Estimating Equations in R: `geex`

B. Saul

2016-10-10

M-estimation theory provides a framework for asympotic properties of estimators that are solutions to estimating equations. Regression methods such as Generalized Linear Models (GLM) and Generalized Estimating Equations (GEE) fit in this framework. Countless R packages implement specific applications of estimating equations. A common reason to use M-estimation is to compute the empirical sandwich variance estimator - an asymptotically Normal and "robust" covariance. Many packages compute this variance estimator automatically, and packages such as `sandwich` take the output of other modeling methods to compute this variance estimate.

`geex` aims to be provide a more general framework that any modelling method can use to compute point and variance estimates for parameters that are solutions to estimating equations. The basic idea:

- Analyst provides three things: (1) data, (2) instructions on how to split the data into independent units and (3) a function that takes unit-level data and returns a function in terms of parameters.
- `geex` computes point estimates and variance estimates for the parameters.

Basic Setup

I mostly follow the notation of Stefanski and Boos. I tried to keep notation in the code similar to mathematical notation.

Suppose we have m independent or nearly independent units of observations.

$$\sum_{i=1}^{m} \psi(O_i, \theta) = 0$$

Where ψ is vector of length p corresponding to the number of parameters in θ.

For notational ease, let $\psi(O_i, \theta) = \psi_i$ Let:

$$A_i = -\frac{\partial \psi(O_i, \theta)}{\partial \theta}$$

$$A = \sum_{i=1}^{m} A_i$$

$$B_i = \psi_i \psi_i^T$$

$$B = \sum_{i=1}^{m} B_i$$

$$\Sigma = A^{-1} B \{A^{-1}\}^T$$

Stefanski & Boos example 1

Example 1 illustrates calculation of sample mean and variance using estimating equations. I generate a data set with 100 observations drawn from a Normal(5, 2) distribution. Table translates the estimating equations into the `R` function needed for `geex`:

Table 1: Translating math to code

$\psi(Y_i, \theta) = \begin{pmatrix} Y_i - \theta_1 \\ (Y_i - \theta_1)^2 - \theta_2 \end{pmatrix}$	`SB1_eefun <- function(data){ function(theta){ with(data, c(Y - theta[1], (Y - theta[1])^2 - theta[2] ) ) } }`

```
SB1_eefun <- function(data){
  function(theta){
    with(data,
      c(Y - theta[1],
       (Y - theta[1])^2 - theta[2] )
    )
  }
}
```

With the `eeFUN` function prepared, it is passed to `estimate_equations` along with the data, a character string naming the variable that identifies groups within the dataset, and starting values for the root finder.

```
estimates <- estimate_equations(eeFUN = SB1_eefun,
                                data  = dt, units = 'id',
                                roots = c(1,1))
```

Table 2: " test "

	$\hat{\theta}$	$\hat{\Sigma}$
Closed form	$\begin{pmatrix} 5.4314 & 4.7080 \end{pmatrix}$	$\begin{pmatrix} 0.0471 & 0.0141 \\ 0.0141 & 0.2983 \end{pmatrix}$
geex	$\begin{pmatrix} 5.4314 & 4.7080 \end{pmatrix}$	$\begin{pmatrix} 0.0471 & 0.0141 \\ 0.0141 & 0.2983 \end{pmatrix}$
Decimal of difference	$\begin{pmatrix} Inf & 16 \end{pmatrix}$	$\begin{pmatrix} 13 & 13 \\ 13 & 12 \end{pmatrix}$

Stefanski & Boos example 2

Example 2 illustrates calculation of a ratio estimator. I generate a data set with 100 observations where $Y \sim N(5, 2)$ and $X \sim N(2, 0.2)$. Table translates the estimating equations into the `R` function needed for `geex`:

