The Calculus of M-Estimation in **R** with geex

Bradley C. Saul
NoviSci

Michael G. Hudgens
The University of North Carolina
at Chapel Hill

Abstract

M-estimation, or estimating equation, methods are widely applicable for point estimation and asymptotic inference. In this paper, we present an R package that can find roots and compute the empirical sandwich variance estimator for any set of user-specified, unbiased estimating equations. Examples from the M-estimation primer by Stefanski and Boos (2002) demonstrate use of the software. The package also includes a framework for finite sample, heteroscedastic, and autocorrelation variance corrections, and a website with an extensive collection of tutorials.

Keywords: empirical sandwich variance estimator, estimating equations, M-estimation, robust statistics, R.

1. Introduction

M-estimation methods are general class of statistical procedures for carrying out point estimation and asymptotic inference (Boos and Stefanski 2013). Also known as estimating equation or estimating function methods, M-estimation was originally developed in studying the large sample properties of robust statistics (Huber and Ronchetti 2009). The general result from M-estimation theory states that if an estimator can be expressed as the solution to an unbiased estimating equation, then under suitable regularity conditions the estimator is asymptotically Normal and its asymptotic variance can be consistently estimated using the empirical sandwich estimator. Many estimators can be expressed as solutions to unbiased estimating equations; thus M-estimation has extensive applicability. The primer by Stefanski and Boos (2002) demonstrates a variety of statistics which can be expressed as M-estimators, including the popular method of generalized estimating equations (GEE) for longitudinal data analysis (Liang and Zeger 1986).

Despite the broad applicability of M-estimation, existing statistical software packages implement M-estimators specific to particular forms of estimating equations such as GEE. This

paper introduces the package **geex** (Saul 2020) for R (R Core Team 2019), which can obtain point and variance estimates from any set of unbiased estimating equations. The analyst translates the mathematical expression of an estimating function into an R function that takes unit-level data and returns a function in terms of parameters. The package **geex** then uses numerical routines to compute parameter estimates and the empirical sandwich variance estimator.

This paper is outlined as follows. Section 2 reviews M-estimation theory and outlines how **geex** translates mathematical expressions of estimating functions into R syntax. Section 3 shows several examples with increasing complexity: three examples from Stefanski and Boos (2002) (hereafter SB), GEE, and a doubly robust causal estimator (Lunceford and Davidian 2004). All of the SB examples and several more are available at the package website (`https://bsaul.github.io/geex/`). Section 4 compares **geex** to existing R packages. Section 5 demonstrates the variance modification feature of **geex** with examples of finite sample corrections and autocorrelation consistent variance estimators for correlated data. Section 6 concludes with a brief discussion of the software's didactic utility and pragmatic applications.

2. From M-estimation math to code

In the basic set-up, M-estimation applies to estimators of the $p \times 1$ parameter $\theta = (\theta_1, \theta_2, \ldots, \theta_p)^\top$ which can be obtained as solutions to an equation of the form

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

where $O_1, \ldots, O_m$ are m independent and identically distributed (iid) random variables, and the function ψ returns a vector of length p and does not depend on i or m. See SB for the case where the O_i are independent but not necessarily identically distributed. The root of Equation 1 is referred to as an M-estimator and denoted by $\hat{\theta}$. M-estimators can be solved for analytically in some cases or computed numerically in general. Under certain regularity conditions, the asymptotic properties of $\hat{\theta}$ can be derived from Taylor series approximations, the law of large numbers, and the central limit theorem (Boos and Stefanski 2013, Section 7.2). In brief, let θ_0 be the true parameter value defined by $\int \psi(o, \theta_0) dF(o) = 0$, where F is the distribution function of O. Let $\dot{\psi}(O_i, \theta) = \partial \psi(O_i, \theta) / \partial \theta^\top$, $A(\theta_0) = \mathsf{E}[-\dot{\psi}(O_1, \theta_0)]$, and $B(\theta_0) = \mathsf{E}[\psi(O_1, \theta_0)\psi(O_1, \theta_0)^\top]$. Then under suitable regularity assumptions, $\hat{\theta}$ is consistent and asymptotically Normal, i.e.,

$$\sqrt{m}(\hat{\theta} - \theta_0) \xrightarrow{d} N(0, V(\theta_0)) \text{ as } m \to \infty,$$

where $V(\theta_0) = A(\theta_0)^{-1}B(\theta_0)\{A(\theta_0)^{-1}\}^\top$. The sandwich form of $V(\theta_0)$ suggests several possible large sample variance estimators. For some problems, the analytic form of $V(\theta_0)$ can be derived and estimators of θ_0 and other unknowns simply plugged into $V(\theta_0)$. Alternatively, $V(\theta_0)$ can be consistently estimated by the empirical sandwich variance estimator, where the expectations in $A(\theta)$ and $B(\theta)$ are replaced with their empirical counterparts. Let $A_i = -\dot{\psi}(O_i, \theta)|_{\theta=\hat{\theta}}$, $\bar{A}_m = m^{-1} \sum_{i=1}^{m} A_i$, $B_i = \psi(O_i, \hat{\theta})\psi(O_i, \hat{\theta})^\top$, and $\bar{B}_m = m^{-1} \sum_{i=1}^{m} B_i$. The empirical sandwich estimator of the variance of $\hat{\theta}$ is:

$$\hat{\Sigma} = \bar{A}_m^{-1} \bar{B}_m \{\bar{A}_m^{-1}\}^\top / m. \tag{2}$$

The **geex** package provides an application programming interface (API) for carrying out M-estimation. The analyst provides a function, called `estFUN`, corresponding to $\psi(O_i, \theta)$ that maps data O_i to a function of θ. Numerical derivatives approximate $\dot{\psi}$ so that evaluating $\hat{\Sigma}$ is entirely a computational exercise. No analytic derivations are required from the analyst.

