# Estimating supply and demand elasticities in STATA

Fabio Gaetano Santeramo

fabiogaetano.santeramo@gmail.com

**Summary**

The present hand-out provides a guide for estimating supply elasticities described in the literature review using STATA. In order to reduce incompatibilities of our code with old versions of STATA, we present codes that can be run in STATA 10 (or in newer versions). We describe codes to estimate different functional forms. The first section describes the estimation of elasticities in producer theory; the second section is dedicated to consumer theory. This note is in addition to AGRODEP Technical Note 10.

**Acknowledgments**

The author is grateful to Antoine Bouët, David Laborde, Simla Tokgoz, and Lauren Deason for their valuable comments and feedback on the present note and on the AGRODEP Technical Note No. 10. All remaining errors are mine.

## Section 1 – Producer theory

This example considers data for input use, output, quantities, costs, and prices for total U.S. nondurable manufacturing for 1949-2001. We use the file Dataset1.csv.

The analysis presented here is similar to this classic paper: Berndt and Wood, "Technology, Prices, and the Derived Demand for Energy", The Review of Economics and Statistics, Vol. 57, No. 3. (Aug., 1975), pp. 259-268.

*Preliminary commands and data import*

```
clear all
set mem 10000K
insheet using "<write your path here>\Dataset1.csv"
```

*Description of the variables in the dataset*

The prefix *y* indicates total output; the prefixes *k, l, e, m,* and *s* indicate inputs corresponding to capital, labor, energy, materials, and purchased services. The subscript *q* indicates respective quantity indexes; the subscript *v* indicates respective values/costs (nominal terms in $billions); the subscript *p* indicates respective price indexes; the subscript *s* indicates respective cost shares.

Year= Date

y_q = Output quantity
k_q = Capital quantity
l_q = Labor quantity
e_q = Energy quantity
m_q = Materials quantity
s_q = Services quantity

y_p = Output price
k_p = Capital price
l_p = Labor price
e_p = Energy price
m_p = Materials price
s_p = Services price

y_v = Output cost
k_v = Capital cost
l_v = Labor cost
e_v = Energy cost
m_v = Materials cost
s_v = Services cost

*Generate log variables*

```
gen q    = log(y_q)
gen pk  = log(k_p)
gen pl   = log(l_p)
gen pe  = log(e_p)
gen pm = log(m_p)
gen ps = log(s_p)
```

*Generate sum of costs*

```
gen c  =  k_v + l_v  + e_v + m_v + s_v
```

*Generate cost shares*

```
gen k_s = k_v/c
gen l_s = l_v/c
gen e_s = e_v/c
gen m_s = m_v/c
gen s_s = s_v/c

gen lc = log(c)
```

- *First we show how to estimate the double-log factor demands*

*In this simple case the coefficients are directly interpretable as elasticities*

```
regr  k_s  pk pl pe pm ps q
regr  l_s   pk pl pe pm ps q
regr  e_s  pk pl pe pm ps q
regr m_s pk pl pe pm ps q
```

- *Second, we describe how to estimate the translog factor demands without a cost function, also imposing homogeneity and symmetry.*

*First, consider how to estimate a seemingly unrelated regression model.*

*SUR estimation - UNCONSTRAINED*

```
sureg (k_s pk pl pe pm ps q) (l_s pk pl pe pm ps q) (e_s pk pl pe pm ps q) (m_s pk pl pe pm ps q), corr
```

*Second, we test for symmetry and homogeneity*[1]

*Tests for HOMOGENEITY*

```
test ([k_s]ps = 0 - ([k_s]pk + [k_s]pl + [k_s]pe + [k_s]pm)) ([l_s]ps = 0 - ([l_s]pk + [l_s]pl + [l_s]pe + [l_s]pm))
([e_s]ps = 0 - ([e_s]pk + [e_s]pl + [e_s]pe + [e_s]pm)) ([m_s]ps = 0 - ([m_s]pk + [m_s]pl + [m_s]pe + [m_s]pm))
```

*Tests for SYMMETRY*

[1]Homogeneity derives from the satisfaction of the maximization problem (see equation 1 *of Agrodep Technical note N. 10*). In other words, the solution of the maximization problem implies that the sum of the marginal value of the factor productivities is zero. Symmetry is implied by the concavity of the production function (please also refer to page 4, line 5, of the same technical note).