Table 3: Translating math to code

$$\psi(Y_i, \theta) = \begin{pmatrix} Y_i - \theta_1 \\ X_i - \theta_2 \\ \theta_1 - \theta_3\theta_2 \end{pmatrix}$$

```
SB2_eefun <- function(data){
  function(theta){
    with(data,
      c(Y - theta[1],
        X - theta[2],
        theta[1] - (theta[3] * theta[2]) )
    )
  }
}
```

```r
estimates <- estimate_equations(eeFUN = SB2_eefun,
                                data  = dt, units = 'id',
                                roots = c(1, 1, 1))
```

Table 4: " test "

	$\hat{\theta}$	$\hat{\Sigma}$
Closed form	$\begin{pmatrix} 4.8704 & 2.0047 & 2.4295 \end{pmatrix}$	$\begin{pmatrix} 0.0382 & 0.0009 & 0.0180 \\ 0.0009 & 0.0005 & -0.0001 \\ 0.0180 & -0.0001 & 0.0091 \end{pmatrix}$
geex	$\begin{pmatrix} 4.8704 & 2.0047 & 2.4295 \end{pmatrix}$	$\begin{pmatrix} 0.0379 & 0.0009 & 0.0178 \\ 0.0009 & 0.0005 & -0.0001 \\ 0.0178 & -0.0001 & 0.0090 \end{pmatrix}$
Decimal of difference	$\begin{pmatrix} Inf & 16 & 10 \end{pmatrix}$	$\begin{pmatrix} 4 & 6 & 4 \\ 6 & 6 & 6 \\ 4 & 6 & 5 \end{pmatrix}$

Stefanski & Boos example 3

Example 3 illustrates calculation of a ratio estimator. I generate a data set with 100 observations where $Y \sim N(5, 4)$. Table translates the estimating equations into the R function needed for `geex`:

Table 5: Translating math to code

$$\psi(Y_i, \theta) = \begin{pmatrix} Y_i - \theta_1 \\ X_i - \theta_2 \\ \sqrt{\theta_2} - \theta_3 \\ log(\theta_2) - \theta_4 \end{pmatrix}$$

```r
SB3_eefun <- function(data){
  function(theta){
    with(data,
      c(Y - theta[1],
       (Y - theta[1])^2 - theta[2],
       sqrt(theta[2]) - theta[3],
       log(theta[2]) - theta[4])
    )
  }
}
```

```r
estimates <- estimate_equations(eeFUN= SB3_eefun,
                                data  = dt, units = 'id',
                                roots = c(1, 1, 1, 1))
```

Table 6: " Example 3 "

	$\hat{\theta}$	$\hat{\Sigma}$
Closed form	$\begin{pmatrix} 5.0117 & 16.5029 & 4.0624 & 2.8035 \end{pmatrix}$	$\begin{pmatrix} 0.1650 & 0.1116 & 0.0137 & 0.0068 \\ 0.1116 & 5.3752 & 0.6616 & 0.3257 \\ 0.0137 & 0.6616 & 0.0814 & 0.0401 \\ 0.0068 & 0.3257 & 0.0401 & 0.0197 \end{pmatrix}$
geex	$\begin{pmatrix} 5.0117 & 16.5029 & 4.0624 & 2.8035 \end{pmatrix}$	$\begin{pmatrix} 0.1650 & 0.1116 & 0.0137 & 0.0068 \\ 0.1116 & 5.3752 & 0.6616 & 0.3257 \\ 0.0137 & 0.6616 & 0.0814 & 0.0401 \\ 0.0068 & 0.3257 & 0.0401 & 0.0197 \end{pmatrix}$
Decimal of difference	$\begin{pmatrix} Inf & Inf & Inf & 16 \end{pmatrix}$	$\begin{pmatrix} 13 & 12 & 13 & 13 \\ 12 & 10 & 12 & 12 \\ 13 & 12 & 12 & 13 \\ 13 & 12 & 13 & 13 \end{pmatrix}$

Stefanski & Boos example 4

Example 4 illustrates calculation of an instumental variable estimator. I generate a data set with 100 observations where INPUT data generation. Table translates the estimating equations into the `R` function needed for `geex`:

Table 7: Translating math to code

$$\psi(Y_i, T_i, W_i\theta) = \begin{pmatrix} \theta_1 - T_i \\ \theta_2 - W_i \\ (Y_i - \theta_3 W_i)(\theta_2 - W_i) \\ (Y_i - \theta_4 W_i)(\theta_1 - T_i) \end{pmatrix}$$

```
SB_eefun <- function(data){
  function(theta){
    with(data,
      c(theta[1] - T_,
        theta[2] - W,
        (Y - (theta[3] * W)) * (theta[2] - W),
        (Y - (theta[4] * W)) * (theta[1] - T_))
    )
  }
}
```

```
estimates <- estimate_equations(eeFUN = SB4_eefun,
                                data  = dt, units = 'id',
                                roots = c(1, 1, 1, 1))
```

```
YW_model <- lm(Y ~ W, data = dt)
YT_model <- lm(Y ~ T_, data = dt)
WT_model <- lm(W ~ T_, data = dt)
## closed form roots
theta_cls <- c(theta1 = mean(dt$T_),
  theta2 = mean(dt$W),
  theta3 = coef(YW_model)[2],
  theta4 = coef(YT_model)[2]/coef(WT_model)[2])

## closed form covariance
# Not sure how compute SB's closed form since it depends on X, which is
# supposed to be unobserved.
Sigma_cls <- matrix(NA, nrow = 2, ncol = 2)
```

Table 8: " Example 4 "

	$\hat{\theta}$	$\hat{\Sigma}$
Closed form	$\begin{pmatrix} 1.9245 & -0.0550 & 3.0541 & 3.0969 \end{pmatrix}$	$\begin{pmatrix} \end{pmatrix}$
geex	$\begin{pmatrix} 1.9245 & -0.0550 & 3.0541 & 3.0969 \end{pmatrix}$	$\begin{pmatrix} 0.0002 & 0.0003 \\ 0.0003 & 0.0004 \end{pmatrix}$
Decimal of difference	$\begin{pmatrix} Inf & 17 & 10 & 10 \end{pmatrix}$	$\begin{pmatrix} \end{pmatrix}$

Stefanski & Boos example 5

Example 5 illustrates calculation of an instumental variable estimator. I generate a data set with 100 observations where $X \sim N(2, 1)$. Let $\theta_0 = 0$. Table translates the estimating equations for the Hodges-Lehmann location estimation and the sample mean into the `R` function needed for `geex`:

Table 9: Translating math to code

$\psi(Y_i, \theta) = \begin{pmatrix} IC_{\hat{\theta}_{HL}}(X; \theta_0) - (\theta_1 - \theta_0) \\ X_i - \theta_2 \end{pmatrix}$	`SB5_eefun <- function(data, theta0 = 0){`
	`  Xi <- data$X`
	`  IC_HL <- (1/IC_denom) * (F0(Xi, theta0) - 0.5)`
	`  function(theta){`
	`     c(IC_HL - (theta[1] - theta0),`
	`       Xi - theta[2])`
	`  }`
	`}`

```
F0 <- function(y, theta0, distrFUN = pnorm){
  distrFUN(y - theta0, mean = 0)
}

f0 <- function(y, densFUN){
  densFUN(y, mean = 0)
}

integrand <- function(y, densFUN = dnorm){
  f0(y, densFUN = densFUN)^2
}

IC_denom <- integrate(integrand, lower = -Inf, upper = Inf)$value

SB5_eefun <- function(data, theta0 = 0){
  Xi <- data$X
  IC_HL <- (1/IC_denom) * (F0(Xi, theta0) - 0.5)
  function(theta){
     c(IC_HL - (theta[1] - theta0),
       Xi - theta[2])
  }
}
```

```
estimates <- estimate_equations(eeFUN = SB5_eefun,
                                data  = dt, units = 'id',
                                roots = c(1, 1))
```

```
X <- dt$X
pair_means <- numeric(length(X) - 1)
for(i in 1:(length(X) - 1)){
 pair_means[i] <-  (X[i] + X[i + 1])/2
}

theta_cls <- c(median(pair_means), mean(X))

## closed form covariance
# Not sure how compute SB's closed form since it depends on X, which is
# supposed to be unobserved.
Sigma_cls <- matrix(c(1/(12 * IC_denom^2) / n, NA, NA, NA),
                    nrow = 2, ncol = 2, byrow = TRUE)
```