Consider estimating the population mean $\theta = \mathsf{E}[Y_i]$ using the sample mean $\hat{\theta} = m^{-1} \sum_{i=1}^{m} Y_i$ of m iid random variables $Y_1, \ldots, Y_m$. The estimator $\hat{\theta}$ can be expressed as the solution to the following estimating equation:

$$\sum_{i=1}^{m} (Y_i - \theta) = 0.$$

which is equivalent to solving Equation 1 where $O_i = Y_i$ and $\psi(O_i, \theta) = Y_i - \theta$. An `estFUN` is a translation of ψ into an R function whose first argument is `data` and returns a function whose first argument is `theta`. An `estFUN` corresponding to the estimating equation for the sample mean of Y is:

```
meanFUN <- function(data) {
  function(theta) {
    data$Y - theta
  }
}
```

The **geex** package exploits R as functional programming language: functions can return and modify other functions (Wickham 2014, Chapter 10). If an estimator fits into the above framework, then the user need only specify `estFUN`. No other programming is required to obtain point and variance estimates. The remaining sections provide examples of translating ψ into an `estFUN`.

3. Calculus of M-estimation examples

The **geex** (Saul 2020) package is available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/package=geex. The first three examples of M-estimation from SB are presented here for demonstration. For these examples, the data are $O_i = \{Y_{1i}, Y_{2i}\}$ where $Y_1 \sim N(5, 16)$ and $Y_2 \sim N(2, 1)$ for $m = 100$ and are included in the `geexex` dataset. Another example applies GEE, which is elaborated on in Section 5 to demonstrate finite sample corrections. Lastly, a doubly-robust causal estimator of a risk difference introduces how estimating functions from multiple models can be stacked using **geex**.

3.1. Example 1: Sample moments

The first example estimates the population mean (θ_1) and variance (θ_2) of Y_1. Figure 1 shows the estimating equations and corresponding `estFUN` code. The solution to the estimating equations in Figure 1 are the sample mean $\hat{\theta}_1 = m^{-1} \sum_{i=1}^{m} Y_{1i}$ and sample variance $\hat{\theta}_2 = m^{-1} \sum_{i=1}^{m} (Y_{1i} - \hat{\theta}_1)^2$.

The primary **geex** function is `m_estimate`, which requires two inputs: `estFUN` (the ψ function), `data` (the data frame containing O_i for $i = 1, \ldots, m$). The package defaults to `rootSolve::multiroot` (Soetaert and Herman 2009; Soetaert 2009) for estimating the roots

$$\psi(Y_{1i}, \theta) = \begin{pmatrix} Y_{1i} - \theta_1 \\ (Y_{1i} - \theta_1)^2 - \theta_2 \end{pmatrix}$$

```
SB1_estfun <- function(data) {
  Y1 <- data$Y1
  function(theta) {
    c(Y1 - theta[1],
      (Y1 - theta[1])^2 - theta[2])
  }
}
```

Figure 1: Estimating equations and `estFUN` for Example 1.

$$\psi(Y_{1i}, Y_{2i}, \theta) = \begin{pmatrix} Y_{1i} - \theta_1 \\ Y_{2i} - \theta_2 \\ \theta_1 - \theta_3\theta_2 \end{pmatrix}$$

```
SB2_estfun <- function(data) {
  Y1 <- data$Y1; Y2 <- data$Y2
  function(theta) {
    c(Y1 - theta[1], Y2 - theta[2],
      theta[1] - (theta[3] * theta[2]))
  }
}
```

Figure 2: Estimating equations and `estFUN` for Example 2.

of Equation 1, though the solver algorithm can be specified in the `root_control` argument. Starting values for `rootSolve::multiroot` are passed via the `root_control` argument; e.g.,

```
R> library("geex")
R> results <- m_estimate(estFUN = SB1_estfun, data = geexex,
+     root_control = setup_root_control(start = c(1, 1)))
```

The `m_estimate` function returns an object of the S4 class 'geex', which contains an `estimates` slot and `vcov` slot for $\hat{\theta}$ and $\hat{\Sigma}$, respectively. These slots can be accessed by the functions `coef` (or `roots`) and `vcov`. The point estimates obtained for θ_1 and θ_2 are analogous to the base R functions `mean` and `var` (after multiplying by $m - 1/m$ for the latter). SB gave a closed form for $A(\theta_0)$ (an identity matrix) and $B(\theta_0)$ (not shown here) and suggest plugging in sample moments to compute $B(\hat{\theta})$. The maximum absolute difference between either the point or variance estimates is `4e-11`, thus demonstrating excellent agreement between the numerical results obtained from **geex** and the closed form solutions for this set of estimating equations and data.