test ([k_s]pl-[l_s]pk=0) ([k_s]pe-[e_s]pk=0) ([k_s]pm-[m_s]pk=0) ([l_s]pm-[m_s]pl=0) ([e_s]pm-[m_s]pe=0) ([l_s]pe-[e_s]pl=0)

We may also jointly test for HOMOGENEITY AND SYMMETRY using Likelihood ratio test (LR TEST). To this extent we need to estimate the unconstrained and the constrained model, save the results, and then conduct a LR test.

*SUR estimation - UNCONSTRAINED*
sureg (k_s pk pl pe pm ps q) (l_s pk pl pe pm ps q) (e_s pk pl pe pm ps q) (m_s pk pl pe pm ps q), corr

*Save results*
est store unrestr

*SUR estimation – CONSTRAINED*

*Define constraints*

constraint 1 [k_s]ps = 0 - ([k_s]pk + [k_s]pl + [k_s]pe + [k_s]pm)
constraint 2 [l_s]ps = 0 - ([l_s]pk + [l_s]pl + [l_s]pe + [l_s]pm)
constraint 3 [e_s]ps = 0 - ([e_s]pk + [e_s]pl + [e_s]pe + [e_s]pm)
constraint 4 [m_s]ps = 0 - ([m_s]pk + [m_s]pl + [m_s]pe + [m_s]pm)
constraint 5 [k_s]pl-[l_s]pk=0
constraint 6 [k_s]pe-[e_s]pk=0
constraint 7 [k_s]pm-[m_s]pk=0
constraint 8 [l_s]pm-[m_s]pl=0
constraint 9 [e_s]pm-[m_s]pe=0
constraint 10 [l_s]pe-[e_s]pl=0

*Estimate the model with constraints*

sureg (k_s pk pl pe pm ps q) (l_s pk pl pe pm ps q) (e_s pk pl pe pm ps q) (m_s pk pl pe pm ps q), constraints(1 2 3 4 5 6 7 8 9 10) corr

*Save results*

est store restr

*Test for homogeneity and symmetry using the LR test*

lrtest restr unrestr

- *Third, we show how to estimate the translog factor demands including the cost function, and imposing homogeneity and symmetry.*

*Generate squared variables*

g q2 = q*q
g pk2 = pk*pk
g pl2 = pl*pl
g pe2 = pe*pe
g pm2 = pm*pm
g ps2 = ps*ps

*Generate cross-products variables*

```
g pspk  = ps*pk
g pspl   = ps*pl
g pspe  = ps*pe
g pspm= ps*pm

g pkps   = pk*ps
g pkpl   = pk*pl
g pkpe  = pk*pe
g pkpm = pk*pm
g plps    = pl*ps
g plpk   = pl*pk
g plpe   = pl*pe
g plpm  = pl*pm

g peps  = pe*ps
g pepl   = pe*pl
g pepk  = pe*pk
g pepm = pe*pm

g pmps = pm*ps
g pmpl  = pm*pl
g pmpe = pm*pe
g pmpk = pm*pk

g qps   = q*ps
g qpl   = q*pl
g qpe  = q*pe
g qpk  = q*pk
g qpm = q*pm
```

*Define additional constraints on cross-products*

```
constraint 11 ps2 = 0-(pkps + pspl + peps + pspm)
constraint 12 qps= 0-(qpk + qpl + qpe + qpm)
constraint 13 ps  = 1-(pk + pl + pe + pm)
```

*Estimate the system of equations imposing constraints for homogeneity and symmetry*

```
sureg (lc q pk pl pe pm ps q2 pk2 pl2 pe2 pm2 ps2 pspl pspm pkps pkpl pkpe pkpm plpm  peps pepl pepm qps qpl qpe qpk qpm) (k_s pk pl pe pm ps q) (l_s pk pl pe pm ps q) (e_s pk pl pe pm ps q) (m_s pk pl pe pm ps q),
constraints(1 2 3 4 11 12 13 ) corr
```

