Open-Source Econometric Computing in R

Achim Zeileis

http://statmath.wu-wien.ac.at/~zeileis/

Overview

- Implementing econometric methodology:
  - Why should we write software (*and make it available*)?
  - Why should it be open-source software?
  - What should be the guiding principles for implementation?
  - Why R?

- Robust covariance matrix estimators
  - Sandwich estimators
  - Implementation in **sandwich**
  - Illustrations

Why software?

Authors of econometric methodology usually have an implementation for own applications and running simulations and benchmarks, *but not necessarily in production quality*.

Why should they be interested in taking the extra effort to adapt them to more general situations, document it and make it available to others?

Supplying software that is sufficiently easy to use is an excellent way of *communicating ideas and concepts* to researchers and practitioners.

Given the description of an excellent method and code for a good one, you choose . . . ?

Why open source?

Claerbout's principle

> An article about computational science in a scientific publication is *not* the scholarship itself, it is merely *advertising* of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.

To evaluate the correctness of all the results in such an article, the source code must also be available for inspection. Only this way gradual refinement of computational (and conceptual) tools is possible.

Implementation principles

Task: Turn conceptual tools into computational tools

Goals: desirable features

- easy to use,
- numerically reliable,
- computationally efficient,
- flexible and extensible,
- re-usable components,
- object oriented,
- reflect features of the conceptual method.

Implementation principles

Problem: often antagonistic. E.g., computational efficiency vs. extensibility.

Guiding principle: The implementation should be guided by the properties of the underlying methods while trying to ensure as much efficiency and accuracy as possible.

The resulting functions should do what we think an algorithm does conceptually.

Implementation principles

In practice: Many implementations are still guided by the limitations that programming languages used to have (and some still have) where everything has to be represented by numeric vectors and matrices.

What language features are helpful for improving this?

Implementation principles

Object orientation: Create (potentially complex) objects that represent an abstraction of a procedure or type of data. Methods performing typical tasks can be implemented.

Functions as first-class objects: Functions are a basic data type that can be passed to and returned by another function.

Lexical scope: more precisely *nested lexically scoped functions*. Returned functions (e.g., prediction methods) can have free variables stored in the function closure.

Implementation principles

Compiled code: Combine convenience of using interpreted code and efficiency of compiled code by (byte) compilation or dynamic linking of compiled code.

Re-usable components: The programming environment should provide procedures that the implementation can build on. Likewise, the implementation should create objects that can be re-used in other programs.

Why R?

R offers all these features and more:

- R is a full-featured interactive computational environment for data analysis, inference and visualization.
- R is an open-source project, released under GPL.
- Developed for the Unix, Windows and Macintosh families of operating systems by the R Development Core Team.
- Offers several means of object orientation, including S3 and S4 classes.

Why R?

- Everything in R is an object, including functions and even function calls.
- Nested functions are lexically scoped.
- Allows for dynamic linking of C, C++ or FORTRAN code.
- Highly extensible with a fast-growing list of add-on packages.

Why R?

Software delivery is particularly easy:

R itself and $\sim$1000 packages are available (most of them under the GPL) from the Comprehensive R Archive Network (CRAN):

http://CRAN.R-project.org/

and can easily be installed from within R via, e.g.

```
R> install.packages("sandwich")
```

Why R?

CRAN Task View: Econometrics

- **linear models:** OLS estimation, diagnostic tests, robust regression, simultaneous equations.
- **microeconometrics:** binary data (GLMs, logit, probit), count data models (poisson, negbin, zero-inflated, hurdle), censored data (tobit).
- **time series models:** ARIMA, structural time series models, unit root, cointegration, structural change.
- **basic infrastructure:** matrix manipulations, optimization, time/date and time series classes.

`http://CRAN.R-project.org/src/contrib/Views/`

Sandwich estimators

Inference for models estimated by estimating equations

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

(including maximum likelihood and least squares estimators) is typically based on a central limit theorem

$$\sqrt{n}\,(\widehat{\theta} - \theta) \quad \xrightarrow{\mathsf{d}} \quad \mathcal{N}(0, S(\theta)),$$

where the covariance matrix is of a sandwich form:

$$\begin{aligned} \text{sandwich:} \quad S(\theta) &= B(\theta)\, M(\theta)\, B(\theta), \\ \text{bread:} \quad B(\theta) &= \mathsf{E}[-\psi'(y, x, \theta)]^{-1}, \\ \text{meat:} \quad M(\theta) &= \mathsf{VAR}[\psi(y, x, \theta)]. \end{aligned}$$

Sandwich estimators

In correctly specified models estimated by ML, $M(\theta) = B(\theta)^{-1}$ is the Fisher information matrix and the covariance matrix is typically based only on an estimator for the bread $\widehat{B}$.

If the estimating functions $\psi(y, x, \theta)$ are correctly specified, but the remaining likelihood is not, a more robust covariance matrix estimate can be obtained by estimating the full sandwich and not only the bread.

Sandwich estimators

HC and HAC sandwich estimate of the meat $\widehat{M}$

$$\widehat{M} \quad = \quad n^{-1} \sum_{i,j=1}^{n} w_{|i-j|} \, \psi(y_i, x_i, \widehat{\theta}) \, \psi(y_j, x_j, \widehat{\theta})^\top$$

where the weights are

- cross-section data: $w_0 = 1$, $w_i = 0$ $(i > 0)$,
- time-series data: w_k chosen by a kernel function (plus bandwidth selection).

Special cases: Eicker-Huber-White sandwich estimator, Andrews kernel HAC estimator, Newey-West estimator.

