SCRIPT `MOD4S3A`: SERIAL CORRELATION, CONSUMPTION APPLICATION

INSTRUCTOR: KLAUS MOELTNER

Load and Prepare Data

This example uses Greene's consumption data from `mod4s1b`. Here we focus on money demand as the dependent variable. This model is estimated in Greene, p. 921, (7^{th} edition).

```
R> load('c:/Klaus/AAEC5126/R/data/consumption.rda')
R> attach(data)
R> n<-nrow(data)
```

Variable definitions:

```
% Contents of data
%%%%%%%%%%%%%%%%%%%%%%
% 1     year = Date
% 2     quarter
% 3     realgdp = Real GDP ($billion)
% 4     realcons = Real consumption expenditures ($billion)
% 5     realinvs = Real investment by private sector ($billion)
% 6     realgovt = Real government expenditures ($billion)
% 7     realdpi = Real disposable personal income ($billion)
% 8     cpi = Consumer price index
% 9     m1 = Nominal money stock
% 10   tbill = Quarterly average of month end 90 day t bill rate
% 11   unemp = Unemployment rate
% 12   pop = Population, million
% 13   infl = Rate of inflation (first observation is missing and set to zero)
% 14   realint = Ex post real interest rate = Tbilrate - Infl.
%       (First observation missing and set to zero)
```

Simple OLS

```
R> # Define variables
R> n<-nrow(data)
R> y<-log(m1)
R> X<-cbind(rep(1,n),log(realgdp),log(cpi))
R> k<-ncol(X)
R> #
R> bols<-solve((t(X)) %*% X) %*% (t(X) %*% y)# compute OLS estimator
R> e<-y-X%*%bols # Get residuals.
R> SSR<-(t(e)%*%e)#sum of squared residuals - should be minimized
R> s2<-(t(e)%*%e)/(n-k) #get the regression error (estimated variance of "eps").
R> s2ols<-s2 #for Hausman test below
R> Vb<-s2[1,1]*solve((t(X))%*%X) # get the estimated VCOV matrix of bols
R> se=sqrt(diag(Vb)) # get the standard erros for your coefficients;
R> tval=bols/se # get your t-values.
```

```
R> #
R> tt<-data.frame(col1=c("constant","log(gpd)","log(cpi)"),
                 col2=bols,
                 col3=se,
                 col4=tval)
R> colnames(tt)<-c("variable","estimate","s.e.","t")
```

TABLE 1. OLS output

variable	estimate	s.e.	t
constant	-1.633	0.229	-7.145
log(gpd)	0.287	0.047	6.058
log(cpi)	0.972	0.034	28.775

