Econometric Computing with HC and HAC Covariance Matrix Estimators

Achim Zeileis
Wirtschaftsuniversität Wien

Abstract

sandwich

Keywords: covariance matrix estimator, heteroskedasticity, autocorrelation, estimating functions, econometric computing, R.

1. Introduction

important for applied econometrics

R Development Core Team (2004) Cribari-Neto and Zarkos (2003)

stress econometric computing

reusable components

covariance matrices not only as options to certain test but as stand-alone functions which can be plugged into various inference procedures

explain heteroskedasticity consistent (HC) and heteroskedasticity and autocorrelation consistent (HAC) of unknown form

sandwich provides functions `vcovHC` and `vcovHAC` implementing general classes of HC and HAC estimators. The names of the functions are chosen to correspond to `vcov`, R's generic function for extracting covariance matrices from fitted model objects.

kurzer literatur review um daraus die Fkt zu motivieren

ein paar Sachen in Fox (aber nicht so flexibel), sonst nix Fox (2002)

2. The linear regression model

To fix notations, we consider the linear regression model

$$y_i \quad = \quad x_i^\top \beta + u_i \qquad (i = 1, \dots, n), \tag{1}$$

with dependent variable y_i, k-dimensional regressor x_i with coefficient vector β and error term u_i. In the usual matrix notation comprising all n observations this can be formulated as $y = X\beta + u$.

In the general linear model, it is typically assumed that the errors have zero mean and variance $\mathsf{VAR}[u] = \Omega$. Under suitable regularity conditions (see e.g., Greene 1993), the coefficients β can be consistently estimated by ordinary least squares (OLS) giving the well-known OLS estimator $\hat\beta$ with corresponding OLS residuals $\hat u_i$:

$$\hat\beta \quad = \quad \left(X^\top X\right)^{-1} X^\top y \tag{2}$$

$$\hat u \quad = \quad (I_n - H)\, u \;=\; (I_n - X\left(X^\top X\right)^{-1} X^\top)\, u \tag{3}$$

where I_n is the n-dimensional identity matrix and H is usually called hat matrix. The estimates $\hat\beta$ are unbiased and are asymptotically normal with covariance matrix Ψ which is usually denoted in one of the two following ways:

$$\Psi \;=\; \mathsf{VAR}[\hat{\beta}] \;=\; \left(X^\top X\right)^{-1} X^\top \Omega X \left(X^\top X\right)^{-1} \qquad (4)$$

$$\;=\; \frac{1}{n}\left(\frac{1}{n}X^\top X\right)^{-1} \Phi \left(\frac{1}{n}X^\top X\right)^{-1} \qquad (5)$$

where $\Phi = n^{-1} X^\top \Omega X^\top$ is essiantlly the covariance matrix of the estimating functions $V_i(\beta) = x_i(y_i - x_i^\top \beta)$. The estimating functions evaluated at the parameter estimates $\hat{V}_i = V_i(\hat{\beta})$ have then sum zero.

For doing inference in the linear regression model, it is essential to have a consistent estimator for Ψ—the most common inferential task being to test whether one of the coefficients β_j is zero which is is usually assessed using the t ratio $\beta_j/\sqrt{\hat{\Psi}_{jj}}$. What kind of estimator is used for Ψ depends on the assumptions about Ω that are used: In the classical linear model independent and homoskedastic errors with variance σ^2 are assumed yielding $\Omega = \sigma^2 I_n$ and $\Psi = \sigma^2 (X^\top X)^{-1}$ which can be easily estimated by plugging in the usual OLS estimator $\hat{\sigma}^2 = (n-k)^{-1} \sum_{i=1}^n \hat{u}_i^2$. But if the independence and/or homoskedasticity assumption is violated, inference based on this estimator $\hat{\Psi}_{\mathrm{const}} = \hat{\sigma}(X^\top X)^{-1}$ will be biased. HC and HAC estimators tackle this problem by plugging an estimate $\hat{\Omega}$ or $\hat{\Phi}$ into (4) or (5) respectively which are consistent in the presence of heteroskedasticity and autocorrelation respectively. Such estimators and their implementation are described in the following section.

3. Estimating the covariance matrix Ψ

3.1. Dealing with heteroskedasticity

If it is assumed that the errors u_i are independent but not necessary homoskedastic—a situation which typically arises with cross-sectional data—their covariance matrix Ω is diagonal but has nonconstant diagonal elements. Therefore, various HC estimators $\hat{\Psi}_{\mathrm{HC}}$ have been suggested which are constructed by plugging an estimate of type $\hat{\Omega} = \mathrm{diag}(\omega_1, \dots, \omega_n)$ into Equation (4). These estimators differ in their choice of the ω_i, an overview of the most important cases is giving in the following:

$$\begin{aligned}
\text{const}: \quad \omega_i &= \hat{\sigma}^2 \\
\text{HC0}: \quad \omega_i &= \hat{u}_i^2 \\
\text{HC1}: \quad \omega_i &= \frac{n}{n-k}\,\hat{u}_i^2 \\
\text{HC2}: \quad \omega_i &= \frac{\hat{u}_i^2}{1-h_i} \\
\text{HC3}: \quad \omega_i &= \frac{\hat{u}_i^2}{(1-h_i)^2} \\
\text{HC4}: \quad \omega_i &= \frac{\hat{u}_i^2}{(1-h_i)^{\delta_i}}
\end{aligned}$$

where $h_i = H_{ii}$ are the diagonal elements of the hat matrix and $\delta_i = \min\{4, h_i/\bar{h}\}$.