- *Finally, we show how to compute elasticities of interest*

*Define the matrix of variables' averages*

```
qui sum k_s
scalar mu_k_s =  r(mean)

qui sum l_s
```

```
scalar mu_l_s =  r(mean)

qui sum e_s
scalar mu_e_s =  r(mean)

qui sum m_s
scalar mu_m_s =  r(mean)

qui sum s_s
scalar mu_s_s =  r(mean)

matrix Mean = J(5,1,.)
matrix Mean[1,1] = mu_k_s
matrix Mean[2,1] = mu_l_s
matrix Mean[3,1] = mu_e_s
matrix Mean[4,1] = mu_m_s
matrix Mean[5,1] = mu_s_s
matrix list Mean

*Estimate and save the results of the system of equations*
sureg (lc q pk pl pe pm ps q2 pk2 pl2 pe2 pm2 ps2 pspl pspm pkps pkpl pkpe pkpm plpm  peps pepl pepm qps qpl qpe qpk qpm) (k_s pk pl pe pm ps q) (l_s pk pl pe pm ps q) (e_s pk pl pe pm ps q) (m_s pk pl pe pm ps q),
constraints (1 2 3 4 11 12 13 ) corr
est store estimate

*Save ALL coefficients in a matrix*
matrix bvec = e(b)
matrix list bvec

*Compute scalars*
scalar gks = 0 - (_b[k_s:pk] + _b[k_s:pl] + _b[k_s:pe] + _b[k_s:pm])
scalar gls = 0 - (_b[k_s:pl] + _b[l_s:pl] + _b[l_s:pe] + _b[l_s:pm])
scalar ges = 0 - (_b[k_s:pe] + _b[l_s:pe] + _b[e_s:pe] + _b[e_s:pm])
scalar gms = 0 - (_b[k_s:pm] + _b[l_s:pm] + _b[e_s:pm] + _b[m_s:pm])
scalar gss = 0 - (_b[k_s:ps] + _b[l_s:ps] + _b[e_s:ps] + _b[m_s:ps])
```

*Fac-simile of output in STATA*

```
Seemingly unrelated regression

Constraints:
 ( 1)  [s_k]pk + [s_k]pl + [s_k]pe + [s_k]pm + [s_k]ps = 0
 ( 2)  [s_l]pk + [s_l]pl + [s_l]pe + [s_l]pm + [s_l]ps = 0
 ( 3)  [s_e]pk + [s_e]pl + [s_e]pe + [s_e]pm + [s_e]ps = 0
 ( 4)  [s_m]pk + [s_m]pl + [s_m]pe + [s_m]pm + [s_m]ps = 0
 ( 5)  [lc]ps2 + [lc]pspl + [lc]pspm + [lc]pkps + [lc]peps = 0
 ( 6)  [lc]kyps + [lc]kypl + [lc]kype + [lc]kypk + [lc]kypm = 0
 ( 7)  [lc]pk + [lc]pl + [lc]pe + [lc]pm + [lc]ps = 1
```

| Equation | Obs | Parms | RMSE | "R-sq" | chi2 | P |
|---|---|---|---|---|---|---|
| lc | 53 | 23 | .0176853 | 0.9998 | 83033.67 | 0.0000 |
| s_k | 53 | 5 | .0076638 | 0.9446 | 904.00 | 0.0000 |
| s_l | 53 | 5 | .0111879 | 0.7137 | 132.15 | 0.0000 |
| s_e | 53 | 5 | .0062256 | 0.8211 | 243.29 | 0.0000 |
| s_m | 53 | 5 | .0115179 | 0.8632 | 334.56 | 0.0000 |

