Object-oriented Computation of Sandwich Estimators

Achim Zeileis
Wirtschaftsuniversität Wien

Abstract

Some ideas about generalizing the tools available in **sandwich** to more models, in particular fully supporting `glm()` and maybe `survreg()`, `gam()`, `betareg()`, ...

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

1. Theory

To fix notations, let us assume we have data in a regression setup, i.e., (y_i, x_i) for $i = 1, \dots, n$, 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 $\mathsf{E}[\psi(y, x, \theta)]$. Then, under certain weak regularity conditions, θ can be estimated using an M-estimator $\hat{\theta}$ implicitely defined as

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

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 $\Psi(\cdot)$:

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

Inference about θ is then typically performed relying on a central limit theorem of typ.

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

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

$$\begin{aligned} S(\theta) &= B(\theta)\, M(\theta)\, B(\theta) \\ B(\theta) &= \left(\mathsf{E}[-\psi'(y, x, \theta)]\right)^{-1} \\ M(\theta) &= \mathsf{VAR}[\psi(y, x, \theta)] \end{aligned}$$

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.

2. Implementation

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 in the previous section. Many of these functions already have a `vcov()` method, but this typically relies on the fact that for (correctly specified) models estimated via ML, the bread and meat are equal

$B(\theta) = M(\theta)$. Therefore, standard inference in these models (typically as reported by `summary()`) is essentially based on the estimated covariance matrix $1/n\, B(\hat{\theta})$.

To be able to compute (more robust) sandwich estimators in this general setup, we propose the following tools.

The meat

To be able to compute various estimators for $M(\theta)$, we need the empirical values for the estimating function: the generic function `estfun()` extracts these values and should return an $n \times k$ matrix with

$$\left(\begin{array}{c} \psi(y_1, x_1, \hat{\theta}) \\ \vdots \\ \psi(y_n, x_n, \hat{\theta}) \end{array} \right).$$

From this the function `meat()` can simply compute crossproducts for deriving a naive estimator of $M(\theta)$. *(does this need to be generic? probably no)* Alternative, to this naive estimator `meatHAC()` can compute an estimate based on the (empirical) autocorrelations of the estimating function.

Probably, `meatHC()` can also be written in such a way that it utilizes `estfun()` although I'm not sure what the underlying theoretical assumptions are for this. But I think it suffices that the estimating functions are of type $r \cdot X$ where X is the regressor matrix and r is the derivative of the objective function with respect to the linear predictor. For linear regression models this is simply the vector of residuals and for GLMs it is typically referred to as the working residuals.

The bread

Estimating $B(\theta)$ is typically easier and not dealt with in most of the publications (I know) about sandwich estimators. Currently, I've written two approaches how to derive it, but maybe one is enough. The first, simpler solution is to provide a generic function `bread()` which should have a method for each model class. Alternatively, one could try to provide an `estfunDeriv()` generic which should have a method for each model class. It should return an $k \times k \times n$ array with

$$\left(\begin{array}{c} -\psi'(y_1, x_1, \hat{\theta}) \\ \vdots \\ -\psi'(y_n, x_n, \hat{\theta}) \end{array} \right).$$

By taking the mean of the n dimension (via `apply()`) and then computing the inverse (via `solve()`), one can compute a default estimate for $B(\theta)$ but this is typically burdensome as the result is already stored (as slot `cov.unscaled`) in many fitted models.

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, ...)
  n <- NROW(estfun(x))
  return(1/n * (bread. %*% meat. %*% bread.))
}
```

and `meat.` could be set to `meatHAC` or `meatHC`. All that a useR/developeR would have to do to make a new class of models fit for this framework is: provide an `estfun()` method and a `bread()` method. If the more complicated second variant for `bread()` is used, an `estfunDeriv()` method could be provided in addition or as an alternative to the `bread()` method.

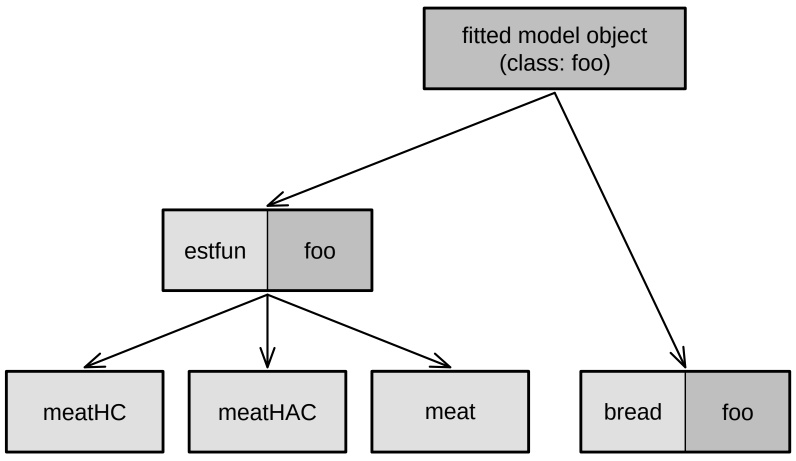

Figure 1: Structure of sandwich estimators