The first equation above yields the standard estimator $\hat{\Psi}_{\mathrm{const}}$ for homoskedastic errors. All others produce different kinds of HC estimators. The estimator HC0 was suggested in the econometrics literature by White (1980) and is justified by asymptotic arguments. The estimators HC1, HC2 and HC3 were suggested by MacKinnon and White (1985) to improve the performance in small samples. A more extensive study of small sample behaviour was carried out by Long and Ervin (2000) which arrive at the conclusion that HC3 provides the best performance in small samples as it gives less weight to influential observations. Recently, Cribari-Neto (2004) suggested the estimator HC4 to further improve small sample performance, especially in the presence of influential observations.

To translate the conceptual properties—determining the covariance matrix estimator $\hat{\Psi}_{\mathrm{HC}}$ by $\omega = (\omega_1, \dots, \omega_n)^\top$—of this class of HC estimators into a computational tool, a function is required which takes a fitted regression and the diagonal elements ω as input and returns $\hat{\Psi}_{\mathrm{HC}}$. In **sandwich** this is implemented in the function `vcovHC` which takes the following arguments:

```
vcovHC(lmobj, type = "HC3", omega = NULL, ...)
```

The first argument `lmobj` is an object as returned by `lm()`, R's standard function for fitting linear regression models via OLS. The argument `omega` can either be the vector ω or a function for data-driven computation of ω based on the residuals $\hat{u}$, the diagonal elements of the hat matrix h and the residual degrees of freedom $n-k$. Thus, it has to be of the form `omega(residuals, diaghat, df)`. As an example, for computing HC3 `omega` would have to be set to `function(residuals, diaghat, df) residuals^2`

As a convenience interface, a `type` argument can be set to `"const"`, `"HC0"` (or equivalently `"HC"`), `"HC1"`, `"HC2"`, `"HC3"` (the default) or `"HC4"` and then `vcovHC` uses the corresponding `omega` functions. As soon as `omega` is set, `type` is ignored.

In summary, by specfying ω—either as a vector or as a function—`vcovHC` can compute arbitrary HC covariance matrix estimates from the class of estimators outlined above. In Section 4, it will be illustrated how this function can be used as a building block when doing inference in linear regression models.

3.2. Dealing with autocorrelation

If the error terms are not independent, Ω is not diagonal and without further specification of a parametic model for the type of dependence it is typically burdensome to estimate Ω directly. However, if the form of heteroskedasticity and autocorrelation is unknown, a solution to this problem is to estimate Φ instead which is essentially the covariance matrix of the estimating functions[1]. This is what HAC estimators do: $\hat{\Psi}_{\mathrm{HAC}}$ is computed by plugging an estimate $\hat{\Phi}$ into Equation (5) with

$$\hat{\Phi} \quad = \quad \frac{1}{n} \sum_{i,j=1}^{n} w_{|i-j|} \hat{V}_i \hat{V}_j^\top \tag{6}$$

where $w = (w_0, \dots, w_{n-1})^\top$ is a vector of weights. An additional finite sample adjustment can be applied by multiplication with $n/(n-k)$. The motivation for this approach is that the autocorrelations should decrease with increasing lag $\ell = |i - j|$—otherwise β can typically not be estimated consistently by OLS—such that the absolute weights $|w_\ell|$ should also decrease. Starting from White and Domowitz (1984) and Newey and West (1987), different choices for the vector of weights w have been suggested in the econometrics literature which have been placed by Andrews (1991) in a more general framework of choosing the weights by kernel functions with automatic bandwidth selection. Andrews and Monahan (1992) show that the bias of the estimators can be reduced by prewhitening the estimating functions $\hat{V}_i$ using a vector autoregression (VAR) of order p. Lumley and Heagerty (1999) suggest an adaptive weighting scheme where the weights are chosen based on the estimated autocorrelations of the residuals $\hat{u}_i$.

As all the estimators mentioned above are of the form (6), a natural implementation for this class of HAC estimators is the following:

```
vcovHAC(lmobj, weights,
  prewhite = FALSE, adjust = TRUE, sandwich = TRUE,
  order.by, ar.method, data)
```

[1] As the estimating functions are used, this approach is not only feasible in linear models estimated by OLS, but also in nonlinear models using other estimating functions such as maximum likelihood (ML), generalized methods of moments (GMM) or Quasi-ML.

where the most important arguments are again the fitted `lmobj` from which the estimating functions $\hat{V}_i$ can easily be extracted using the generic function `estfun(lmobj)` and the argument `weights` which specifys w. This can be either the vector w directly or a function to compute it from `lmobj`. If `weights` is a vector shorter than n, it is assumed that the remaining weights are zero. The argument `prewhite` specifies wether prewhitening should be used or not—the order p is set to `as.numeric(prewhite)`—and `adjust` determines wether a finite sample correction by multiplication with $n/(n-k)$ should be made or not. By setting `sandwich` it can be controlled wether the full sandwich estimator $\hat{\Psi}_{\mathrm{HAC}}$ or only the "meat" $\hat{\Phi}$ of the sandwich should be returned. The remaining arguments are a bit more technical: `order.by` specifies by which variable the data should be ordered (the default is that they are already ordered, as is natural with time series data), which `ar.method` should be used for fitting the VAR(p) model (the default is OLS) and `data` provides a data frame from which `order.by` can be taken (the default is the environment from which `vcovHAC` is called).[2] More detailed technical documentation of these and other arguments of the functions described in this section are available in the reference manual included in **sandwich**.