| | Coef. | Std. Err. | z | P>\|z\| | [95% Conf. | Interval] |
|---|---|---|---|---|---|---|
| **lc** | | | | | | |
| q | 2.765617 | 2.008916 | 1.38 | 0.169 | -1.171787 | 6.70302 |
| pk | -5.379265 | 2.40729 | -2.23 | 0.025 | -10.09747 | -.6610628 |
| pl | 4.050648 | 3.39529 | 1.19 | 0.233 | -2.603998 | 10.70529 |
| pe | -7.825782 | 2.382777 | -3.28 | 0.001 | -12.49594 | -3.155625 |
| pm | 10.1544 | 2.946727 | 3.45 | 0.001 | 4.378921 | 15.92988 |
| ps | (dropped) | | | | | |
| q2 | -.2519045 | .2111798 | -1.19 | 0.233 | -.6658093 | .1620003 |
| pk2 | -.0526136 | .1873094 | -0.28 | 0.779 | -.4197333 | .3145062 |
| pl2 | 1.916256 | .9938491 | 1.93 | 0.054 | -.0316519 | 3.864165 |
| pe2 | -.4709568 | .1601791 | -2.94 | 0.003 | -.7849021 | -.1570114 |
| pm2 | -1.279961 | .5576191 | -2.30 | 0.022 | -2.372875 | -.1870478 |
| ps2 | -.5874477 | .7073773 | -0.83 | 0.406 | -1.973882 | .7989864 |
| pspl | -1.603024 | 1.023476 | -1.57 | 0.117 | -3.609001 | .4029518 |
| pspm | -2.963176 | 1.591245 | -1.86 | 0.063 | -6.081958 | .1556066 |
| pkps | 2.758356 | 1.54755 | 1.78 | 0.075 | -.2747858 | 5.791497 |
| pkpl | -2.089726 | 1.080537 | -1.93 | 0.053 | -4.207539 | .0280874 |
| pkpe | -.2774026 | .2412397 | -1.15 | 0.250 | -.7502236 | .1954185 |
| pkpm | .4966732 | .4173333 | 1.19 | 0.234 | -.3212851 | 1.314631 |
| plpm | 2.152827 | 1.167778 | 1.84 | 0.065 | -.1359753 | 4.441629 |
| peps | 2.395292 | 1.406462 | 1.70 | 0.089 | -.3613222 | 5.151906 |
| pepl | -2.351903 | 1.054381 | -2.23 | 0.026 | -4.418451 | -.2853547 |
| pepm | 1.997128 | .5894774 | 3.39 | 0.001 | .8417735 | 3.152483 |
| kyps | .6899202 | .7048817 | 0.98 | 0.328 | -.6916225 | 2.071463 |
| kypl | -.7552678 | .8030733 | -0.94 | 0.347 | -2.329262 | .8187269 |
| kype | .9062487 | .2591079 | 3.50 | 0.000 | .3984065 | 1.414091 |
| kypk | .4505248 | .2986195 | 1.51 | 0.131 | -.1347587 | 1.035808 |
| kypm | -1.291426 | .5416076 | -2.38 | 0.017 | -2.352957 | -.2298946 |
| _cons | -14.12569 | 4.626841 | -3.05 | 0.002 | -23.19414 | -5.057253 |
| **s_k** | | | | | | |
| pk | .1693965 | .0091916 | 18.43 | 0.000 | .1513814 | .1874116 |
| pl | .0743455 | .0245731 | 3.03 | 0.002 | .026183 | .1225079 |
| pe | .0167598 | .0077753 | 2.16 | 0.031 | .0015206 | .0319991 |
| pm | -.13598 | .0155572 | -8.74 | 0.000 | -.1664716 | -.1054883 |
| ps | -.1245218 | .0313617 | -3.97 | 0.000 | -.1859895 | -.0630541 |
| q | -.0420186 | .0088999 | -4.72 | 0.000 | -.0594621 | -.0245752 |
| _cons | .4677765 | .040679 | 11.50 | 0.000 | .388047 | .5475059 |
| **s_l** | | | | | | |
| pk | -.1187613 | .0134182 | -8.85 | 0.000 | -.1450605 | -.092462 |
| pl | .0410933 | .035873 | 1.15 | 0.252 | -.0292166 | .1114031 |
| pe | -.0768488 | .0113507 | -6.77 | 0.000 | -.0990957 | -.0546018 |
| pm | .0249392 | .0227112 | 1.10 | 0.272 | -.0195739 | .0694523 |
| ps | .1295775 | .0457833 | 2.83 | 0.005 | .0398439 | .2193111 |
| q | -.039849 | .0129924 | -3.07 | 0.002 | -.0653137 | -.0143842 |
| _cons | .4520831 | .059385 | 7.61 | 0.000 | .3356906 | .5684757 |
| **s_e** | | | | | | |
| pk | -.0373586 | .0074667 | -5.00 | 0.000 | -.0519931 | -.0227241 |
| pl | -.0088574 | .0199619 | -0.44 | 0.657 | -.047982 | .0302672 |
| pe | .0077917 | .0063162 | 1.23 | 0.217 | -.0045878 | .0201712 |
| pm | .0669172 | .0126378 | 5.29 | 0.000 | .0421475 | .0916869 |
| ps | -.0284929 | .0254765 | -1.12 | 0.263 | -.078426 | .0214402 |
| q | .0199632 | .0072298 | 2.76 | 0.006 | .0057931 | .0341332 |
| _cons | -.0489827 | .0330454 | -1.48 | 0.138 | -.1137504 | .0157851 |
| **s_m** | | | | | | |
| pk | -.0011581 | .013814 | -0.08 | 0.933 | -.0282329 | .0259168 |
| pl | -.0485293 | .036931 | -1.31 | 0.189 | -.1209127 | .0238541 |
| pe | .075533 | .0116854 | 6.46 | 0.000 | .0526299 | .098436 |
| pm | .0418717 | .023381 | 1.79 | 0.073 | -.0039541 | .0876976 |
| ps | -.0677173 | .0471335 | -1.44 | 0.151 | -.1600974 | .0246627 |
| q | -.0035644 | .0133756 | -0.27 | 0.790 | -.0297802 | .0226513 |
| _cons | .2909419 | .0611364 | 4.76 | 0.000 | .1711167 | .410767 |

