Object-oriented Computation of Sandwich Estimators

Achim Zeileis
Wirtschaftsuniversität Wien

Abstract

Conceptual and computational tools for calculating various types of sandwich estimators in different kinds of parametric regression models are discussed.

The theory underlying the package is based on parameter estimates which are computed from estimating functions and for which a central limit theorem holds. The covariance matrix is then of a sandwich type: a slice of meat between two slices of bread, pictorially speaking.

The implementation suggested in the **sandwich** package aims at providing computational translations for these conceptual tools. This is achieved by defining generic functions for extracting the bread and the empirical estimating functions from fitted models and by supplying functions that compute various types of meat fillings from these.

In particular, this should enable computations of sandwich estimators models fitted not only by `lm()`, but also `glm()`, `survreg()`, and maybe `gam()`, `betareg()`, etc.

Keywords: covariance matrix estimators, estimating functions, object orientation, R.

1. Model frame

To fix notations, let us assume we have data in a regression setup, i.e., (y_i, x_i) for $i = 1, \ldots, n$, that follow some distribution that is controlled by a k-dimensional parameter vector θ. In many situations, an estimating function $\psi(\cdot)$ is available for this type of models such that $\mathsf{E}[\psi(y, x, \theta)] = 0$. Then, under certain weak regularity conditions, θ can be estimated using an M-estimator $\hat{\theta}$ implicitely defined as

$$\sum_{i=1}^{n} \psi(y_i, x_i, \hat{\theta}) \quad = \quad 0. \tag{1}$$

This includes in particular maximum likelihood (ML) and ordinary least squares (OLS) estimation, where the estimating function $\psi(\cdot)$ is the derivative of an objective function $\Psi(\cdot)$:

$$\psi(y, x, \theta) \quad = \quad \frac{\partial \Psi(y, x, \theta)}{\partial \theta}. \tag{2}$$

Inference about θ is then typically performed relying on a central limit theorem (CLT) of type

$$\sqrt{n}\,(\hat{\theta} - \theta) \quad \xrightarrow{\mathrm{d}} \quad N(0, S(\theta)), \tag{3}$$

where $\xrightarrow{\mathrm{d}}$ denotes convergence in distribution. For the covariance matrix $S(\theta)$, a sandwich formula can be given

$$S(\theta) \quad = \quad B(\theta)\, M(\theta)\, B(\theta) \tag{4}$$

$$B(\theta) \quad = \quad (\mathsf{E}[-\psi'(y, x, \theta)])^{-1} \tag{5}$$

$$M(\theta) \quad = \quad \mathsf{VAR}[\psi(y, x, \theta)] \tag{6}$$

i.e., the "meat" of the sandwich $M(\theta)$ is the variance of the estimating function and the "bread" is the inverse of the expectation of its first derivative ψ' (again with respect to θ). (Note that

we use the more evocative names S, B and M instead of the more conventional notation $V(\theta) = A(\theta)^{-1}B(\theta)A(\theta)^{-1}$.)

In correctly specified models, estimated by ML, this sandwich expression for $S(\theta)$ can be simplified because both $M(\theta) = B(\theta)^{-1}$ correspond to the Fisher information matrix. Hence, the variance $S(\theta)$ in the CLT from Equation 3 is typically estimated by an empirical version of $B(\theta)$. However, covariance matrix estimates that are in some sense robust against misspecification can usually be obtained by plugging estimates for both $B(\theta)$ and $M(\theta)$ into the sandwich formula for $S(\theta)$ from Equation 4.