For the most important estimators from the literature mentioned above there are functions for computing the corresponding weights readily available in **sandwich**. They are all of the form `weights(lmobj, order.by, prewhite, ar.method, data)`, i.e., functions that compute the weights depending on the fitted model object `lmobj` and the arguments `order.by`, `prewhite`, `data` which are only needed for ordering and prewhitening. The function `weightsAndrews` implements the class of weights of Andrews (1991) and `weightsLumley` implements the class of weights of Lumley and Heagerty (1999). Both functions have convenience interfaces: `kernHAC` calls `vcovHAC` with `weightsAndrews` (and different defaults for some parameters) and `weave` calls `vcovHAC` with `weightsLumley` (where `weave` stands for <u>w</u>eighted <u>e</u>mpirical <u>a</u>daptive <u>v</u>ariance <u>e</u>stimators). Finally, a third convenience interface to `vcovHAC` is available for computing the estimator of Newey and West (1987).

Newey and West (1987) suggested to use linearly decaying weights

$$w_\ell \quad = \quad 1 - \frac{\ell}{L+1} \tag{7}$$

where L is the maximum lag, all other weights are zero. This is implemented in the function `NeweyWest(lmobj, lag, ...)` where `lag` specifies L and `...` are (here, and in the following) further arguments passed to other functions, detailed information is always available in the reference manual.

Andrews (1991) placed this and other estimators in a more general class of kernel-based HAC estimators with weights of the form $w_\ell = K(\ell/B)$ where $K(\cdot)$ is a kernel function and B the bandwidth parameter used. The kernel functions considered are the truncated, Bartlett, Parzen, Tukey-Hanning and quadratic spectral kernel which are depicted in Figure 1. The Bartlett kernel leads to the weights used by Newey and West (1987) in Equation (7) when the bandwidth B is set to $L+1$. The kernel recommended by Andrews (1991) and probably most used in the literature is the quadratic spectral kernel which leads to the following weights:

$$w_\ell \quad = \quad \frac{3}{z^2}\left(\frac{\sin(z)}{z} - \cos(z)\right), \tag{8}$$

where $z = 6\pi/5 \cdot \ell/B$.

All of the kernel weights mentioned above are available in `weightsAndrews(lmobj, kernel, bw, ...)` where `kernel` specifies one of the kernels via a character string (`"Truncated"`, `"Bartlett"`, `"Parzen"`, `"Tukey-Hanning"` or `"Quadratic Spectral"`) and `bw` the bandwidth either as a scalar or as a function. The automatic bandwidth selection described in Andrews (1991) via AR(1) or

[2] Note, that not only HAC estimators for fitted linear models can be computed with `vcovHAC`. If there is an `estfun` method extracting the estimating functions from the fitted model supplied, the estimate $\hat{\Phi}$ can be computed for arbitrary fitted models. To compute the full sandwich $\hat{\Psi}_{\mathrm{HAC}}$, the `summary` method has to have a slot `cov.unscaled` in addition.

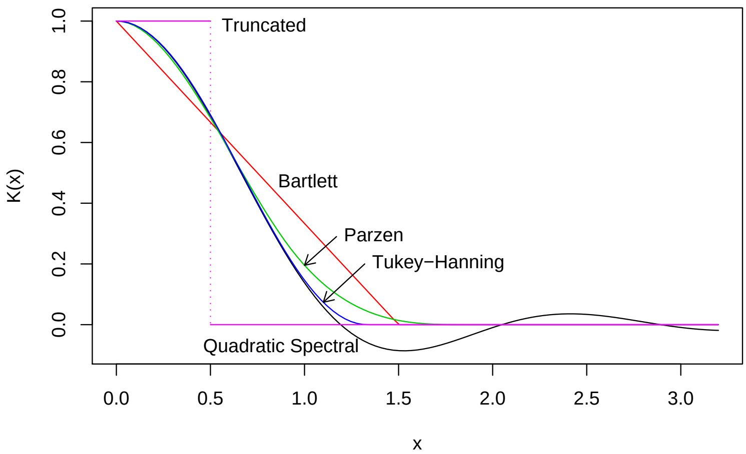

Figure 1: Kernel functions for kernel-based HAC estimation

ARMA(1,1) approximations is implemented in a function `bwAndrews` which is set as the default in `weightsAndrews`. As the flexibility of this conceptual framework of estimators leads to a lot of knobs and switches in the computational tools, a convenience function `kernHAC` for kernel-based HAC estimation has been added to **sandwich** that calls `vcovHAC` with `weightsAndrews` and `bwAndrews` with defaults as motivated by Andrews (1991) and Andrews and Monahan (1992): by default, it computes a quadratic spectral kernel HAC estimator with VAR(1) prewhitening and automatic bandwidth selection based on an AR(1) approximation. Of course, all the options described above can also be changed by the user when calling `kernHAC`.