RESIDUAL PLOT

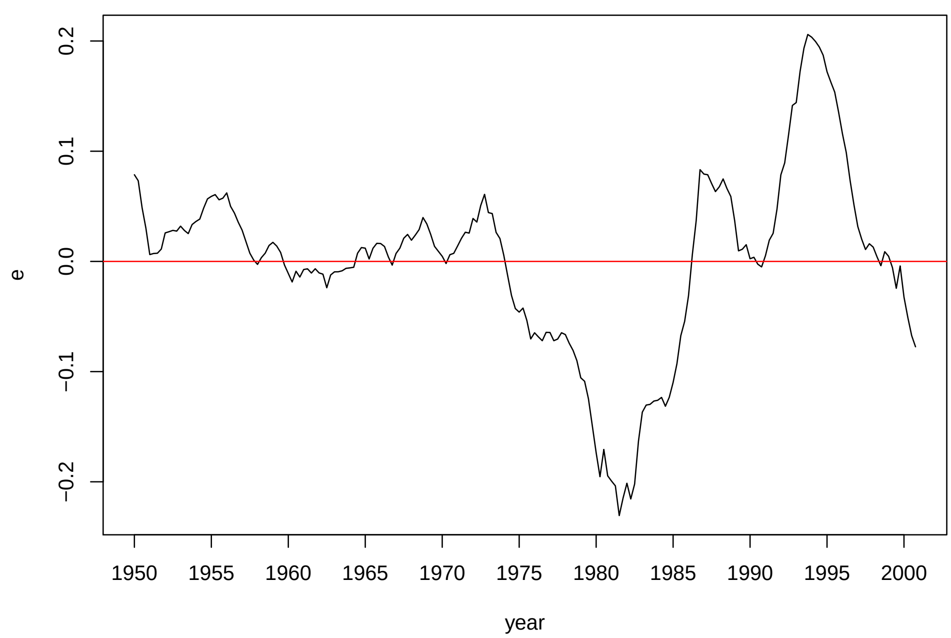

FIGURE 1. OLS residual plots

ROBUST OLS

We'll use the Newey-West (1987) procedure as shown in the lecture notes. The tricky part is composing the S_1 matrix. Setting $L = 5$ produces Greene's results. A more common alternative would be to choose $L = n^{1/4} \approx 4$ (rounded to the nearest integer).

```
R> #L<-ceiling(n^(1/4)); #rounds upwards to nearest integer;
R> # this would be the generic choice
R> L<-5 #this produces Greene's results
R> H<-matrix(0,k,k)
R> for (j in 1:L) {
     t<-j+1
     G<-matrix(0,k,k)
     for (i in t:n)  {
         m<-(1-(j/(L+1)))*e[i]*e[i-j]*
         (t(X[i,,drop=FALSE])%*% X[i-j,]+t(X[i-j,,drop=FALSE]) %*% X[i,])
         #drop=FALSE forces the transpose to be a column vector
         G<-G+m
     }
     H<-H+G
 }
R> e<-as.vector(e)
R> S1<-(t(X) %*% diag(e^2) %*% X)+H
R> Vb<-solve((t(X))%*%X) %*% S1 %*% solve((t(X))%*%X)
R> se=sqrt(diag(Vb))
R> tval=bols/se
R> tt<-data.frame(col1=c("constant","log(gpd)","log(cpi)"),
                col2=bols,
                col3=se,
                col4=tval)
R> colnames(tt)<-c("variable","estimate","s.e.","t")
```

Table 2. Robust OLS output

variable	estimate	s.e.	t
constant	-1.633	0.335	-4.868
log(gpd)	0.287	0.078	3.677
log(cpi)	0.972	0.066	14.758

Testing for AR(1) Serial Correlation

We first plot, then regress the OLS residuals against their lag-1 neighbors.

```
R> n<-length(ecurr)  #can't use nrow() for a vector
R> y<-ecurr
R> X<-elag
R> k<-1
R> #
R> bols<-solve((t(X)) %*% X) %*% (t(X) %*% y)# compute OLS estimator
R> e<-y-X%*%bols # Get residuals.
R> SSR<-(t(e)%*%e)#sum of squared residuals - should be minimized
R> s2<-(t(e)%*%e)/(n-k) #get the regression error (estimated variance of "eps").
R> s2ols<-s2 #for Hausman test below
R> Vb<-s2[1,1]*solve((t(X))%*%X) # get the estimated VCOV matrix of bols
```