*Define the matrix of coefficients*

matrix gij=(_b[k_s:pk], _b[k_s:pl], _b[k_s:pe], _b[k_s:pm], _b[k_s:ps]_b[k_s:pl], _b[l_s:pl], _b[l_s:pe], _b[l_s:pm], _b[l_s:ps]_b[k_s:pe], _b[l_s:pe], _b[e_s:pe], _b[e_s:pm], _b[e_s:ps]_b[k_s:pm], _b[l_s:pm], _b[e_s:pm], _b[m_s:pm], _b[m_s:ps]_b[k_s:ps], _b[l_s:ps], _b[e_s:ps], _b[m_s:ps], gss)

matrix eos = J(5,5,.)
matrix list eos

matrix mos = J(5,5,.)
matrix list mos

```
matrix ep  = J(5,5,.)
matrix list ep

Define an identity matrix
matrix I = -I(5)
matrix lis I

Compute Elasticities of substitution and price elasticites
forvalues i = 1/5  {
forvalues j = 1/5  {
matrix eos[`i',`j'] = (gij[`i',`j']+Mean[`i',1]*Mean[`j',1]+I[`i',`j']*Mean[`i',1])/(Mean[`i',1]*Mean[`j',1])
matrix ep[`i',`j']= (Mean[`i',1]*eos[`i',`j'])
}
}

Compute Morishima elasticities
forvalues i = 1/5  {
forvalues j = 1/5  {
matrix mos[`i',`j'] = (ep[`i',`j']-ep[`j',`i'])
}
}
Show the matrices

mat lis ep
mat lis eos
mat lis mos
```

*Fac-simile of prices' elasticities matrix*

| | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| **1** | *-0.06* | *0.50* | *0.57* | *-0.21* | *-0.83* |
| **2** | *0.59* | *-0.57* | *-1.23* | *0.37* | *1.41* |
| **3** | *0.12* | *-0.22* | *-0.80* | *0.27* | *-0.20* |
| **4** | *-0.25* | *0.39* | *1.62* | *-0.56* | *-0.28* |
| **5** | *-0.39* | *0.57* | *-0.45* | *-0.11* | *-0.10* |

*Elasticities of substitution*

| | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| **1** | *-0.25* | | | | |
| **2** | *2.06* | *-1.98* | | | |
| **3** | *2.35* | *-4.27* | *-15.68* | | |
| **4** | *-0.84* | *1.29* | *5.36* | *-1.85* | |
| **5** | *-3.40* | *4.89* | *-3.85* | *-0.93* | *-0.84* |

*Morishima elasticities*

| | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| **1** | *0.00* | *-0.09* | *0.45* | *0.05* | *-0.44* |
| **2** | *0.09* | *0.00* | *-1.01* | *-0.02* | *0.84* |
| **3** | *-0.45* | *1.01* | *0.00* | *-1.35* | *0.25* |
| **4** | *-0.05* | *0.02* | *1.35* | *0.00* | *-0.17* |
| **5** | *0.44* | *-0.84* | *-0.25* | *0.17* | *0.00* |

## Section 2 – Consumer theory

We present an example for consumer theory that shows how to estimate a system of demand functions by using the absolute version of the Rotterdam Model[2]. We use the file Dataset2.csv.