Lumley and Heagerty (1999) suggested a different approach for specifying the weights in (6) based on some estimate $\hat{\varrho}_\ell$ of the autocorrelation of the residuals $\hat{u}_i$ at lag $l = 1, \dots, n-1$. They suggest either to use truncated weights $w_\ell = I\{n\,\hat{\varrho}_\ell^2 > C\}$ (where $I(\cdot)$ is the indicator function) or smoothed weights $w_\ell = \min\{1, C\,n\,\hat{\varrho}_\ell^2\}$, where for both a suitable constant C has to be specified. Lumley and Heagerty (1999) suggest using a default of $C = 4$ and $C = 1$ for the truncated and smoothed weights respectively. Note, that the truncated weights are equivalent to the truncated kernel from the framework of Andrews (1991) using a different method for computing the truncation lag. To ensure that the weights $|w_\ell|$ are decreasing, the autocorrelations have to be decreasing for increasing lag ℓ which can be achieved by using isotonic regression methods. In **sandwich** these weighting schemes are implemented in a function `weightsLumley` with a convenience interface `weave` which again sets up the weights and then calls `vcovHAC`. Its most important arguments are `weave(lmobj, method, C, ...)` where `method` can be set either to `"truncate"` or `"smooth"` and `C` is by default 4 or 1 respectively.

To sum up, `vcovHAC` provides a simple yet flexible interface for general HAC estimation as defined in Equation (6) and arbitrary weight vectors can be easily supplied. But data-driven computation of these weights and especially their bandwidth parameter might not be as simple, therefore two strategies suggested in the literature are also readily available in **strucchange**: First, the weighting scheme introduced by Andrews (1991) for kernel-based HAC estimation with automatic bandwidth selection is implemented in `weightsAndrews` and corresponding convenience interface `kernHAC`. Second, the weighted empirical adaptive variance estimation scheme suggested by Lumley and Heagerty (1999) is available in `weightsLumley` with convenience interface `weave`.

How these functions can be easily used in applications is illustrated in the following section.

4. Applications and illustrations

In econometric analyses, the practitioner is only seldom interested in the covariance matrix $\hat{\Psi}$ (or $\hat{\Omega}$ or $\hat{\Phi}$) *per se*, but mainly wants to compute them to use them for inferential procedures. Therefore, it is important that the functions `vcovHC` and `vcovHAC` described in the previous section can be easily supplied to other procedures such the user does not necessarily have to compute the variances in advance.

A typical field of application for HC and HAC covariances are partial t or z tests for test whether a parameter β_j is significantly different from zero. These tests are based on the t ratio $\beta_j/\sqrt{\hat{\Psi}_{jj}}$ and either use the asymptotic normal distribution or the t distribution with $n-k$ degrees of freedom for computing p values. This procedure is available in the R package **lmtest** (Zeileis and Hothorn 2002) in the generic function `coeftest` with a method applicable to fitted `"lm"` objects.

```
coeftest(lmobj, vcov = NULL, df = NULL, ...)
```

where `vcov` specifies the covariances either as a matrix (corresponding to the covariance matrix estimate) or as a function computing it from `lmobj` (corresponding to the covariance matrix estimator). By default, it uses the `vcov` method which computes $\hat{\Psi}_{\text{const}}$ assuming spherical errors. The `df` argument determines the degrees of freedom: if `df` is finite and positive, a t distribution with `df` degrees of freedom is used, otherwise a normal approximation is employed. The default is to set `df` to $n-k$.

This and other methods will be used using three real-world data sets: testing coefficients in two models from Greene (1993) and a structural change problem from Bai and Perron (2003).

To make the results exactly reproducible for the reader, the commands for the inferential procedures is given along with their output within the text. All full list of commands, including those which produce the figures in the text, are provided (without output) in the appendix along with the version of R and the packages used. Before we start with the examples, the **sandwich** and **lmtest** package have to be loaded:

```
R> library(sandwich)
R> library(lmtest)
```

4.1. Testing coefficients in cross-sectional data

A quadratic regression model for per capita expenditures on public schools explained by per capita income in the United States in 1979 has been analyzed by Greene (1993) and re-analyzed in Cribari-Neto (2004). The corresponding cross-sectional data for the 51 US states is given in Table 14.1 in Greene (1993) and available in **sandwich** in the data frame `PublicSchools` which can be loaded by:

```
R> data(PublicSchools)
R> ps <- na.omit(PublicSchools)
R> ps$Income <- ps$Income * 1e-04
```

where the second line omits a missing value (`NA`) in Wisconsin and assigns the result to a new data frame `ps` and the third line transforms the income to be in USD $10,000$. The quadratic regression can now easily be fit using the function `lm()` which fits linear regression models specified by a symbolic formula via OLS.

```
R> fm.ps <- lm(Expenditure ~ Income + I(Income^2), data = ps)
```

