Chapter 18: Disaggregation

GROCER contains several procedures for the disaggregation of time series[1]: the Boot-Feibes-Lisman method, that provides a simple interpolation of a low frequency series alone; the Chow-Lin, Litterman, Fernandez and Denton methods that provide the distribution of aggregates using a high frequency indicator.

1. The bfl function

The function bfl derives disaggregated data by the Boot-Feibes-Lisman method: the disaggregated series is calculated by the minimisation of a quadratic function of the disaggregated series, with the function equalto the level (d=0), the first order difference (d=1) or the second order difference (d=2). A call to this function takes the form:

[y,res]=bfl(nameY,ta,d,s)

where:

- nameY is a vector, a ts or a string representing such an object
- ta, which is the type of aggregation; if ta=-1, as in the default case, then the aggregated series is supposed to be the sum of the disaggregate; if ta=0, then the aggregated series is supposed to be the mean of the disaggregate; lastly, if ta=i with i>0, then the aggregated series is supposed to be the i th value of the disaggregate;
- d is the parameter relative to the objective function to be minimize: if d=0, then this is the variance of the level of the series which is minimized; if d=1, then this is the variance of the first difference of the series which is minimized; if d=2, then this is the variance of the second difference of the series which is minimized;
- s is the factor by which the low frequency has to be multiplied to obtain the high frequency one (12 for instance to transform annual data into monthly ones)

The output of the function is twofold: the value of the high frequency disaggregate of the low frequency series and a results tlist with various results. When the indicator is a time series, then y is returned as a time series over the same period.

To have an example, load the database xesp going with Grocer and that contains annual exportations for Spain:

```
-->load('SCI\macros\grocer\db\xesp.dat')
```

And apply the function bfl to the series Y, with options set to 0 for ta (aggregate is the mean of the disaggregate), 1 for d (this is the norm of the first difference of the disaggregated) and 12 for the disaggregating factor:

```
-->[y,res] = bfl(Y,0,1,12)
```

Which gives the following value for y:

```
!   19790.14  !
!   19819.882 !
!   19879.367 !
```

[1] For the most part, these functions has been built by translating and adapting matlab programms written by Carlos Alberto Castro

```
!   19968.594 !
!   20087.564 !
!   20236.276 !
!   20414.73  !
!   20622.927 !
!   20860.866 !
!   21128.547 !
!   21425.971 !
!   21753.137 !
!   22110.046 !
!   22441.946 !
!   22748.837 !
!   23030.719 !
!   23287.592 !
!   23519.457 !
!   23726.313 !
```

2. The functions working with indicators: <u>chowlin</u>, <u>fernandez</u> and <u>litterman</u>.

The 3 functions Chow-Lin, Fernandez and Litterman methods differ only by the assumptions they make on the residuals of the relationship between the low-level aggregate and the indicator: autoregressive in the case of the Chow-Lin method; random walk in the case of Fernandez method; ARMA(1,1,0) in the case of Litterman method. Their specification is therefore very similar in GROCER.

Take function <u>chowlin</u>. The simplest call to <u>chowlin</u> takes the following form:
chowlin('endo','exo1','exo2',…, exon)

In that case, endo is the name of the low frequency aggregate and exo1,…, exon are the names of the high frequency indicators. This is in fact equivalent to the following command:
chowlin('endo','exo1','exo2',…, exon, 'ta=-1', typemin=llike','delta=sqrt(%eps)')

This formulation shows 3 options for which there are default values:

- ta, which is the type of aggregation; if ta=-1, as in the defaults case, then the aggregated series is supposed to be the sum of the disaggregate; if ta=0, then the aggregated series is supposed to be the mean of the disaggregate; lastly, if ta=I with i>0, then the aggregated series is supposed to be the I th value of the disaggregate;
- typemin, which is the function that is optimized; 2 options are possible; the default case is 'llike' for log-likelihood; the other option is 'wls' for "weighted least squares';
- delta is the increment used to calculate the numerical derivative; the default value is sqrt(%eps).