Many of the models of interest to us, provide some more structure: the objective function $\Psi(y, x, \theta)$ depends on x and θ in a special way, namely it does only depend on the univariate linear predictor $\eta = x^\top\theta$. Then, the estimating function is of type

$$\psi(y, x, \theta) \quad = \quad \frac{\partial\Psi}{\partial\eta} \cdot \frac{\partial\eta}{\partial\theta} \quad = \quad \frac{\partial\Psi}{\partial\eta} \cdot x.$$

The partial derivative $r(y, \eta) = \partial\Psi(y, \eta)/\partial\eta$ is in some models also called "working residual" correpsonding to the usual residuals in linear regression models. In such models based on linear predictors, the meat of the sandwich can also be written as

$$M(\theta) \quad = \quad x\,\mathsf{VAR}[r(y, x^\top\theta)]\,x^\top. \tag{7}$$

2. Covariance matrix estimators

To make use of the theory outlined in the previous section, the model has to be fitted (i.e., the parameters have to be estimated as in Equation 1) and subsequently inference can be performed relying on the CLT from Equation 3 and employing different types of sandwich estimators for $S(\theta)$.

In R, there are already many model fitting functions which compute estimates $\hat{\theta}$ for a multitude of regression models that can be seen as special cases of the framework outlined above. These typically have a `coef()` method which extracts the parameter estimates $\hat{\theta}$ and an estimate for the covariance matrix is available via a `vcov()` method. Inference relying on this covariance matrix is typically provided in the `summary()` method (partial t or z tests) and `anova()` method (F or χ^2 tests for nested models). The covariance estimate returned by `vcov()` usually relies on the assumption of correctly specified models estimated via ML and hence is simply an empirical version of $B(\theta)$ that is then divided by n. The simplest estimator is

$$\hat{B} \quad = \quad \left(\frac{1}{n}\sum_{i=1}^{n} -\psi'(y_i, x_i, \hat{\theta})\right)^{-1}. \tag{8}$$

Re-using these tools, the **sandwich** package aims at providing other (more robust) sandwich estimators in this general setup. Therefore, we propose the following tools.

The bread

Estimating the bread $B(\theta)$ is usually relatively easy and hence often not dealt with in publications about sandwich estimators. As the simple estimator from Equation 8 is typically already provided in fitted model objects, it suffices to simply provide a generic function `bread()` which has a method for each class of fitted models. For "`lm`" and "`glm`" all the necessary information can be conveniently obtained from the `summary()` method:

```
bread.lm <- function(x, ...)
{
```

```
  sx <- summary(x)
  sx$cov.unscaled * as.vector(sum(sx$df[1:2]))
}
```

Similarly simple functions are provided for "survreg", "coxph and "gam" objects.

The meat

While the bread $B(\theta)$ is almost always estimated by $\hat{B}$ from Equation 8, various different types of estimators are available for the meat $M(\theta)$, most of which are based on the empirical values of estimating functions. Hence, a natural idea for object-oriented implementation of such estimators is the following: provide various functions that compute different estimators for the meat based only on an `estfun()` extractor function that extracts the empirical estimating function from a fitted model object. The `estfun()` method should return an $n \times k$ matrix with

$$\begin{pmatrix} \psi(y_1, x_1, \hat{\theta}) \\ \vdots \\ \psi(y_n, x_n, \hat{\theta}) \end{pmatrix}.$$

Based on this the function `meat()` can simply compute crossproducts for deriving a simple estimator of $M(\theta)$. This corresponds to the Eicker-Huber-White HC estimator and is sometimes also called outer product of gradients estimator. A simplified version of the R code is

```
meat <- function(x)
{
  psi <- estfun(x)
  crossprod(psi)/NROW(psi)
}
```

More elaborate estimators are also available: `meatHAC()` computes an estimate based on the weighted (empirical) autocorrelations of the (empirical) estimating function. An extremely simplified version of `meatHAC()` is

```
meatHAC <- function(x, weights)
{
  psi <- estfun(x)
  n <- NROW(psi)

  rval <- 0.5 * crossprod(psi) * weights[1]
  for(i in 2:length(weights))
    rval <- rval + weights[i] * crossprod(psi[1:(n-i+1),], psi[i:n,])

  (rval + t(rval))/n
}
```