The fitted `"lm"` object `fm.ps` now contains the regression of the variable `Expenditure` on the variable `Income` and its sqared value, both variables are taken from the data frame `ps`. The question in this data set is whether the quadratic term is really needed, i.e., whether the coefficient of `I(Income^2)` is significantly different from zero. The partial quasi-t tests (or z tests) for all coefficients can be computed using the function `coeftest`. Greene (1993) assesses the significance using the HC0 estimator of White (1980).

```
R> coeftest(fm.ps, df = Inf, vcov = vcovHC(fm.ps, type = "HC0"))

z test of coefficients of "lm" object 'fm.ps':

             Estimate Std. Error z value Pr(>|z|)
(Intercept)    832.91     460.89  1.8072  0.07073 .
Income       -1834.20    1243.04 -1.4756  0.14006
I(Income^2)   1587.04     829.99  1.9121  0.05586 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The `vcov` argument specifies the covariance matrix as a matrix (as opposed to a function) which is returned by `vcovHC(fm.ps, type = "HC0")`. As `df` is set to infinity (`Inf`) a normal approximation is used for computing the p values which seem to suggest that the quadratic term might be weakly significant. In his analysis, Cribari-Neto (2004) uses his HC4 estimator (among others) giving the following result:

```
R> coeftest(fm.ps, df = Inf, vcov = vcovHC(fm.ps, type = "HC4"))

z test of coefficients of "lm" object 'fm.ps':

             Estimate Std. Error z value Pr(>|z|)
(Intercept)    832.91    3008.01  0.2769   0.7819
Income       -1834.20    8183.19 -0.2241   0.8226
I(Income^2)   1587.04    5488.93  0.2891   0.7725
```

The quadratic term is clearly non-significant. The reason for this result is depicted in Figure 2 which shows the data along with the fitted linear and quadratic model—the latter being obviously highly influenced by a single outlier: Alaska. The reason for this improved performance of the HC4 estimator is that the HC0 but not the HC4 estimator is highly influenced by high leverage points.

4.2. Testing coefficients in time-series data

Greene (1993) also anayzes a time-series regression model based on robust covariance matrix estimates: his Table 15.1 provides data on the nominal gross national product (GNP), nominal gross private domestic investment, a price index and an interest rate which is used to formulate a model that explains real investment by real GNP and real interest. The corresponding transformed variable `RealInv`, `RalGNP` and `RealInt` are stored in the data frame `Investment` in **sandwich** which can be loaded by:

```
R> data(Investment)
```

Subsequently, the fitted linear regression model is computed by:

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

and the significance of the coefficients can again be assessed by partial z tests using `coeftest()`. Greene (1993) uses the estimator of Newey and West (1987) for this purpose which is in R passed as a matrix (as opposed to a function) to `coeftest`.

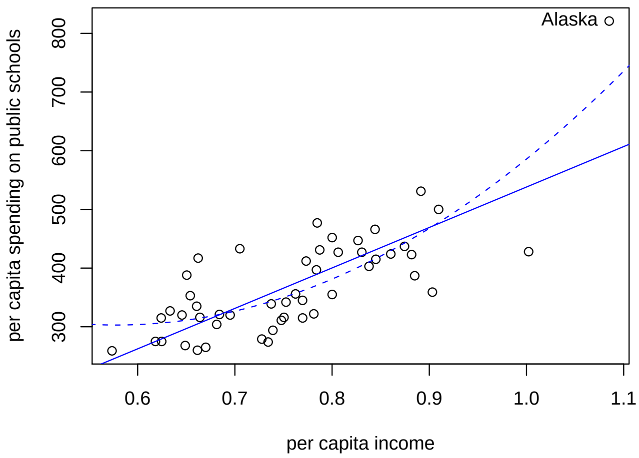

Figure 2: Expenditure on public schools and income with fitted models

```
R> coeftest(fm.inv, df = Inf, vcov = NeweyWest(fm.inv, lag = 4))

z test of coefficients of "lm" object 'fm.inv':