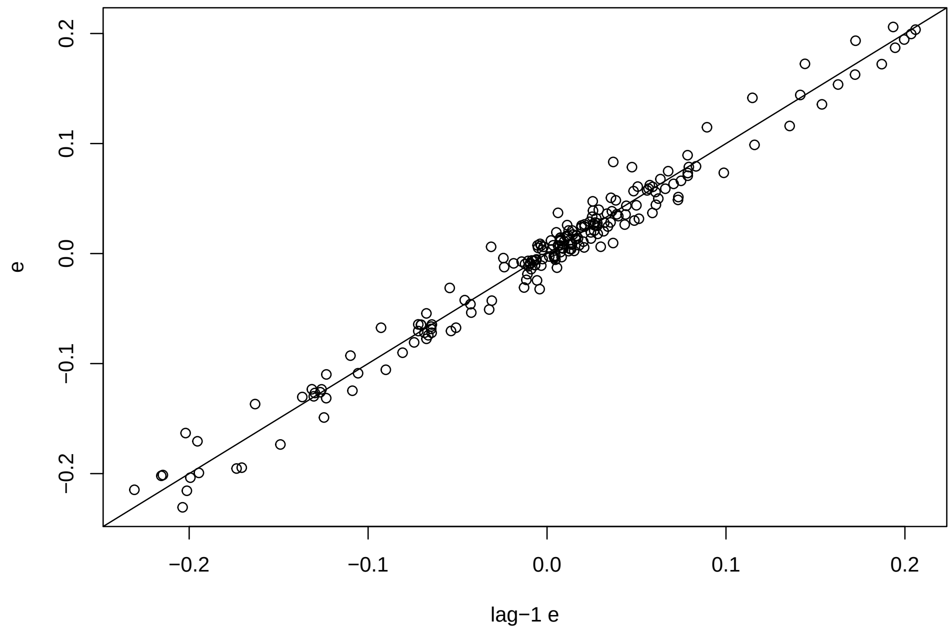

Figure 2. residuals vs. lag-1 residuals

```
R> se=sqrt(diag(Vb)) # get the standard erros for your coefficients;
R> tval=bols/se # get your t-values.
R> #
R> tt<-data.frame(col1=c("lag-1 e"),
                 col2=bols,
                 col3=se,
                 col4=tval)
R> colnames(tt)<-c("variable","estimate","s.e.","t")
```

Table 3. Residual vs. lagged residual plot

variable	estimate	s.e.	t
lag-1 e	0.987	0.011	89.269

1. Breusch-Godfrey Multipier Test for AR(1)

```
R> #re-run original OLS and capture residuals
R> n<-nrow(data) #re-define original n
R> y<-log(m1)
R> X<-cbind(rep(1,n),log(realgdp),log(cpi)) #re-build orig. X
R> k<-ncol(X)
R> bols<-solve((t(X)) %*% X) %*% (t(X) %*% y)# compute OLS estimator
```

```
R> e<-y-X%*%bols # Get residuals.
R> elag<-e[1:(n-1)]
R> #
R> e0lag<-c(0,elag) # fill first position with 0 */
R> Xo=cbind(X, e0lag) #augment X with a column of lagged residuals
R> #
R> LM<-n*((t(e) %*% Xo %*% solve(t(Xo) %*% Xo) %*% t(Xo) %*% e)/(t(e) %*% e))
R> pval=1-pchisq(LM,1)
```

The BG-statistic for this test is 198.225. The degrees of freedom for the test are 1. The corresponding p-value is 0.

2. Durbin-Watson Test

```
R> ecurr<-e[2:n]
R> elag<-e[1:(n-1)]
R> d<-(t(ecurr-elag) %*% (ecurr-elag))/(t(e) %*% e)
```

The DW-statistic for this test is 0.025. The sample size is 204. The column space of X is 3.

3. Prais-Winsten FGLS

```
R> # Step 1: Get a consistent estimate of rho:
R> rho<-solve(t(elag) %*% elag) %*% t(elag) %*% ecurr #OLS solution for our
R> # "e vs. e-lag 1 regression model above
R> #
R> #Step 2: compose the correlation matrix R
R> R<-matrix(0,n,n)
R> up<-seq(1,(n-1),1)
R> down<-seq((n-1),1,-1)
R> int<- c(rho^(down), 1, rho^(up)) #1 by 2*(n-1)+1
R> for (i in 1:n){
 R[i,]<-int[(n-(i-1)):(length(int)-(i-1))]
 }
R> #
R> #Step 3: compute FGLS estimator
R> bgls<-solve((t(X)) %*% solve(R) %*% X) %*% (t(X) %*% solve(R) %*% y)
R> #
R> #Step 4: compute a consistent estimate of sig(eps)
R> e<-y-X%*%bgls
R> sige<-(1/n)*t(e) %*% solve(R) %*% (e)
R> #
R> #Step 5: Compute consistent variance-covariance matrix for b_fgls
R> Om<-sige[1,1]*R
R> Vb<-solve((t(X))%*% solve(Om) %*% X)
R> se=sqrt(diag(Vb))
R> tval=bgls/se
R> #
R> ttgls<-data.frame(col1=c("constant","log(gpd)","log(cpi)"),
                col2=bgls,
                col3=se,
```

```
                col4=tval)
R> colnames(ttgls)<-c("variable","estimate","s.e.","t")
```

Table 4. FGLS output

variable	estimate	s.e.	t
constant	-1.538	0.416	-3.700
log(gpd)	0.471	0.068	6.947
log(cpi)	0.662	0.062	10.676

```
R> proc.time()-tic
   user  system elapsed
   0.33    0.11    0.44
```