3.2. Example 2: Ratio estimator

This example calculates a ratio estimator (Figure 2) and illustrates the delta method via M-estimation. The estimating equations target the means of Y_1 and Y_2, labelled θ_1 and θ_2, as well as the estimand $\theta_3 = \theta_1/\theta_2$. The solution to Equation 1 for this ψ function yields $\hat{\theta}_3 = \bar{Y}_1/\bar{Y}_2$, where $\bar{Y}_j$ denotes the sample mean of $Y_{j1}, \ldots, Y_{jm}$ for $j = 1, 2$.

SB gave closed form expressions for $A(\theta_0)$ and $B(\theta_0)$, into which we plug in appropriate estimates for the matrix components and compare to the results from **geex**. The point estimates again show excellent agreement (maximum absolute difference `4.4e-16`), while the covariance estimates differ by the third decimal (maximum absolute difference `2e-12`).

$$\psi(Y_{1i}, \theta) = \begin{pmatrix} Y_{1i} - \theta_1 \\ (Y_{1i} - \theta_1)^2 - \theta_2 \\ \sqrt{\theta_2} - \theta_3 \\ \log(\theta_2) - \theta_4 \end{pmatrix}$$

```
SB3_estfun <- function(data) {
  Y1 <- data$Y1
  function(theta) {
    c(Y1 - theta[1],
      (Y1 - theta[1])^2 - theta[2],
      sqrt(theta[2]) - theta[3],
      log(theta[2]) - theta[4])
  }
}
```

Figure 3: Estimating equations and `estFUN` for Example 3.

3.3. Example 3: Delta method continued

This example extends Example 1 to again illustrate the delta method. The estimating equations target not only the mean (θ_1) and variance (θ_2) of Y_1, but also the standard deviation (θ_3) and the log of the variance (θ_4) of Y_1. SB again provided a closed form for $A(\theta_0)$ and $B(\theta_0)$, which we compare to the **geex** results. The maximum absolute difference between **geex** and the closed form estimates for both the parameters and the covariance is `3.8e-11`.

3.4. Example 4: Generalized estimating equations

In their seminal paper, Liang and Zeger (1986) introduced generalized estimating equations (GEE) for the analysis of longitudinal or clustered data. Let m denote the number of independent clusters. For cluster i, let n_i be the cluster size, Y_i be the $n_i \times 1$ outcome vector, and X_i be the $n_i \times p$ matrix of covariates. Let $\mu(X_i; \theta) = \mathsf{E}[Y_i | X_i; \theta]$ and assume $\mu(X_i; \theta) = g^{-1}(X_i\theta)$, where g is some user-specified link function. The generalized estimating equations are:

$$\sum_{i=1}^{m} \psi(O_i, \theta) = \sum_{i=1}^{m} D_i^\top V_i^{-1} \{Y_i - \mu(X_i; \theta)\} = 0, \tag{3}$$

where $O_i = \{Y_i, X_i\}$ and $D_i = \partial\mu(X_i; \theta)/\partial\theta$. The covariance matrix is modeled by $V_i = \phi W_i^{0.5} R(\alpha) W_i^{0.5}$ where the matrix $R(\alpha)$ is the "working" correlation matrix. The example below uses an exchangeable correlation structure with off-diagonal elements α. The matrix W_i is a diagonal matrix with elements containing $\partial^2\mu(X_i; \theta)/\partial\theta^2$. Equation 3 can be translated into an `estFUN` as:

```
R> gee_estfun <- function(data, formula, family) {
+    X <- model.matrix(object = formula, data = data)
+    Y <- model.response(model.frame(formula = formula, data = data))
+    n <- nrow(X)
+    function(theta, alpha, psi) {
+      mu <- drop(family$linkinv(X %*% theta))
+      Dt <- crossprod(X, diag(mu, nrow = n))
+      W <- diag(family$variance(mu), nrow = n)
+      R <- matrix(alpha, nrow = n, ncol = n)
+      diag(R) <- 1
+      V <- psi * (sqrt(W) %*% R %*% sqrt(W))
```

```
+         Dt %*% solve(V, (Y - mu))
+     }
+  }
```

This `estFUN` treats the correlation parameter α and scale parameter ϕ as fixed, though some estimation algorithms use an iterative procedure that alternates between estimating θ_0 and these parameters. By customizing the root finding function, such an algorithm can be implemented using **geex** (see `vignette("v03_root_solvers")` for more information).

We use this example to compare covariance estimates obtained from the `gee` function (Carey 2019), and so do not estimate roots using **geex**. To compute only the sandwich variance estimator, set `compute_roots = FALSE` and pass estimates of θ_0 via the `roots` argument. For this example, estimated roots of Equation 3, i.e., $\hat{\theta}$, and estimates for α and ϕ are extracted from the object returned by `gee`. This example shows that an `estFUN` can accept additional arguments to be passed to either the outer (`data`) function or the inner (`theta`) function. Unlike previous examples, the independent units are clusters (types of wool), which is specified in `m_estimate` by the `units` argument. By default, m equals the number of rows in the data frame.

```
R> g <- gee::gee(breaks ~ tension, id = wool, data = warpbreaks,
+     corstr = "exchangeable")
R> results <- m_estimate(estFUN = gee_estfun, data = warpbreaks,
+     units = "wool", roots = coef(g), compute_roots = FALSE,
+     outer_args = list(formula = breaks ~ tension, family = gaussian()),
+     inner_args = list(alpha = g$working.correlation[1, 2],
+       psi = g$scale))
```

The maximum absolute difference between the estimated covariances computed by `gee` and **geex** is `2.7e-09`.