              Estimate Std. Error z value Pr(>|z|)
(Intercept) -12.533601  18.958298 -0.6611   0.5085
RealGNP       0.169136   0.016751 10.0972   <2e-16 ***
RealInt      -1.001438   3.342375 -0.2996   0.7645
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If we would want to use the same estimator, but with automatic bandwidth selection, we could replace the `NeweyWest()` call with a call to `kernHAC` with the appropriate arguments. Alternatively, we could set up a new function that does this job, which is particularly helpful if the same estimator is to be applied several times. First, we set up the function `nwHAC` which is a Bartlett kernel-based HAC estimator without prewhitening and finite sample adjustment. Second, we pass this function (as opposed to a matrix) to `coeftest`.

```
R> nwHAC <- function(x, ...) kernHAC(x, kernel = "Bartlett", prewhite = FALSE,
+     adjust = FALSE)
R> coeftest(fm.inv, df = Inf, vcov = nwHAC)

z test of coefficients of "lm" object 'fm.inv':

              Estimate Std. Error z value Pr(>|z|)
(Intercept) -12.533601  21.234126 -0.5903   0.5550
RealGNP       0.169136   0.020064  8.4300   <2e-16 ***
RealInt      -1.001438   3.611035 -0.2773   0.7815
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This leads to slightly different standard errors because the automatic bandwidths selection chooses $B = 2.28$ (instead of 5), but both tests agree that real GNP has a highly significant influence while the real interest rate has not. The data along with the fitted regression plane are depicted in Figure 3.

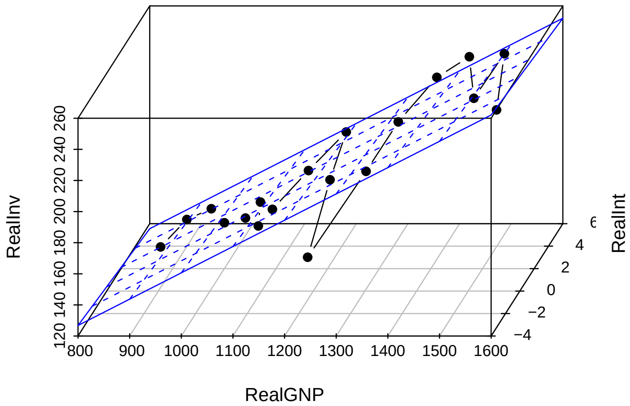

Figure 3: Investment equation data with fitted model

4.3. Testing and dating structural changes in the presence of heteroskedasticity and autocorrelation

To illustrate, that the functionality provided by the covariance estimators implemented in **sandwich** cannot only be used in simple settings, such as partial quasi-t tests but also more complicated settings, we employ the real interest time series analyzed by Bai and Perron (2003). This series contains changes in the mean (see Figure 4, right panel) which Bai and Perron (2003) detect using several structural change tests based on F statistics and date using a dynamic programming algorithm. Here, we use the same dating procedure but assess the significance using an OLS-based CUSUM test (Ploberger and Krämer 1992) with a robust variance estimator. The data are available in the package **strucchange** as the quarterly time series `RealInt` containing the US ex-post real interest rate from 1961(1) to 1986(3) and they are analyzed by a simple regression on the mean.

Under the assumptions in the classical linear model with spherical errors, the test statistic of the OLS-based CUSUM test is

$$\sup_{j=1,\dots,n} \left| \frac{1}{\sqrt{n\,\hat{\sigma}^2}} \sum_{i=1}^{j} \hat{u}_i \right|. \qquad (9)$$

If autocorrelation and heteroskedasticity are present in the data, a robust variance estimator should be used: if x_i is equal to unity, this can simply be achieved by replacing $\hat{\sigma}^2$ with $\hat{\Phi}$ or $n\hat{\Psi}$ respectively. Here, we use the quadratic spectral kernel HAC estimator of Andrews (1991) with VAR(1) prewhitening and automatic bandwidth selection based on an AR(1) approximation as implemented in the function `kernHAC`. The p values for the OLS-based CUSUM test can be computed from the distribution of the supremum of a Brownian bridge (see e.g., Ploberger and Krämer 1992). This and other methods for testing, dating and monitoring structural changes are implemented in the R package **strucchange** (Zeileis, Leisch, Hornik, and Kleiber 2002) which contains the function `gefp()` for fitting and assessing fluctuation processes including OLS-based CUSUM processes (see Zeileis 2004, for more details).

After loading the package and the data,

```
R> library(strucchange)
R> data(RealInt)
```

the command

```
R> ocus <- gefp(RealInt ~ 1, fit = lm, vcov = kernHAC)
```

fits the OLS-based CUSUM process for a regression on the mean (`RealInt ~ 1`), fitted by the function `lm` and estimating the variance using the function `kernHAC`. The fitted OLS-based CUSUM process can then be visualized together with its 5% critical value (horizontal lines) by `plot(scus)` which leads to a similar as in the left panel of Figure 4 (see the appendix for more details). As the process crosses its boundary, there is a significant change in the mean, while the clear peak in the process conveys that there is at least one strong break in the early 1980s. A formal significance test can also be carried out by `sctest(ocus)` which leads to a highly significant p value of 0.0082. Similarly, the same quadratic spectral kernel HAC estimator could also be used for computing and visualizing the sup$F$ test of Andrews (1993), the code is provided in the appendix.

Finally, the breakpoints in this model along with their confidence intervals can be computed by:

```
R> bp <- breakpoints(RealInt ~ 1)
R> confint(bp, vcov = kernHAC)

         Confidence intervals for breakpoints
         of optimal 3-segment partition:

Call:
confint.breakpointsfull(object = bp, vcov = kernHAC)

Breakpoints at observation number:
  2.5 % breakpoints 97.5 %
1    37          47     48
2    77          79     81