The user can also enter the factor by which the low frequency has to be multiplied to obtain the high frequency one: this done through the option 'divfq=n' where n is the factor needed to transform low frequency data into high ones (12 for instance to transform annual data into monthly ones). If this option is not given then this factor is determined automatically be the program. It gives the good number either if the low frequency and some of the high frequency data are ts or if the sizes of these data are derived one from another by around their factor: if you are not sure, better use the option 'divfq=n'!

These function have the following output: firstly, the high frequency disaggregated series; secondly is a results tlist containing the input to the function as well as various output results.

An example can be found in Grocer function chowlin_d:

```
--> load('SCI\macros\grocer\db\xesp.dat'); [y,res] = chowlin('Y','x','ta=-1','typemin=wls')
```

With the following display:

```
Chow-Lin estimation results for dependent variable: Y
number of low-frequency observations: 22
frequency conversion: annual to quarterly
number of high-frequency observation: 88
number of extrapolations: 0
number of indicators (incl. constant): 2
type of disaggregation: sum
estimation method: wls
log-likelihood = -175.20304
AIC = 9.4354866      BIC = 9.5346722
innovationnal parameter (rho): 0.99

variable coeff     t-statistic p-value
cte      857.15271 1.1814633   0.2512773
x        0.9450798 37.765527   0

                        *
                     *     *
```

And the following result for y:

```
!   5403.2908 !
!   5300.6574 !
!   4353.0462 !
!   5442.0057 !
!   5238.0203 !
!   5834.7712 !
!   6047.1281 !
!   6357.0803 !
!   6636.0948 !
!   5833.1694 !
!   5838.4641 !
!   6750.2717 !
!   6930.7772 !
!   7201.7284 !
!   6426.2006 !
!   7149.2939 !
!   8556.3442 !
…
!   23539.702 !
!   27579.35  !
!   27866.959 !
!   29223.248 !
!   25696.094 !
!   30239.699 !
!   29955.619 !
!   30033.7   !
!   25994.272 !
!   29589.409
```

When the indicator is a time series, then y is returned as a time series over the same period.

The function litterman has exactly the same calling syntax. The function fernandez too, except for the option 'typemin' that is not available because it is not relevant.

3. Estimation of several high frequency data with their annual counterpart and the high frequency available.

Grocer contains 2 functions that allow to estimate high frequency disaggregate when an indicator for each annual data and the aggregate exist at a high frequency: denton and difonzo.

The syntax is the same for the 2 functions. Take for instance denton. The simplest call to denton takes the following form:

denton(nameY, x, z, d)

Where :

- nameY is a real matrix, a string vector or a list of ts, real vectors, representing the low frequency series that will be disaggregated;
- x is is a real matrix, a string vector or a list of ts, real vectors, representing the high frequency indicators (one for each annual series);
- z is a ts, a real vector or a string representing a ts or a real vector; z is the high frequency sum of the unknown high frequency series;
- d is the parameter relative to the objective function to be minimize: if d=0, then this is the variance of the level of the series which is minimized; if d=1, then this is the variance of the first difference of the series which is minimized; if d=2, then this is the variance of the second difference of the series which is minimized;

Two options can be entered into the function:

- the option 'divfq=n', where n indicates as for functions described in part 2 the factor needed to transform low frequency data into high ones;
- ta, which is the type of aggregation; if ta=-1, as in the default case, then the aggregated series is supposed to be the sum of the disaggregate; if ta=0, then the aggregated series is supposed to be the mean of the disaggregate; lastly, if ta=i with i>0, then the aggregated series is supposed to be the i th value of the disaggregate;

The function as the following output: a matrix y where each column is the high frequency disaggregate of the corresponding series in x; a results tlist that contains various input and output results. An example can be found in function denton_d.

3. The disaggregation functions and their specifications.

aggreg1 _____ temporal aggregation matrix

CALLING SEQUENCE

[C]=aggreg1(N,s,opt)

PARAMETERS

INPUT:
* N = number of low frequency data
* s = freq. conversion
* opt = type of temporal aggregation:
  - opt = -1 ---> sum (flow)
  - opt = 0 ---> average (index)
  - opt = k ---> k-th element (k>=1 and k<=s)

OUTPUT:
C = (N x sN) temporal aggregation matrix

DESCRIPTION
Generates a temporal aggregation matrix.