3.5. Example 5: Doubly robust causal effect estimator

Estimators of causal effects often have the form:

$$\sum_{i=1}^{m} \psi(O_i, \theta) = \sum_{i=1}^{m} \begin{pmatrix} \psi(O_i, \nu) \\ \psi(O_i, \beta) \end{pmatrix} = 0, \tag{4}$$

where ν are parameters in nuisance model(s), such as a propensity score model, and β are the target causal parameters. Even when ν represent parameters in common statistical models, deriving a closed form for a sandwich variance estimator for $\hat{\beta}$ based on Equation 4 may involve tedious and error-prone derivative and matrix calculations (e.g., see the appendices of Lunceford and Davidian 2004; Perez-Heydrich, Hudgens, Halloran, Clemens, Ali, and Emch 2014). In this example, we show how an analyst can avoid these calculations and compute the empirical sandwich variance estimator using **geex**.

Lunceford and Davidian (2004) review several estimators of causal effects from observational data. To demonstrate a more complicated estimator involving multiple nuisance models, we implement the doubly robust estimator:

$$\hat{\Delta}_{DR} = \sum_{i=1}^{m} \frac{Z_i Y_i - (Z_i - \hat{e}_i) m_1(X_i, \hat{\alpha}_1)}{\hat{e}_i} - \frac{(1 - Z_i) Y_i - (Z_i - \hat{e}_i) m_0(X_i, \hat{\alpha}_0)}{1 - \hat{e}_i}. \tag{5}$$

This estimator targets the average causal effect, $\Delta = \mathsf{E}[Y(1) - Y(0)]$, where $Y(z)$ is the potential outcome for an observational unit had it been exposed to the level z of the binary exposure variable Z. The estimated propensity score, $\hat{e}_i$, is the estimated probability that unit i received $z = 1$ and $m_z(X_i, \hat{\alpha}_z)$ is an outcome regression model with baseline covariates X_i and estimated paramaters $\hat{\alpha}_z$ for the subset of units with $Z = z$. This estimator has the property that if either the propensity score model or the outcome models are correctly specified, then the solution to Equation 5 will be a consistent and asymptotically Normal estimator of Δ.

This estimator and its estimating equations can be translated into an `estFUN` as:

```
R> dr_estFUN <- function(data, models) {
+    Z <- data$Z
+    Y <- data$Y
+    Xe <- grab_design_matrix(data = data,
+      rhs_formula = grab_fixed_formula(models$e))
+    Xm0 <- grab_design_matrix(data = data,
+      rhs_formula = grab_fixed_formula(models$m0))
+    Xm1 <- grab_design_matrix(data = data,
+      rhs_formula = grab_fixed_formula(models$m1))
+    e_pos <- 1:ncol(Xe)
+    m0_pos <- (max(e_pos) + 1):(max(e_pos) + ncol(Xm0))
+    m1_pos <- (max(m0_pos) + 1):(max(m0_pos) + ncol(Xm1))
+    e_scores <- grab_psiFUN(models$e, data)
+    m0_scores <- grab_psiFUN(models$m0, data)
+    m1_scores <- grab_psiFUN(models$m1, data)
+    function(theta) {
+      e <- plogis(Xe %*% theta[e_pos])
+      m0 <- Xm0 %*% theta[m0_pos]
+      m1 <- Xm1 %*% theta[m1_pos]
+      rd_hat <- (Z*Y - (Z - e) * m1) / e -
+        ((1 - Z) * Y - (Z - e) * m0) / (1 - e)
+      c(e_scores(theta[e_pos]),
+        m0_scores(theta[m0_pos]) * (Z == 0),
+        m1_scores(theta[m1_pos]) * (Z == 1),
+        rd_hat - theta[length(theta)])
+    }
+  }
```

This `estFUN` presumes that the user will pass a list containing fitted model objects for the three nuisance models: the propensity score model and one regression model for each treatment group. The functions `grab_design_matrix` and `grab_fixed_formula` are **geex** utilities for extracting relevant pieces of a model object. The function `grab_psiFUN` converts a fitted model object to an estimating function; for example, for a 'glm' object, `grab_psiFUN` uses the `data` to create a `function` of `theta` corresponding to the generalized linear model score function. The `m_estimate` function can be wrapped in another function, wherein nuisance models are fit and passed to `m_estimate`.

```
R> estimate_dr <- function(data, propensity_formula, outcome_formula) {
+     e_model <- glm(propensity_formula, data = data, family = binomial)
+     m0_model <- glm(outcome_formula, subset = (Z == 0), data = data)
+     m1_model <- glm(outcome_formula, subset = (Z == 1), data = data)
+     models <- list(e = e_model, m0 = m0_model, m1 = m1_model)
+     nparms <- sum(unlist(lapply(models, function(x) length(coef(x))))) + 1
+     m_estimate(estFUN = dr_estFUN, data = data,
+       root_control = setup_root_control(start = rep(0, nparms)),
+       outer_args = list(models = models))
+ }
```