Finally, we would like to provide HC estimators via `meatHC()` for which we need a bit more infrastructure. Relying on Equation 7 we want to estimate $M(\theta)$ by a matrix of type $1/n\, X^\top \Omega X$ where X is the regressor matrix and Ω is a diagonal matrix estimating the variance of $r(y, \eta)$. Various forms for this diagonal matrix Ω have been suggested, they all depend on the vector of the observed $(r(y_1, x_1^\top \hat{\theta}), \ldots, r(y_n, x_n^\top \hat{\theta}))^\top$, the hat values and the degrees of freedom. Instead of basing this on a seperate generic for this type of residuals, it is also possible to recover it from the estimating function. As $\psi(y_i, x_i, \hat{\theta}) = r(y_i, x_i^\top \hat{\theta}) \cdot x_i$, we can simply divide the empirical estimating function by x_i and obtain the residual. Consequently, we only need a way to extract the regressor matrix from the fitted model (via the `model.matrix()` method), the hat values (via the `hatvalues()` method) and the residual degrees of freedom (via the `df.residual()` method). A condensed version of `meatHC()` can then be given as

```
meatHC <- function(x, omega)
{
  X <- if(is.matrix(x$x)) x$x else model.matrix(x)
  n <- NROW(X)

  diaghat <- hatvalues(x)
  df <- df.residual(x)
  if(is.null(df)) df <- n - NCOL(X)

  res <- rowMeans(estfun(x)/X, na.rm = TRUE)

  if(is.function(omega)) omega <- omega(res, diaghat, df)
  rval <- sqrt(omega) * X

  crossprod(rval)/n
}
```

The sandwich

Computing the sandwich is easy given the previous building blocks. Currently, the function `sandwich()` computes an estimate for $1/n\, S(\theta)$ via

```
sandwich <- function(x, bread. = bread, meat. = meat, ...)
{
  if(is.function(bread.)) bread. <- bread.(x)
  if(is.function(meat.)) meat. <- meat.(x, ...)
  1/NROW(estfun(x)) * (bread. %*% meat. %*% bread.)
}
```

and `meat.` could also be set to `meatHAC` or `meatHC`.

Therefore, all that a useR/developeR would have to do to make a new class of models, "`foo`" say, fit for this framework is: provide an `estfun()` method `estfun.`*foo*`()` and a `bread()` method `bread.`*foo*`()`. See also Figure 1.

Only for HC estimators (other than HC0 and HC1 which are available via `meat()`), it has to be assured in addtion that

- the model only depends on a linear predictor (this cannot be easily checked by the software, but has to be done by the user),
- the model matrix X is available (via a `model.matrix.`*foo*`()` method),
- a `hatvalues.`*foo*`()` method exists (for HC2–HC4), and
- a `df.residual.`*foo*`()` method exists (or the default $n - k$ are correct).

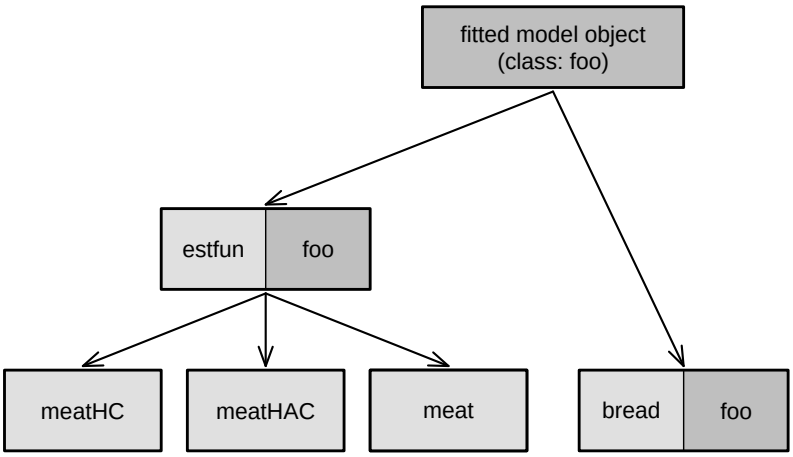

Figure 1: Structure of sandwich estimators