Examples
1) aggreg1(100,3,-1)
2) aggreg1(150,12,12)

Example 1 provides an aggregation matrix for a series of length 100, with a frequency conversion of 3 (from annual to quarterly for instance) and where the aggregation is made by taking the sum of the high frequency components.
Example 2 provides an aggregation matrix for a series of length 150, with a frequency conversion of 12 (from annual to monthly for instance) and where the aggregation is made by taking the 12th element of the high frequency components.

bf1_____ Boot-Feibes-Lisman disaggregation method

CALLING SEQUENCE
[y,res]=bfl(nameY,ta,d,s)

PARAMETERS

INPUT:
* nameY = a ts or a Nx1 vector of low frequency data or a string representing the name of a vector or a ts
* ta = type of disaggregation
  - ta=-1 ---> sum (flow)
  - ta=0 ---> average (index)
  - ta=i ---> i th element (stock) ---> interpolation
* d = objective function to be minimized: volatility of ...
  - d=0 ---> levels
  - d=1 ---> first differences
  - d=2 ---> second differences
* s = number of high frequency data points for each low frequency data point
  - s= 4 ---> annual to quarterly
  - s=12 ---> annual to monthly

- s= 3 ---> quarterly to monthly

OUTPUT:
* y = High frequency estimate
* res = a results tlist with:
  - res('meth') = 'Boot-Feibes-Lisman'
  - res('nobs_lf') = Number of low frequency data
  - res('aggreg_mode') = Type of disaggregation
  - res('s') = Frequency conversion
  - res('diff') = Degree of differencing
  - res('y_lf') = Low frequency data
  - res('y') = High frequency estimate
  - res('namey') = name of the low frequency data

DESCRIPTION
Performs temporal disaggregation using the Boot-Feibes-Lisman method.

Example
load('SCI\macros\grocer\db\xesp.dat'); [y,res] = bfl(Y,0,1,12);

Example taken from function bfl_d. Provides the minimisation of volatility of the first difference of monthly Spanish exportations data, the annual series being the sum of the monthly series.

bf11_____ Boot-Feibes-Lisman disaggregation method

CALLING SEQUENCE
[y,res]=bfl1(Y,ta,d,s)

PARAMETERS

INPUT:
* Y = Nx1 vector of low frequency data
* ta = type of disaggregation
  - ta=-1 ---> sum (flow)
  - ta=0 ---> average (index)
  - ta=i ---> i th element (stock) ---> interpolation
* d = objective function to be minimized: volatility of ...
  - d=0 ---> levels
  - d=1 ---> first differences
  - d=2 ---> second differences
* s = number of high frequency data points for each low frequency data point
  - s= 4 ---> annual to quarterly
  - s=12 ---> annual to monthly
  - s= 3 ---> quarterly to monthly

OUTPUT:
* y = High frequency estimate
* res = a results tlist with:
  - res('meth') = 'Boot-Feibes-Lisman';
  - res('nobs_lf') = Number of low frequency data
  - res('aggreg_mode') = Type of disaggregation
  - res('s') = Frequency conversion
  - res('diff') = Degree of differencing
  - res('y_lf') = Low frequency data
  - res('y') = High frequency estimate

DESCRIPTION
Performs temporal disaggregation using the Boot-Feibes-Lisman method. This is the low level version that works with only a vector.