The following code provides a function to replicate the simulation settings of Lunceford and Davidian (2004).

```
R> library("mvtnorm")
R> set.seed(123)
R> tau_0 <- c(-1, -1, 1, 1)
R> tau_1 <- tau_0 * -1
R> Sigma_X3 <- matrix(c(1, 0.5, -0.5, -0.5, 0.5, 1, -0.5, -0.5,
+     -0.5, -0.5, 1, 0.5, -0.5, -0.5, 0.5, 1), ncol = 4, byrow = TRUE)
R> gen_data <- function(n, beta, nu, xi) {
+     X3 <- rbinom(n, 1, prob = 0.2)
+     V3 <- rbinom(n, 1, prob = (0.75 * X3 + (0.25 * (1 - X3))))
+     hold <- rmvnorm(n,  mean = rep(0, 4), Sigma_X3)
+     colnames(hold) <- c("X1", "V1", "X2", "V2")
+     hold <- cbind(hold, X3, V3)
+     hold <- apply(hold, 1, function(x) {
+       x[1:4] <- x[1:4] + tau_1^(x[5]) * tau_0^(1 - x[5])
+       x
+     })
+     hold <- t(hold)[, c("X1", "X2", "X3", "V1", "V2", "V3")]
+     X <- cbind(Int = 1, hold)
+     Z <- rbinom(n, 1, prob = plogis(X[, 1:4] %*% beta))
+     X <- cbind(X[, 1:4], Z, X[, 5:7])
+     data.frame(Y = X %*% c(nu, xi) + rnorm(n), X[ , -1])
+ }
```

To show that `estimate_dr` correctly computes $\hat{\Delta}_{DR}$, the results from **geex** can be compared to computing $\hat{\Delta}_{DR}$ "by hand" for a simulated dataset.

```
R> dt <- gen_data(n = 1000, beta = c(0, 0.6, -0.6, 0.6),
+     nu =  c(0, -1, 1, -1, 2), xi = c(-1, 1, 1))
R> geex_results <- estimate_dr(dt, Z ~ X1 + X2 + X3, Y ~ X1 + X2 + X3)
R> e <- predict(glm(Z ~ X1 + X2 + X3, data = dt, family = binomial()),
+     type = "response")
R> m0 <- predict(glm(Y ~ X1 + X2 + X3, data = dt, subset = Z == 0),
+     newdata = dt)
```

```
R> m1 <- predict(glm(Y ~ X1 + X2 + X3, data = dt, subset = Z == 1),
+    newdata = dt)
R> del_hat <- with(dt, mean((Z * Y - (Z - e) * m1) / e)) -
+    with(dt, mean(((1 - Z) * Y - (Z - e) * m0) / (1 - e)))
```

This shows that `coef(geex_results)[13]` has a maximum absolute difference of `7.3e-10` to `del_hat`.

4. Comparison to existing software

The above examples demonstrate the basic utility of the **geex** package and the power of R's functional programming capability. The **gmm** package (Chaussé 2010) computes generalized methods of moments and generalized empirical likelihoods, estimation strategies similar to M-estimation, using user-defined functions like **geex**. To our knowledge, **geex** is the first R package to create an extensible API for any estimator that is the solution to estimating equations in the form of Equation 1. Existing R packages such as **gee** (Carey 2019), **geepack** (Halekoh, Hojsgaard, and Yan 2006), and **geeM** (McDaniel, Henderson, and Rathouz 2013) solve for parameters in a GEE framework. Other packages such as **fastM** (Duembgen, Nordhausen, and Schuhmacher 2018) and **smoothmest** (Hennig 2012) implement M-estimators for specific use cases.

For computing a sandwich variance estimator, **geex** is similar to the popular **sandwich** package (Zeileis 2004, 2006), which computes the empirical sandwich variance estimator from modeling methods such as `lm`, `glm`, `gam`, `survreg`, and others. For comparison to the exposition herein, the infrastructure of **sandwich** is explained in Zeileis (2006). Advantages of **geex** compared to **sandwich** include: (i) for custom applications, a user only needs to specify a single `estFUN` function as opposed to both the `bread` and `estfun` functions; (ii) as demonstrated in the examples above, the syntax of an `estFUN` may closely resemble the mathematical expression of the corresponding estimating function; (iii) estimating functions from multiple models are easily stacked; and (iv) point estimates can be obtained. The precision and computational speed of point and variance estimation in **geex**, however, depends on numerical approximations rather than analytic expressions.

