrt(diag(glm_simex$variance.jackknife)), sqrt(diag(vcov(glm_MCEM))))
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
```

```
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
## 10.400497  5.307824
```

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

The Milan mortality air pollution data set is part of the `SemiPar` package. Here, we fit GAM models via the `mgcv` package where one predictor (daily total suspended particles measurements) is error-contaminated.

Load data and `R`-packages.

```
suppressWarnings(suppressMessages(library(refitME)))
suppressWarnings(suppressMessages(library(SemiPar)))
suppressWarnings(suppressMessages(library(mgcv)))

set.seed(2020)

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]) # The error-contaminated predictor (total suspended particles).
dat <- data.frame(cbind(Y, w1, z1, z2, z3))
```

Fit the naive model.

```
gam_naiv <- 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 if `B` is large (consider decreasing `B` if you want a rough answer).

```
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)

gam_MCEM1 <- refitME(gam_naiv, sigma.sq.u)
```

```
## One specified error-contaminated predictor/covariate.
```

```
## convergence :-)
```

```
## [1] 14
```

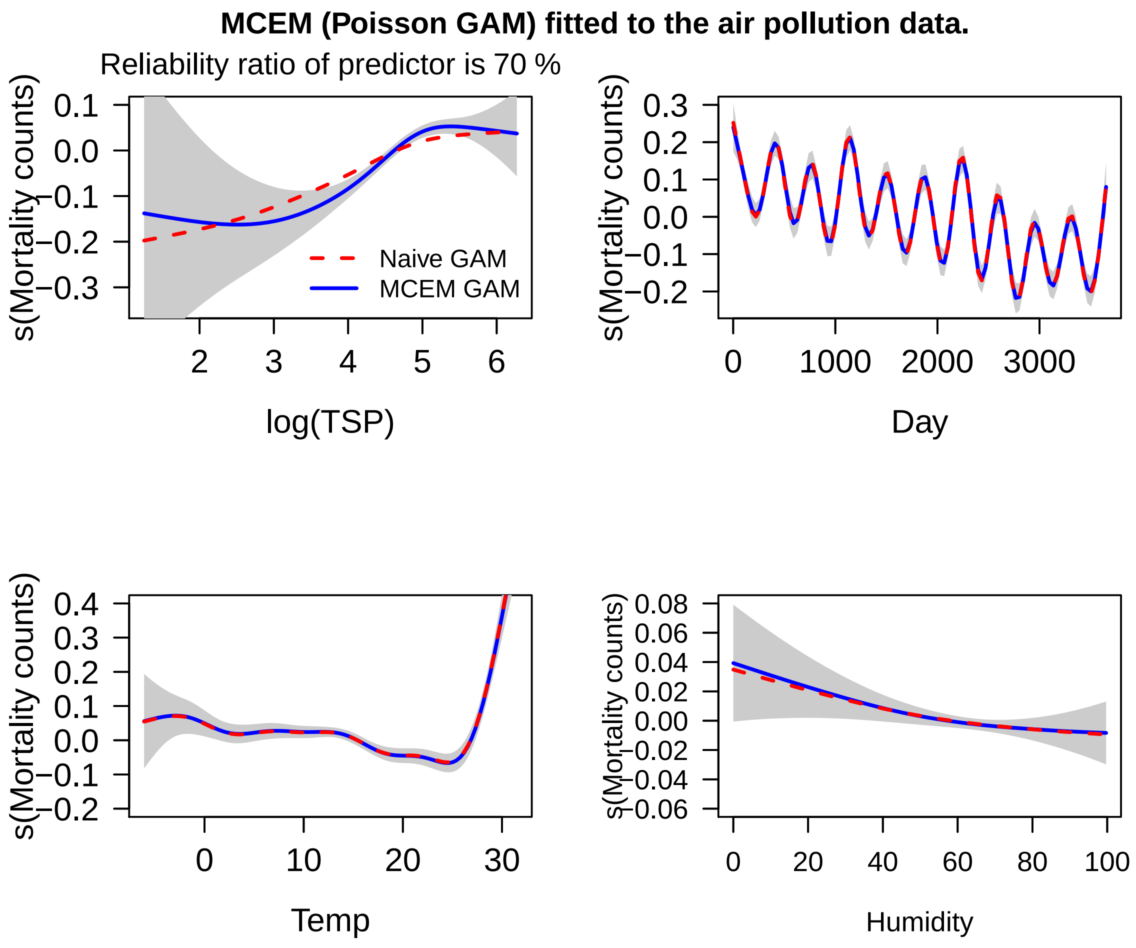

Figure 1: *Plots of smooths against each predictor. TSP (top left is the error contaminated variable).*

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.

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

data(Corymbiaeximiadata)

dat <- Corymbiaeximiadata
```

Setup all variables.

```
Y1 <- dat$Y.obs

Rain <- dat$Rain
D.Main <- dat$D.Main
MNT <- dat$MNT

p.wt <- rep(1.e-6, length(Y1))
p.wt[Y1 == 0] <- 1

Y <- Y1/p.wt

X <- cbind(rep(1, length(Y)), poly(MNT, 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")

dat1 <- data.frame(cbind(Y, p.wt, X))
colnames(dat1)[1:2] <- c("Y", "p.wt")
```

Fit the naive (PPM) model.