Table 10: " Example 5 "

	$\hat{\theta}$	$\hat{\Sigma}$
Closed form	$\begin{pmatrix} 1.9866 & 1.9056 \end{pmatrix}$	$\begin{pmatrix} 0.0105 & \\ & \end{pmatrix}$
geex	$\begin{pmatrix} 1.4103 & 1.9056 \end{pmatrix}$	$\begin{pmatrix} 0.0038 & 0.0056 \\ 0.0056 & 0.0121 \end{pmatrix}$
Decimal of difference	$\begin{pmatrix} 1 & 9 \end{pmatrix}$	$\begin{pmatrix} 3 & \\ & \end{pmatrix}$

Stefanski & Boos example 6

Example 6 illustrates calculation of the Huber estimator of the center of symmetric distributions. I generate a data set with 100 observations where $Y \sim N(2, 1)$. Table translates the estimating equations for the Huber estimator for the center of symmetric distributions into the `R` function needed for `geex`:

Table 11: Translating math to code

$$\psi_k(Y_i, \theta) = \left((Y_i - \theta) * I(|(Y_i - \theta)| \leq k) + k * sgn(Y_i - \theta)\right)$$

```
SB6_eefun <- function(data, k = 1.345){
  function(theta){
    x <- data$Y - theta[1]
    if(abs(x) <= k) x else sign(x) * k
  }
}
```

```
estimates <- estimate_equations(eeFUN = SB6_eefun,
                                data  = dt, units = 'id',
                                roots = 1)
```

```
theta_cls <- MASS::huber(dt$Y)$mu

psi_k <- function(x, k = 1.5){
  if(abs(x) <= k) x else sign(x) * k
}

A <- lapply(dt$Y, function(y){
  x <- y - theta_cls
  -numDeriv::grad(psi_k, x = x)
```

```
}) %>% unlist() %>% mean()

B <- lapply(dt$Y, function(y){
  x <- y - theta_cls
  psi_k(x = x)^2
}) %>% unlist() %>% mean()

## closed form covariance
Sigma_cls <- matrix(A * B * A / n)
```

Table 12: " Example 6 "

	$\hat{\theta}$	$\hat{\Sigma}$
Closed form	(2.0387)	(0.0058)
geex	(2.0340)	(0.0111)
Decimal of difference	(3)	(3)

Stefanski & Boos example 7

Example 7 illustrates calculation of sample quantiles using M-estimation. I generate a data set with 100 observations where $Y \sim N(2, 1)$. Table translates the estimating equations for two sample quantiles (median and 65th percentile) into the `R` function needed for `geex`:

Table 13: Translating math to code

$$\psi_k(Y_i, \theta) = \begin{pmatrix} 0.5 - I(Y_i \leq \theta_1) \\ 0.65 - I(Y_i \leq \theta_2) \end{pmatrix}$$

```
SB7_eefun <- function(data){
  function(theta){
    with(data,
      c(0.5  - (Y <= theta[1]),
        0.65 - (Y <= theta[2]))
    )
  }
}
```

```
estimates <- estimate_equations(eeFUN = SB7_eefun,
                                data  = dt, units = 'id',
                                roots = c(.5, .65))
```

```
theta_cls <- c(quantile(dt$Y, 0.5), quantile(dt$Y, 0.65))
```