Corresponding to breakdates:
  2.5 %   breakpoints 97.5 %
1 1970(1) 1972(3)     1972(4)
2 1980(1) 1980(3)     1981(1)
```

The dating algorithm `breakpoints` implements the procedure described in Bai and Perron (2003) and estimates the timing of the structural changes by OLS. Therefore, in this step no covariance matrix estimate is required, but for computing the confidence intervals using a consistent covariance matrix estimator is again essential. The `confint` method for computing confidence intervals takes again a `vcov` argument which has to be a function (and not a matrix) because it has to be applied to several segments of the data. By default, it computes the breakpoints for the minimum BIC partition which gives in this case two breaks. The fitted three-segment model along with the breakpoints and their confidence intervals is depicted in the right panel of Figure 4.

5. Summary

Acknowledgements

We are grateful to Thomas Lumley for putting his code in the **weave** package at disposal and helping in the design of **sandwich** and to Christian Kleiber for helpful suggestions in the development of **sandwich**.

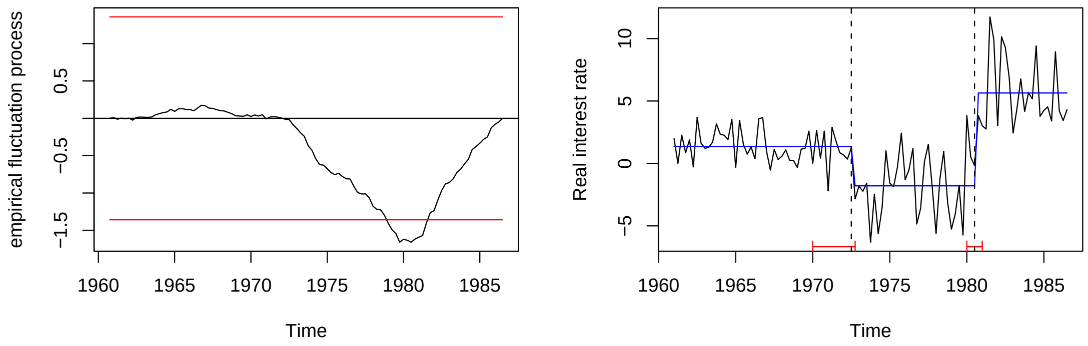

Figure 4: OLS-based CUSUM test (left) and fitted model (right) for real interest data

References

Andrews DWK (1991). "Heteroskedasticity and Autocorrelation Consistent Covariance Matrix Estimation." *Econometrica*, **59**, 817–858.

Andrews DWK (1993). "Tests for Parameter Instability and Structural Change With Unknown Change Point." *Econometrica*, **61**, 821–856.

Andrews DWK, Monahan JC (1992). "An Improved Heteroskedasticity and Autocorrelation Consistent Covariance Matrix Estimator." *Econometrica*, **60**(4), 953–966.

Bai J, Perron P (2003). "Computation and Analysis of Multiple Structural Change Models." *Journal of Applied Econometrics*, **18**, 1–22.

Cribari-Neto F (2004). "Asymptotic Inference Under Heteroskedasticity of Unknown Form." *Computational Statistics & Data Analysis*, **45**, 215–233.

Cribari-Neto F, Zarkos SG (2003). "Econometric and Statistical Computing Using Ox." *Computational Economics*, **21**, 277–295.

Fox J (2002). *An R and S-PLUS Companion to Applied Regression*. Sage Publications, Thousand Oaks, CA.

Greene WH (1993). *Econometric Analysis*. Macmillan Publishing Company, New York, 2nd edition.

Long JS, Ervin LH (2000). "Using Heteroscedasticity Consistent Standard Errors in the Linear Regression Model." *The American Statistician*, **54**, 217–224.

Lumley T, Heagerty P (1999). "Weighted Empirical Adaptive Variance Estimators for Correlated Data Regression." *Journal of the Royal Statistical Society B*, **61**, 459–477.

MacKinnon JG, White H (1985). "Some Heteroskedasticity-consistent Covariance Matrix Estimators with Improved Finite Sample Properties." *Journal of Econometrics*, **29**, 305–325.

Newey WK, West KD (1987). "A Simple, Positive-Definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix." *Econometrica*, **55**, 703–708.

Ploberger W, Krämer W (1992). "The CUSUM Test With OLS Residuals." *Econometrica*, **60**, 271–285.

R Development Core Team (2004). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-00-3, URL http://www.R-project.org/.

White H (1980). "A Heteroskedasticity-consistent Covariance Matrix and a Direct Test for Heteroskedasticity." *Econometrica*, **48**, 817–838.

White H, Domowitz I (1984). "Nonlinear Regression with Dependent Observations." *Econometrica*, **52**, 143–161.

Zeileis A (2004). "Implementing a Class of Structural Change Tests: An Econometric Computing Approach." *Report 7*, Department of Statistics and Mathematics, Wirtschaftsuniversität Wien, Research Report Series. URL http://epub.wu-wien.ac.at/.

Zeileis A, Hothorn T (2002). "Diagnostic Checking in Regression Relationships." *R News*, **2**(3), 7–10. URL http://CRAN.R-project.org/doc/Rnews/.

Zeileis A, Leisch F, Hornik K, Kleiber C (2002). "**strucchange**: An R Package for Testing for Structural Change in Linear Regression Models." *Journal of Statistical Software*, **7**(2), 1–38. URL http://www.jstatsoft.org/v07/i02/.

A. R code

The packages **sandwich**, **lmtest** and **strucchange** are required for the applications in this paper. Furthermore, the packages depend on **zoo**. For the computations in this paper R 1.9.1 and **sandwich** 0.9–0, **lmtest** 0.9–8, **strucchange** 1.2–5 and **zoo** 0.9–0 have been used. R itself and all packages used are available from CRAN at http://CRAN.R-project.org/.

To make the packages available for the examples the following commands are necessary:

```
library(sandwich)
library(lmtest)
library(strucchange)
```

A.1. Testing coefficients in cross-sectional data

Load public schools data, omit `NA` in Wisconsin and scale income:

```
data(PublicSchools)
ps <- na.omit(PublicSchools)
ps$Income <- ps$Income * 1e-04
```

Fit quadratic regression model:

```
fm.ps <- lm(Expenditure ~ Income + I(Income^2), data = ps)
```

Compare standard errors:

```
sqrt(diag(vcov(fm.ps)))
sqrt(diag(vcovHC(fm.ps, type = "const")))
sqrt(diag(vcovHC(fm.ps, type = "HC0")))
sqrt(diag(vcovHC(fm.ps, type = "HC3")))
sqrt(diag(vcovHC(fm.ps, type = "HC4")))
```

Test coefficient of quadratic term:

```
coeftest(fm.ps, df = Inf, vcov = vcovHC(fm.ps, type = "HC0"))
coeftest(fm.ps, df = Inf, vcov = vcovHC(fm.ps, type = "HC4"))
```

Visualization:

```
  plot(Expenditure ~ Income, data = ps, xlab = "per capita income",
+     ylab = "per capita spending on public schools")
  inc <- seq(0.5, 1.2, by = 0.001)
  lines(inc, predict(fm.ps, data.frame(Income = inc)), col = 4,
+     lty = 2)
  fm.ps2 <- lm(Expenditure ~ Income, data = ps)
  abline(fm.ps2, col = 4)
  text(ps[2, 2], ps[2, 1], rownames(ps)[2], pos = 2)
```

A.2. Testing coefficients in time-series data

Load investment equation data:

```
data(Investment)
```

Fit regression model:

```
fm.inv <- lm(RealInv ~ RealGNP + RealInt, data = Investment)
```

Test coefficients using Newey-West HAC estimator with user-defined and data-driven bandwidth:

```
  coeftest(fm.inv, df = Inf, vcov = NeweyWest(fm.inv, lag = 4))
  nwHAC <- function(x, ...) kernHAC(x, kernel = "Bartlett", prewhite = FALSE,
+     adjust = FALSE)
  coeftest(fm.inv, df = Inf, vcov = nwHAC)
```

Time-series visualization:

```
plot(Investment[, "RealInv"], type = "b", pch = 19, ylab = "Real investment")
lines(ts(fitted(fm.inv), start = 1964), col = 4)
```

3-dimensional visualization:

```
  library(scatterplot3d)
  s3d <- scatterplot3d(Investment[, c(5, 7, 6)], type = "b", angle = 65,
+     scale.y = 1, pch = 16)
  s3d$plane3d(fm.inv, lty.box = "solid", col = 4)
```

A.3. Testing and dating structural changes in the presence of heteroskedasticity and autocorrelation

Load real interest series:

```
data(RealInt)
```

OLS-based CUSUM test with quadratic spectral kernel HAC estimate:

```
ocus <- gefp(RealInt ~ 1, fit = lm, vcov = kernHAC)
plot(ocus, aggregate = FALSE)
sctest(ocus)
```

sup$F$ test with quadratic spectral kernel HAC estimate:

```
fs <- Fstats(RealInt ~ 1, vcov = kernHAC)
plot(fs)
sctest(fs)
```

Breakpoint estimation and confidence intervals with quadratic spectral kernel HAC estimate:

```
bp <- breakpoints(RealInt ~ 1)
confint(bp, vcov = kernHAC)
plot(bp)
```

Visualization:

```
par(mfrow = c(1, 2))
plot(ocus, aggregate = FALSE, main = "")
plot(RealInt, ylab = "Real interest rate")
lines(ts(fitted(bp), start = start(RealInt), freq = 4), col = 4)
lines(confint(bp, vcov = kernHAC))
```

A.4. Integrating covariance matrix estimators in other functions

If programmers want to allow for the same flexibility regarding the specification of covariance matrices in their own functions as illustrated in `coeftest`, only a few simple additions have to be made which are illustrated in the following. Say, a function `foo(lmobj, vcov = NULL, ...)` wants to compute some quantity involving the standard errors associated with the `"lm"` object `lmobj`. Then, `vcov` should use by default the standard `vcov` method for `"lm"` objects, otherwise `vcov` is assumed to be either a function returning the covariance matrix estimate or the estimate itself. The following piece of code is sufficient for computing the standard errors.

```
if(is.null(vcov)) {
  se <- vcov(lmobj)
} else {
  if (is.function(vcov))
    se <- vcov(lmobj)
  else
    se <- vcov
}
se <- sqrt(diag(se))
```

In the first step the default method is called: note, that R can automatically distinguish between the variable `vcov` (which is `NULL`) and the generic function `vcov` (from the **stats** package which dispatches to the `"lm"` method) that is called here. Otherwise, it is just distinguished between a function or non-function. In the final step the square root of the diagonal elements is computed and stored in the vector `se` which can subsequently used for further computation in `foo()`.