First, recall that the Rotterdam Model has the following form:

$$\acute{w}_{it}\Delta log q_{it} = a_i + b_i \Delta log \acute{x}_t + \sum_{k=1}^{4} c_{ik}\, \Delta log p_{kt} + u_{it}$$

where $E(u_{it}) = 0,\ E(u_{it}u_{kt}) = \sigma_{ik}\forall i,k$, and $E(u_{it}u_{kt'}) = 0 \forall t \neq t'$. For the relative change in income, we use $\Delta log \acute{x}_t = \sum \acute{w}_{kt}\Delta log q_{kt}$[3]. (Note that the model includes an intercept as in Barten, *R.E. Stat., 1967*).

*Preliminary commands and data import*

clear all
set mem 10000K
insheet using "<write your path here> \Dataset2.csv"

*Description of the variables in the dataset*

| | | |
|---|---|---|
| # beefq | per capita retail quantity of beef | (lbs/person) |
| # vealq | per capita retail quantity of veal | (lbs/person) |
| # porkq | per capita retail quantity of pork | (lbs/person) |
| # poultq | per capita retail quantity of poultry | (lbs/person) |
| # fishq | per capita retail quantity of fish & seafood | (lbs/person) |
| # bfvlq | per capita retail quantity of beef & veal | (lbs/person) |
| # bfvlp | consumer price index beef & veal | (1982-84=100) |
| # porkp | consumer price index pork | (1982-84=100) |
| # poultp | consumer price index poultry | (1982-84=100) |
| # fishp | consumer price index fish & seafood | (1982-84=100) |
| # cbfvlq | constant dollar per capita retail quantity beef & veal | ($/person) |
| # cporkq | constant dollar per capita retail quantity pork | ($/person) |
| # cpoultq | constant dollar per capita retail quantity poultry | ($/person) |
| # cfishq | constant dollar per capita retail quantity fish & seafood | ($/person) |
| # cpi | consumer price index-all items | (1982-84=100) |
| # pop | total civilian population, July 1 | (millions) |
| # pce | personal consumption expenditures | ($billion) |
| # xbfvl | per capita beef & veal expenditures | ($/person) |
| # xpork | per capita pork expenditures | ($/person) |
| # xpoult | per capita poultry expenditures | ($/person) |
| # xfish | per capita fish & seafood expenditures | ($/person) |
| # xtotal | per capita personal consumption expenditures | ($/person) |

[2] An excellent example to estimate QAIDS models is provided in the following article: Poi, B. P. (2002). "From the help desk: Demand system estimation". *The Stata Journal*, Vol. 2, No. 4, pp. 403–410.