```
PPM_naiv1 <- glm(Y ~ X1 + X2 + Z1 + Z2 + Z3 + Z4, family = "poisson",
 weights = p.wt, data = dat1)
```

```
## Warning: glm.fit: fitted rates numerically 0 occurred
```

Fit the MCEM PPM.

```
sigma.sq.u <- 0.25

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

```
## One specified error-contaminated predictor/covariate.

## convergence :-)

## [1] 4

## Warning in limits.and.aspect(default.prepanel, prepanel = prepanel, have.xlim =
## have.xlim, : 'aspect="iso"' approximate since 'relation="free"'
```

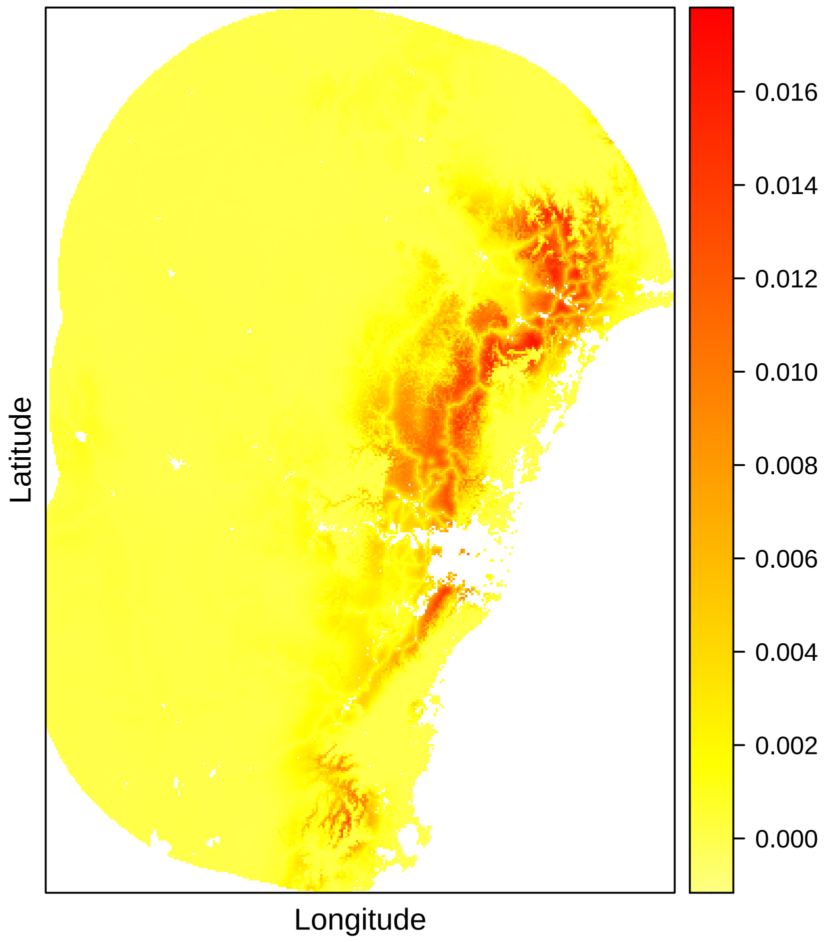

Figure 2: *Plot of predicted presences of Corymbia eximia using presence-only data when fitting the MCEM model. Here the max temperature predictor is assumed to be error-contaminated.*

Example 4: A VGAM example using the Prinia flaviventris capture-recapture data.

We use the *Prinia flaviventris* capture-recapture data set from Hwang, Huang and Wang (2007). Here, we fit naive `vglm` and `vgam` capture-recapture models, and the MCEM capture-recapture model. For all models we used the `posbinomial()` family provided in `VGAM`.

Load data and R-packages.

```r
suppressWarnings(suppressMessages(library(refitME)))
suppressMessages(library(VGAM))
data(Priniadata)
```

Setup all variables.

```r
tau <- 17   # No. of capture occasions.
w2 <- c(scale(Priniadata$w1)) # Standardize the wing length predictor.
Priniadata$w2 <- w2
```

Fit the naive vglm and vgam capture-recapture models.

```r
CR_naiv1 <- vglm(cbind(cap, noncap) ~ w2,
                 VGAM::posbinomial(omit.constant = TRUE,
                                   parallel = TRUE ~ w2),
                 data = Priniadata, trace = FALSE)
CR_naiv2 <- vgam(cbind(cap, noncap) ~ s(w2, df = 2),
                 VGAM::posbinomial(omit.constant = TRUE,
                                   parallel = TRUE ~ s(w2, df = 2)),
                 data = Priniadata, trace = FALSE)
```

Fit the MCEM capture-recapture model.

```r
sigma.sq.u <- 0.37/var(Priniadata$w1) # Measurement error variance.
B <- 100
CR_MCEM <- refitME(CR_naiv2, sigma.sq.u, B)
```

```
## [1] "convergence :-)"
## [1] 10
```

```r
detach(package:VGAM)
```