To compare **sandwich** and **geex**, consider estimating $\hat{\Sigma}$ for the θ parameters in the following simple linear model contained in the `geexex` data: $Y_4 = \theta_1 + \theta_2 X_1 + \theta_3 X_2 + \epsilon$, where $\epsilon \sim N(0, 1)$. The estimating equation for θ in this model can be expressed in an `estFUN` as:

```
R> lm_estfun <- function(data) {
+    X <- cbind(1, data[["X1"]], data[["X2"]])
+    Y <- data[["Y4"]]
+    function(theta) {
+      crossprod(X, Y - X %*% theta)
+    }
+ }
```

Then $\hat{\theta}$ and $\hat{\Sigma}$ can be computed in **geex**:

```
R> results <- m_estimate(estFUN = lm_estfun, data = geexex,
+    root_control = setup_root_control(start = c(0, 0, 0)))
```

or from the `lm` and `sandwich` functions:

```
R> fm <- lm(Y4 ~ X1 + X2, data = geexex)
R> sand_vcov <- sandwich::sandwich(fm)
```

The results are virtually identical (maximum absolute difference `1.4e-12`). The `lm`/`sandwich` option is faster computationally, but **geex** can be sped up by, for example, changing the options of the derivative function via `deriv_control` or computing $\hat{\Sigma}$ using the parameter estimates from `lm`. While **geex** will never replace computationally optimized modeling functions such as `lm`, the important difference is that **geex** lays bare the estimating function used, which is both a powerful didactic tool as well as a programming advantage when developing custom estimating functions.

5. Variance corrections

The standard empirical sandwich variance estimator is known to perform poorly in certain situations. In small samples, $\hat{\Sigma}$ will tend to underestimate the variance of $\hat{\theta}$ (Fay and Graubard 2001). When observational units are not independent and/or do not share the same variance, consistent variance estimators can be obtained by modifying how $B(\theta_0)$ is estimated. The next two examples demonstrate using **geex** for finite sample and autocorrelation corrections, respectively.

5.1. Finite sample correction

Particularly in the context of GEE, many authors (e.g., see Paul and Zhang 2014; Li and Redden 2015) have proposed corrections that modify components of $\hat{\Sigma}$ and/or by assuming $\hat{\theta}$ follows a t (or F), as opposed to Normal, distribution with some estimated degrees of freedom. Many of the proposed corrections somehow modify a combination of the A_i, $\bar{A}_m$, B_i, or $\bar{B}_m$ matrices.

Users may specify functions that utilize these matrices to form corrections within **geex**. A finite sample correction function requires at least the argument `components`, which is an S4 object with slots for the `A` ($= \sum_i A_i$) matrix, `A_i` (a list of all m A_i matrices), the `B` ($= \sum_i B_i$) matrix, `B_i` (a list of all m B_i matrices), and `ee_i` (a list of the observed estimating function values for all m units). Additional arguments may also be specified, as shown in the example. The **geex** package includes the bias correction and degrees of freedom corrections proposed by Fay and Graubard (2001) in the `fay_bias_correction` and `fay_df_correction` functions respectively. The following demonstrates the construction and use of the bias correction. Fay and Graubard (2001) proposed the modified variance estimator $\hat{\Sigma}^{bc}(b) = \bar{A}_m^{-1}\bar{B}_m^{bc}(b)\{\bar{A}_m^{-1}\}^{\top}/m$, where:

$$B_m^{bc}(b) = \sum_{i=1}^{m} H_i(b)B_iH_i(b)^{\top},$$

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

and W_{jj} denotes the jj element of a matrix W. When $\{A_i\bar{A}_m^{-1}\}_{jj}$ is close to 1, the adjustment to $\hat{\Sigma}^{bc}(b)$ may be extreme, and the constant b is chosen by the analyst to limit over adjustments. The bias corrected estimator $\hat{\Sigma}^{bc}(b)$ can be implemented in **geex** by the following function:

```
R> bias_correction <- function(components, b) {
+     A <- grab_bread(components)
+     A_i <- grab_bread_list(components)
+     B_i <- grab_meat_list(components)
+     Ainv <- solve(A)
+     H_i <- lapply(A_i, function(m) {
+         diag((1 - pmin(b, diag(m %*% Ainv)))^(-0.5))
+     })
+     Bbc_i <- lapply(seq_along(B_i), function(i) {
+         H_i[[i]] %*% B_i[[i]] %*% H_i[[i]]
+     })
+     Bbc <- compute_sum_of_list(Bbc_i)
+     compute_sigma(A = A, B = Bbc)
+ }
```

The `compute_sum_of_list` sums over a list of matrices, while the `compute_sigma(A, B)` function simply computes $A^{-1}B\{A^{-1}\}^{\top}$. To use this bias correction, the `m_estimate` function accepts a named list of corrections to perform. Each element of the list is a 'correct_control' S4 object that can be created with the helper function `correction`, which accepts the argument `FUN` (the correction function) plus any arguments passed to `FUN` besides `components`; e.g.,

```
R> results <- m_estimate(estFUN = gee_estfun, data = warpbreaks,
+     units = "wool", roots = coef(g), compute_roots = FALSE,
+     outer_args = list(formula = breaks ~ tension,
+         family = gaussian(link = "identity")),
+     inner_args = list(alpha = g$working.correlation[1, 2],
+         psi = g$scale),
+     corrections = list(
+         bias_correction_.1 = correction(FUN = bias_correction, b = 0.1),
+         bias_correction_.3 = correction(FUN = bias_correction, b = 0.3)))
```

In the **geex** output, the slot `corrections` contains a list of the results of computing each item in the `corrections`, which can be accessed with the `get_corrections` function. The corrections of Fay and Graubard (2001) are also implemented in the **saws** package (Fay and Graubard 2001). Comparing the **geex** results to the results of the `saws::geeUOmega` function, the maximum absolute difference for any of the corrected estimated covariance matrices is `3.8e-09`.