[3] The restrictions that must hold in order for this to be a valid demand system are as follows: $\sum_{i=1}^{4} \boldsymbol{a_i} = \boldsymbol{0}; \sum_{i=1}^{4} \boldsymbol{b_i} = \boldsymbol{1}; \sum_{i=1}^{4} \boldsymbol{c_{ik}} = \boldsymbol{0}; \sum_{k=1}^{4} \boldsymbol{c_{ik}} = \boldsymbol{0}\forall \boldsymbol{i}; \boldsymbol{c_{ik}} = \boldsymbol{c_{ki}}\forall \boldsymbol{i} \neq \boldsymbol{k}$. Recall that the system is singular so that one equation will need to be deleted prior to estimation. The testable restrictions are the homogeneity restrictions, $\sum_{k=1}^{4} \boldsymbol{c_{ik}} = \boldsymbol{0}\forall \boldsymbol{i}$, and the symmetry restrictions, $\boldsymbol{c_{ik}} = \boldsymbol{c_{ki}}$. The total number of restrictions to test is (n-1) +(1/2)(n-2)(n-1).

```
# QUANTITIES
gen log_bfvlq= log(bfvlq)
gen log_porkq = log(porkq)
gen log_poultq= log(poultq)
gen log_fishq = log(fishq)

# PRICES
gen log_bfvlp= log(bfvlp)
gen log_porkp = log(porkp)
gen log_poultp= log(poultp)
gen log_fishp = log(fishp)

gen log_xtotal=log(xtotal)

*Define the time series nature of data*
tset year

*Generate deltalog variables*
gen dlog_bfvlq= d.log_bfvlq
gen dlog_porkq = d.log_porkq
gen dlog_poultq= d.log_poultq
gen dlog_fishq = d.log_fishq

gen dlog_bfvlp= d.log_bfvlp
gen dlog_porkp = d.log_porkp
gen dlog_poultp= d.log_poultp
gen dlog_fishp = d.log_fishp

gen dlog_xtotal=d.log_xtotal

*Generate the variable W hat*
gen xbfvl_hat       = 0.5*(xbfvl  +l.xbfvl)
gen xpork_hat     = 0.5*(xpork  +l.xpork)
gen xpoultry_hat = 0.5*(xpoult +l.xpoult)
gen xfish_hat       = 0.5*(xfish  +l.xfish)

*Generate the dependent variables(ẃ)*
gen wdlog_bfvlq = dlog_bfvlq*xbfvl_hat
gen wdlog_porkq  =  dlog_porkq*xpork_hat
gen wdlog_poultq =  dlog_poultq*xpoultry_hat
gen wdlog_fishq    =  dlog_fishq*xfish_hat

*Generate real income variables*
gen real_bfvl    = dlog_xtotal-(dlog_porkp*xpork_hat  + dlog_poultp*xpoultry_hat + dlog_fishp*xfish_hat )
gen real_pork  = dlog_xtotal-(dlog_bfvlp*xbfvl_hat + dlog_poultp*xpoultry_hat + dlog_fishp*xfish_hat )
gen real_poult = dlog_xtotal-(dlog_porkp*xpork_hat + dlog_bfvlp*xbfvl_hat +dlog_fishp*xfish_hat  )
gen real_fish    = dlog_xtotal-(dlog_porkp*xpork_hat + dlog_poultp*xpoultry_hat+ dlog_bfvlp*xbfvl_hat)
```

- *First we present the regressions we need to estimate*

```
regr wdlog_bfvlq   real_bfvl    dlog_bfvlp dlog_porkp dlog_poultp dlog_fishp
```

regr wdlog_porkq  real_pork  dlog_bfvlp dlog_porkp dlog_poultp dlog_fishp
regr wdlog_poultq real_poult dlog_bfvlp dlog_porkp dlog_poultp dlog_fishp
regr wdlog_fishq  real_fish  dlog_bfvlp dlog_porkp dlog_poultp dlog_fishp

- *Second, we describe how to estimate the Rotterdam model (where one equation has been dropped) and we test for homogeneity and symmetry.*

*Estimate the unrestricted model*
sureg ( wdlog_bfvlq real_bfvl dlog_bfvlp dlog_porkp dlog_poultp) (wdlog_porkq real_pork dlog_bfvlp dlog_porkp dlog_poultp) (wdlog_poultq real_poult dlog_bfvlp dlog_porkp dlog_poultp)

*Save the results*
est store unrestr

*Show the Likelihood*
di e(ll)

*Define the constraint for homogeneity*
constraint 1 [wdlog_bfvlq]dlog_bfvlp = 0 - ([wdlog_porkq]dlog_porkp + [wdlog_poultq]dlog_poultp)

Define the constraints for symmetry
constraint 2 [wdlog_bfvlq]dlog_porkp  -  [wdlog_porkq]dlog_bfvlp  = 0
constraint 3 [wdlog_porkq]dlog_poultp - [wdlog_poultq]dlog_porkp  = 0
constraint 4 [wdlog_bfvlq]dlog_poultp  -  [wdlog_poultq]dlog_bfvlp  = 0