Stefanski & Boos example 8

Example 8 illustrates robust regression. I generate a data set with 50 observations where half of the observation have $X_i = 1$ and the others have $X_i = 0$. \$Y = \$ 0.5 + 2 $X_i + \epsilon_i$ and $\epsilon_i \sim N(0, 1)$. Table translates the estimating equations for two sample quantiles (median and 65th percentile) into the `R` function needed for `geex`:

Table 14: Translating math to code

$\psi_k(Y_i, \theta) = \left(\psi_k(Y_i - \mathbf{x}_i^T\beta)\mathbf{x}_i\right)$	`SB8_eefun <- function(data){`
	`  Yi <- data$Y`
	`  xi <- model.matrix(Y ~ X, data = data)`
	`  function(theta){`
	`    r <- Yi - xi %*% theta`
	`    c(psi_k(r) %*% xi)`
	`  }`
	`}`

```
estimates <- estimate_equations(eeFUN = SB8_eefun,
                                data  = dt, units = 'id',
                                roots = c(1, 1))
```

```
m <- MASS::rlm(Y ~ X, data = dt, method = 'M')
theta_cls <- coef(m)
Sigma_cls <- vcov(m)
```

Table 15: " Example 8 "

	$\hat{\theta}$	$\hat{\Sigma}$
Closed form	$\begin{pmatrix} 0.3012 & 2.2534 \end{pmatrix}$	$\begin{pmatrix} 0.0495 & -0.0495 \\ -0.0495 & 0.0990 \end{pmatrix}$
geex	$\begin{pmatrix} 0.2966 & 2.2408 \end{pmatrix}$	$\begin{pmatrix} 0.0308 & -0.0308 \\ -0.0308 & 0.1030 \end{pmatrix}$
Decimal of difference	$\begin{pmatrix} 3 & 2 \end{pmatrix}$	$\begin{pmatrix} 2 & 2 \\ 2 & 3 \end{pmatrix}$

Stefanski & Boos example 9

Example 8 illustrates estimation of a generalized linear model. I generate a data set with 100 observations where half of the observation have $X_{1i} = 1$ and the others have $X_{1i} = 0$. $Y_i \sim Bern[\text{logit}^{-1}(\ 0.5 + 2\ X_{1i} + 0.1\ X_{2i})]$. Table translates the estimating equations for two sample quantiles (median and 65th percentile) into the `R` function needed for `geex`:

Table 16: Translating math to code

$\psi_k(Y_i, \theta) = \left(D_i(\beta)\frac{Y_i - \mu_i(\beta)}{V_i(\beta)\tau}\right)$	`SB9_eefun <- function(data){`
	`  Yi <- data$Y`
	`  xi <- model.matrix(Y ~ X1 + X2, data = data, drop = FALSE)`
	`  function(theta){`
	`    lp <- xi %*% theta`
	`    mu <- plogis(lp)`
	`    D  <- t(xi) %*% dlogis(lp)`
	`    V  <- mu * (1 - mu)`
	`    D %*% solve(V) %*% (Yi - mu)`
	`  }`
	`}`

```
estimates <- estimate_equations(eeFUN = SB9_eefun,
                                data  = dt, units = 'id',
                                roots = c(1, 1, 1))
```

```r
m <- glm(Y ~ X1 + X2, data = dt, family = binomial(link = 'logit'))
theta_cls <- coef(m)
Sigma_cls <- sandwich(m)
```

Table 17: " Example 9 "

	$\hat{\theta}$	$\hat{\Sigma}$
Closed form	$\begin{pmatrix} -0.1500 & 2.3191 & 0.6287 \end{pmatrix}$	$\begin{pmatrix} 0.1449 & -0.1134 & -0.1295 \\ -0.1134 & 0.3705 & 0.0657 \\ -0.1295 & 0.0657 & 0.2638 \end{pmatrix}$
geex	$\begin{pmatrix} -0.1500 & 2.3191 & 0.6287 \end{pmatrix}$	$\begin{pmatrix} 0.1449 & -0.1134 & -0.1295 \\ -0.1134 & 0.3705 & 0.0657 \\ -0.1295 & 0.0657 & 0.2638 \end{pmatrix}$
Decimal of difference	$\begin{pmatrix} 14 & 13 & 14 \end{pmatrix}$	$\begin{pmatrix} 9 & 8 & 9 \\ 8 & 8 & 8 \\ 9 & 8 & 8 \end{pmatrix}$

Stefanski & Boos example 10

Example 10 illustrates testing equality of success probablities.