5.2. Newey-West autocorrelation correction

When error terms are dependent, as in time series data, $\mathsf{E}[B]$ is challenging to estimate (Zeileis 2004). A solution is to estimate B using the pairwise sum,

$$\hat{B}_{AC} = \sum_{i,j=1}^{m} w_{|i-j|}\psi(O_i; \hat{\theta})\psi(O_j; \hat{\theta})^{\top},$$

where $w_{|i-j|}$ is a vector of weights that often reflect decreasing autocorrelation as the distance between i and j increases. Many authors have proposed ways of computing weights (see for

example, White and Domowitz 1984; Newey and West 1987; Andrews 1991; Lumley and Heagerty 1999).

To illustrate autocorrelation correction using **geex**, we implement the Newey-West correction (without pre-whitening) and compare to the `NeweyWest` function in **sandwich** (Zeileis 2004). The example is taken from the `NeweyWest` documentation.

```
R> x <- sin(1:100)
R> y <- 1 + x + rnorm(100)
R> dt <- data.frame(x = x, y = y)
R> fm <- lm(y ~ x)
R> lm_estfun <- function(data) {
+     X <- cbind(1, data[["x"]])
+     Y <- data[["y"]]
+     function(theta) {
+       crossprod(X, Y - X %*% theta)
+     }
+ }
R> nwFUN <- function(i, j, lag) {
+     ifelse(abs(i - j) <= lag, 1 - abs(i - j) / (lag + 1), 0)
+ }
R> nw_correction <- function(components, lag) {
+     A <- grab_bread(components)
+     ee <- grab_ee_list(components)
+     Bac <- compute_pairwise_sum_of_list(ee, .wFUN = nwFUN, lag = lag)
+     compute_sigma(A = A, B = Bac)
+ }
R> results <- m_estimate(estFUN = lm_estfun, data = dt,
+     root_control = setup_root_control(start = c(0, 0)),
+     corrections = list(
+       NW_correction = correction(FUN = nw_correction, lag = 1)))
R> get_corrections(results)[[1]]

            [,1]          [,2]
[1,]   0.008390531 -0.003379649
[2,] -0.003379649  0.018784695

R> sandwich::NeweyWest(fm, lag = 1, prewhite = FALSE)

             (Intercept)            x
(Intercept)  0.008390531 -0.003379649
x           -0.003379649  0.018784695
```

The function `lm_estfun` is essentially the same as the previous comparison to **sandwich** in Section 4. The function `nw_correction` performs the Newey-West adjustment using `nwFUN` which computes the Newey-West weights for lag L,

$$w_{|i-j|} = 1 - \frac{|i-j|}{L+1}.$$

The function `grab_ee_list` returns the list of observed estimating functions, $\psi(O_i, \hat{\theta})$, from the 'sandwich_components' object. The utility function `compute_pairwise_sum_of_list` computes $\hat{B}_{AC}$ using either (but not both) a fixed vector (argument `.w`) of weights or a function of `i` and `j` (argument `.wFUN`), which may include additional arguments such as `lag`, as in this case. For this example, **geex** and **sandwich** return nearly identical results.

6. Summary

This paper demonstrates how M-estimators and finite sample corrections can be transparently implemented in **geex** (Saul 2020). The package website (https://bsaul.github.io/geex/) showcases many examples of M-estimation including instrumental variables, sample quantiles, robust regression, generalized linear models, and more. A valuable feature of M-estimators is that estimating functions corresponding to parameters from multiple models may be combined, or "stacked," in a single set of estimating functions. The **geex** (Saul 2020) package makes it easy to stack estimating functions for the target parameters with estimating functions from each of the component models, as shown in the package vignette `v06_causal_example`. Indeed, the software was motivated by causal inference problems (Saul, Hudgens, and Mallin 2019) where target causal parameters are functions of parameters in multiple models.

The theory of M-estimation is broadly applicable, yet existing R packages only implement particular classes of M-estimators. With its functional programming capabilities, R routines can be more general. The **geex** framework epitomizes the extensible nature of M-estimators and explicitly translates the estimating function ψ into a corresponding `estFUN`. In this way, **geex** should be useful for practitioners developing M-estimators, as well as students learning estimating equation theory.

Acknowledgments

References

Andrews DWK (1991). "Heteroskedasticity and Autocorrelation Consistent Covariance Matrix Estimation." *Econometrica*, **59**(3), 817–858. doi:10.2307/2938229.

Boos DD, Stefanski LA (2013). *Essential Statistical Inference: Theory and Methods*. Springer-Verlag, New York.

Carey VJ (2019). *gee: Generalized Estimation Equation Solver*. Ported to R by Thomas Lumley and Brian D. Ripley; R package version 4.13-20, URL https://CRAN.R-project.org/package=gee.