Implementation in sandwich

Translation to R: Provide functions (similar to `vcov()`)

```
sandwich(obj)
  vcovHC(obj, ...)
 vcovHAC(obj, weights, ...)
```

that work for (in principle) arbitrary fitted model objects `obj`.

- uses the `estfun(obj)` method to extract the empirical values of the estimating function,
- expects `weights` to be either a vector of weights, or a function that computes the weights in a data-driven way from `obj`.
- can deal with `weights` functions that again rely on a function for choosing the bandwidth.

Implementation in sandwich

This implementation uses

- **object orientation**: different models can be plugged in as long as there is a `estfun()` method.
- **functions as first-class objects**: strategies for selecting weights (and bandwidths) can be implemented as functions and are then evaluated by `vcovHAC`.
- **lexical scope**: these functions can also be defined locally in convenience wrapper functions.
- **re-usable components**: by taking fitted objects it relies on existing model fitting functions and returns a covariance estimators that can easily be re-used in inference functions.

Illustrations

Example: time series regression (investment equation from Greene, 1993).

```
R> fm <- lm(RealInv ~ RealGNP + RealInt, data = Investment)
R> vcovHAC(fm)
```

```
             (Intercept)       RealGNP     RealInt
(Intercept) 615.5987887 -0.5679537232  9.24170225
RealGNP      -0.5679537  0.0005448671 -0.02222363
RealInt       9.2417023 -0.0222236251 14.53971300
```

Illustrations

Re-use in partial Wald tests:

```
R> coeftest(fm, vcov = vcovHAC(fm))


t test of coefficients:

              Estimate Std. Error t value  Pr(>|t|)
(Intercept) -12.533601  24.811263 -0.5052    0.6203
RealGNP       0.169136   0.023342  7.2459 1.955e-06 ***
RealInt      -1.001438   3.813098 -0.2626    0.7962
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Illustrations

Or even simpler:

```
R> coeftest(fm, vcov = vcovHAC)


t test of coefficients:

              Estimate Std. Error t value  Pr(>|t|)
(Intercept) -12.533601  24.811263 -0.5052    0.6203
RealGNP       0.169136   0.023342  7.2459 1.955e-06 ***
RealInt      -1.001438   3.813098 -0.2626    0.7962
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Illustrations

Example: demand for medical care count data regression (Deb and Trivedi, 1997).

Simple poisson model

```
R> fm1 <- glm(ofp ~ numchron + health + gender + privins,
+     data = DebTrivedi, family = poisson)
```

and negative binomial hurdle model

```
R> fm2 <- hurdle(ofp ~ numchron + health + gender + privins |
+     numchron + gender + privins, data = DebTrivedi,
+     dist = "negbin")
```

Illustrations

The hurdle is clearly superior as a likelihood model

```
R> logLik(fm1)
```

```
'log Lik.' -18390.07 (df=6)
```

```
R> logLik(fm2)
```

```
'log Lik.' -12174.09 (df=12)
```

but as regression models for the mean both lead to similar results. Account for likelihood misspecification of poisson model by sandwich standard errors.

Illustrations

```
R> coeftest(fm1, vcov = sandwich)


z test of coefficients:

                 Estimate Std. Error z value  Pr(>|z|)
(Intercept)      1.253836   0.044976 27.8780 < 2.2e-16 ***
numchron         0.167858   0.012217 13.7392 < 2.2e-16 ***
healthexcellent -0.356653   0.077575 -4.5975 4.275e-06 ***
healthpoor       0.288914   0.054659  5.2858 1.252e-07 ***
gendermale      -0.109147   0.035898 -3.0405  0.002362 **
privinsyes       0.285169   0.042250  6.7496 1.483e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Illustrations

```
R> coeftest(fm2)


z test of coefficients:

                         Estimate Std. Error z value  Pr(>|z|)
count_(Intercept)        1.368469   0.046516 29.4192 < 2.2e-16 ***
count_numchron           0.146406   0.012702 11.5264 < 2.2e-16 ***
count_healthexcellent -0.347083   0.067535 -5.1393 2.757e-07 ***
count_healthpoor         0.353430   0.048641  7.2661 3.700e-13 ***
count_gendermale        -0.051415   0.033260 -1.5458    0.1221
count_privinsyes         0.193528   0.041439  4.6702 3.009e-06 ***
zero_(Intercept)         0.510442   0.097952  5.2112 1.877e-07 ***
zero_numchron            0.565456   0.042770 13.2209 < 2.2e-16 ***
zero_gendermale         -0.408241   0.086993 -4.6928 2.695e-06 ***
zero_privinsyes          0.904685   0.093858  9.6389 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

References

Achim Zeileis (2004). "Econometric Computing with HC and HAC Covariance Matrix Estimators," *Journal of Statistical Software*, **11**(10). URL http://www.jstatsoft.org/v11/i10/

Achim Zeileis (2005). "Implementing a Class of Structural Change Tests: An Econometric Computing Approach," *Computational Statistics & Data Analysis*, **50**(11), 2987–3008. doi:10.1016/j.csda.2005.07.001

Achim Zeileis (2006). "Object-oriented Computation of Sandwich Estimators," *Journal of Statistical Software*, **16**(9). URL http://www.jstatsoft.org/v16/i09/

Achim Zeileis, Christian Kleiber, and Simon Jackman (2007). "Count Data Regression in R," Forthcoming.

Christian Kleiber and Achim Zeileis (2008?). *Applied Econometrics with R*. Springer-Verlag, New York.