*Estimate the restricted model, save the results, and show the Likelihood*
sureg ( wdlog_bfvlq real_bfvl dlog_bfvlp dlog_porkp dlog_poultp) (wdlog_porkq real_pork dlog_bfvlp dlog_porkp dlog_poultp) (wdlog_poultq real_poult dlog_bfvlp dlog_porkp dlog_poultp), constraints(1 2 3 4)

est store restr

di e(ll)

*Compute the likelihood ratio test*
lrtest restr unrestr

(Note: the result show that the Probability is close to zero. We reject the null of restricted model)

Alternatively, we can compute the LR test as follows:

*LL unrest*:-312.12
*LL restr*: -341.05

*LR: -2*(LL restr-LL unrest)*
*R: -2*(-341.05-(-312.12))*

Therefore LR is 57.9. The Chi squared with 4 d.o.f., at 5% significance level, is 9.45. Therefore, we reject the null.

In order to compute the elasticities, we estimate the model imposing the restrictions and use the prices to compute the expenditure elasticities, the compensated elasticities and the uncompensated elasticities.

*First, generate a scalar for prices as follows:*

```
gen P_bv = 0.98
gen P_pork  = 0.77
gen P_poul  = 0.56
gen P_fish= 1.01

*Compute the expenditure elasticities*
Expenditure elasticity  for beef and veal
di [wdlog_bfvlq]real_bfvl/P_bv
Expenditure elasticity  for pork
di [wdlog_porkq]real_pork/P_pork
Expenditure elasticity  for poultry
di [wdlog_poultq]real_poult/P_poul
Expenditure elasticity  for fish
di 1-(0.46+0.13-0.15)

*Define the sample size*
gen obs=41

*Generate average variables*
gen mean_wbfvl   = xbfvl/obs
gen mean_pork     = xpork/obs
gen mean_poult    = xpoult/obs
gen mean_fish      = xfish/obs

*Compute the compensated elasticities*
di [wdlog_bfvlq]dlog_bfvlp/mean_wbfvl
di [wdlog_bfvlq]dlog_porkp/mean_wbfvl
di [wdlog_bfvlq]dlog_poult/mean_wbfvl

di [wdlog_porkq]dlog_porkp/mean_pork
di [wdlog_porkq]dlog_bfvlp/mean_pork
di [wdlog_porkq]dlog_poult/mean_pork

di [wdlog_poultq]dlog_poultp/mean_poult
di [wdlog_poultq]dlog_bfvlp/mean_poult
di [wdlog_poultq]dlog_porkp/mean_poult

*Generate the compensated elasticities*
gen e_c_bf_bf     = [wdlog_bfvlq]dlog_bfvlp/mean_wbfvl
gen e_c_bf_por   =  [wdlog_bfvlq]dlog_porkp/mean_wbfvl
gen e_c_bf_poul = [wdlog_bfvlq]dlog_poult/mean_wbfvl

gen e_c_por_por = [wdlog_porkq]dlog_porkp/mean_pork
gen e_c_por_bf    = [wdlog_porkq]dlog_bfvlp/mean_pork
gen e_c_por_poul= [wdlog_porkq]dlog_poult/mean_pork

gen e_c_poul_poul= [wdlog_poultq]dlog_poultp/mean_poult
gen e_c_poul_bf    = [wdlog_poultq]dlog_bfvlp/mean_poult
gen e_c_poul_por = [wdlog_poultq]dlog_porkp/mean_poult

Generate expenditure elasticities
```

```
gen e_bv  =  [wdlog_bfvlq]real_bfvl/P_bv
gen e_pork=  [wdlog_porkq]real_pork/P_pork
gen e_poult= [wdlog_poultq]real_poult/P_poul

*Generate the uncompensated elasticities*
gen e_uc_bf_por   =  e_c_bf_por  - mean_pork*e_bv
gen e_uc_bf_poul = e_c_bf_poul - mean_poult*e_bv

gen e_uc_por_bf    =   e_c_por_bf  - mean_wbfvl*e_pork
gen e_uc_por_poul= e_c_por_poul- mean_poult*e_pork

gen e_uc_poul_bf    =   e_c_poul_bf  - mean_wbfvl*e_poult
gen e_uc_poul_por = e_c_poul_por - mean_pork*e_poult
```