Chaussé P (2010). "Computing Generalized Method of Moments and Generalized Empirical Likelihood with R." *Journal of Statistical Software*, **34**(11), 1–35. `doi:10.18637/jss.v034.i11`.

Duembgen L, Nordhausen K, Schuhmacher H (2018). **fastM**: *Fast Computation of Multivariate M-Estimators.* R package version 0.0-4, URL `https://CRAN.R-project.org/package=fastM`.

Fay MP, Graubard BI (2001). "Small-Sample Adjustments for Wald-Type Tests Using Sandwich Estimators." *Biometrics*, **57**(4), 1198–1206. `doi:10.1111/j.0006-341x.2001.01198.x`.

Halekoh U, Hojsgaard S, Yan J (2006). "The R Package **geepack** for Generalized Estimating Equations." *Journal of Statistical Software*, **15**(2), 1–11. `doi:10.18637/jss.v015.i02`.

Hennig C (2012). **smoothmest**: *Smoothed M-Estimators for 1-Dimensional Location.* R package version 0.1-2, URL `https://CRAN.R-project.org/package=smoothmest`.

Huber PJ, Ronchetti EM (2009). *Robust Statistics.* 2nd edition. John Wiley & Sons, Hoboken.

Li P, Redden DT (2015). "Small Sample Performance of Bias-Corrected Sandwich Estimators for Cluster-Randomized Trials with Binary Outcomes." *Statistics in Medicine*, **34**(2), 281–96. `doi:10.1002/sim.6344`.

Liang KY, Zeger SL (1986). "Longitudinal Data Analysis Using Generalized Linear Models." *Biometrika*, **73**(1), 13–22. `doi:10.1093/biomet/73.1.13`.

Lumley T, Heagerty P (1999). "Weighted Empirical Adaptive Variance Estimators for Correlated Data Regression." *Journal of the Royal Statistical Society B*, **61**(2), 459–477. `doi:10.1111/1467-9868.00187`.

Lunceford JK, Davidian M (2004). "Stratification and Weighting via the Propensity Score in Estimation of Causal Treatment Effects: A Comparative Study." *Statistics in Medicine*, **23**(19), 2937–2960. `doi:10.1002/sim.1903`.

McDaniel LS, Henderson NC, Rathouz PJ (2013). "Fast Pure R Implementation of GEE: Application of the **Matrix** Package." *The* R *Journal*, **5**, 181–187. `doi:10.32614/rj-2013-017`.

Newey WK, West KD (1987). "A Simple, Positive Semi-Definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix." *Econometrica*, **55**(3), 703–708. `doi:10.2307/1913610`.

Paul S, Zhang X (2014). "Small Sample GEE Estimation of Regression Parameters for Longitudinal Data." *Statistics in Medicine*, **33**(22), 3869–81. `doi:10.1002/sim.6198`.

Perez-Heydrich C, Hudgens MG, Halloran ME, Clemens JD, Ali M, Emch ME (2014). "Assessing Effects of Cholera Vaccination in the Presence of Interference." *Biometrics*, **70**(3), 731–741. `doi:10.1111/biom.12184`.

R Core Team (2019). R: *A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. URL `https://www.R-project.org/`.

Saul B (2020). **geex**: *An API for M-Estimation.* R package version 1.0.12, URL https://CRAN.R-project.org/package=geex.

Saul BC, Hudgens MG, Mallin MA (2019). "Downstream Effects of Upstream Causes." *Journal of the American Statistical Association*, **114**(528), 1493–1504. doi:10.1080/01621459.2019.1574226.

Soetaert K (2009). **rootSolve**: *Nonlinear Root Finding, Equilibrium and Steady-State Analysis of Ordinary Differential Equations.* R package version 1.6, URL https://CRAN.R-project.org/web/packages/rootSolve/index.html.

Soetaert K, Herman PMJ (2009). *A Practical Guide to Ecological Modelling. Using R as a Simulation Platform.* Springer-Verlag. doi:10.1007/978-1-4020-8624-3.

Stefanski LA, Boos DD (2002). "The Calculus of M-Estimation." *The American Statistician*, **56**(1), 29–38. doi:10.1198/000313002753631330.

White H, Domowitz I (1984). "Nonlinear Regression with Dependent Observations." *Econometrica*, **52**(1), 143–162. doi:10.2307/1911465.

Wickham H (2014). *Advanced R.* CRC Press. doi:10.1201/b17487-2.

Zeileis A (2004). "Econometric Computing with HC and HAC Covariance Matrix Estimators." *Journal of Statistical Software*, **11**(10), 1–17. doi:10.18637/jss.v011.i10.

Zeileis A (2006). "Object-Oriented Computation of Sandwich Estimators." *Journal of Statisical Software*, **16**(9), 1–16. doi:10.18637/jss.v016.i09.

Affiliation:

Bradley C. Saul
NoviSci
PMB 218
201 W Main St, STE 200
Durham, NC 27701, United States of America
E-mail: bradleysaul@gmail.com
URL: http://www.github.com/bsaul

Michael G. Hudgens
The University of North Carolina at Chapel Hill
3107C McGavran-Greenberg Hall
CB #7420, Chapel Hill, NC 27599, United States of America
E-mail: mhudgens@email.unc.edu