Table 18: Translating math to code

$$\psi_k(Y_i, n_i, \theta) = \begin{pmatrix} \frac{(Y_i - n_i\theta_2)^2}{n_i\theta_2(1-\theta_2)} - \theta_1 \\ Y_i - n_i\theta_2 \end{pmatrix}$$

```r
SB10_eefun <- function(data){
  Y <- data$ft_made
  n <- data$ft_attp
  function(theta){
    p <- theta[2]
    c(((Y - (n * p))^2)/(n * p * (1 - p))  - theta[1],
      Y - n * p)
  }
}
```

```r
estimates <- estimate_equations(eeFUN = SB10_eefun,
                                data  = shaq, units = 'game',
                                roots = c(.5, .5))
```

```r
V11 <- function(p) {
  k <- length(nrow(shaq))
  sumn <- sum(shaq$ft_attp)
  sumn_inv <- sum(1/shaq$ft_attp)
  term2_n <- 1 - (6 * p) + (6 * p^2)
  term2_d <- p * (1 - p)
  term2 <- term2_n/term2_d
  print(term2)
  term3 <- ((1 - 2 * p)^2)/( (sumn/k) * p * (1 - p))
  print(term3)
  2 + (term2 * (1/k) * sumn_inv)  - term3
}

### ???? I keep getting a negative value for V11
```

```r
p_tilde <- sum(shaq$ft_made)/sum(shaq$ft_attp)
V <- V11(.45)
```

```
## [1] -1.959596
## [1] 0.0001365001
```

```r
V
```

```
## [1] -2.497375
```

```r
pnorm(estimates$parameters[1], mean = 1, sd = sqrt(V))
```

```
## Warning in sqrt(V): NaNs produced
```

```
## [1] NaN
```

Small Sample Corrections of Fay (2001)

Bias correction

$$H_i = \{1 - min(b, \{A_i A\}_{jj})\}^{-1/2}$$

Where b is a constant chosen by the analyst. Fay lets $b = 0.75$. Note that H_i is a diagonal matrix.

$$B_i^{bc} = H_i \psi_i \psi_i^T H_i$$

$$B^{bc} = \sum_{i=1}^{m} B_i^{bc}$$

$$\Sigma^{bc} = A^{-1} B^{bc} \{A^{-1}\}^T$$

Degrees of Freedom corrections

Let L be the contrast of interest (e.g.) $(0, \dots, 0, 1, -1)$ for a causal difference when the last two elements of the estimating equations are the counterfactual means.

$$\mathcal{I} = [I_p \cdots I_p]$$

where I_p is a $p \times p$ identity matrix.

$$G = I_{pm} - \begin{bmatrix} A_1^{bc} \\ \vdots \\ A_m \end{bmatrix} A^{-1} \mathcal{I}$$

$$M = diag\{H_i A^{-1} L L^T (A^{-1})^T H_i\}$$

$$C = G^T M G$$

$$w_i = L^T \left[\left\{ \sum_{j \neq i} A_i \right\}^{-1} - A^{-1} \right] L$$

$$\bar{w} = \sum_{i=1}^{m} w_i$$

$$A_i^{bc} = \frac{w_i}{\bar{w}} B^{bc}$$

$$\hat{df}_1 = \frac{\{Tr(diag(A_i)C)\}^2}{Tr(diag(A_i)Cdiag(A_i)C)}$$

$$\hat{df}_2 = \frac{\left\{Tr(diag(A_i^{bc})C)\right\}^2}{Tr(diag(A_i^{bc})Cdiag(A_i^{bc})C)}$$