Example
1) load('SCI\macros\grocer\db\xesp.dat'); [y,res] = bfl(Y,0,1,12);
2) [y,res]=bfl1(Y,ta,d,s)

Example 1 provides the minimisation of volatility of the first difference of monthly Spanish exportations data, the annual series being the sum of the monthly series. Example 2 is taken from function bfl.

chowlin_____ Chow-Lin disaggregation method

CALLING SEQUENCE
[y,res]=chowlin(namey,argi1,…,argn)

PARAMETERS

INPUT:
* namey = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quote, representing the low frequency data that must be disaggregated
* arg1,…, argn = arguments which can be:
  . a time series
  . a real (nx1) vector
  . a string equal to the name of a time series or a (nx1) real vector between quotes
  . a list of such objects
  . the string 'divfq=n' where n is the number of high frequency data points for each low frequency data points
  . the string 'typemin=xxx' where xxx is the maximisation method (llike -default- or wls)
  . the string 'ta=n' where n is the aggregation type:
    n=-1 (default) ---> sum (flow)
    n=0 ---> average (index)
    n=i ---> i th element (stock) ---> interpolation
  . the string 'delta=x' where x is the increment used for numerical derivation
  ---
OUTPUT:
* y = the disaggregated variable

* res = a results tlist with:
- res('meth') = 'Chow-Lin'
- res('ta') = type of disaggregation
- res('nobs_lf') = nobs. of low frequency data
- res('nobs_hf') = nobs. of high-frequency data
- res('pred') = number of extrapolations
- res('s') = frequency conversion between low and high freq.
- res('p') = number of regressors (including intercept)
- res('y_lf') = low frequency data
- res('indicator') = high frequency indicators
- res('y') = high frequency estimate
- res('y_dt') = high frequency estimate: standard deviation
- res('y_up') = high frequency estimate: sd + sigma
- res('y_lo') = high frequency estimate: sd - sigma
- res('resid_hf') = high frequency residuals
- res('resid_lf) = low frequency residuals
- res('beta') = estimated model parameters
- res('sd') = estimated model parameters: standard deviation
- res('tstat') = estimated model parameters: t ratios
- res('rho') = innovational parameter
- res('aic') = Information criterion: AIC
- res('bic') = Information criterion: BIC
- res('typemin') = method of estimation
- res('llike') = Log-likelihood at the estimated parameters
- res('sigma') = Variance at the estimated parameters
- res('namey') = Name of the high frequency aggregate
- res('namex') = Name of the low frequency indicators
- res('prests') = a boolean indicating the presence or absence of a time series in the regression
- res('bounds') = if there is a timeseries in the regression, the bounds of the regression

DESCRIPTION
Performs Temporal disaggregation using the Chow-Lin method (high level function with vectors, matrices or ts and the possibility of default parameters).

Example
1) load('SCI\macros\grocer\db\xesp.dat') ; [y,res] = chowlin('Y','x','ta=-1','typemin=wls');
2) load('SCI\macros\grocer\db\xesp.dat') ; [y,res] = chowlin('Y','x','ta=-1','typemin=llike');

Provides the disaggregation of the Spain's Exports of Goods by the Chow-lin method, using Spain's Registered exports of goods deflated by unit value index as an indicator. In example 1, the weighted least squares method is used to estimate the autocorrelation parameter, while the maximum likelihood method is used in example 2. The annual data are the sum of the monthly data (ta=-1).

chowlin1_____ Chow-Lin disaggregation method

CALLING SEQUENCE
[y,res]=fernandez(namey, arg1,…,argn)

PARAMETERS

INPUT:
* namey = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes, representing the low frequency data that must be desaggregated
* arg1,…,argn = arguments which can be:
  . a time series
  . a real (nxk) matrix
  . a string matrix whose elements represent the names of a time series or a (nx1) real vector between quotes
  . a list of such objects
  . the string 'divfq=n' where n is the number of high frequency data points for each low frequency data points (default: recovered from the data)
  . the string 'ta=n' where n is the aggregation type:
    n=-1 (default) ---> sum (flow)
    n=0 ---> average (index)
    n=i ---> i th element (stock) ---> interpolation

OUTPUT:
* y = the disaggregated variable
* res = a results tlist with:
  - res('meth') ='Fernandez';
  - res('ta') = type of disaggregation
  - res('nobs_lf') = nobs. of low frequency data
  - res('nobs_hf') = nobs. of high-frequency data
  - res('pred') = number of extrapolations
  - res('s') = frequency conversion between low and high freq.
  - res('p') = number of regressors (including intercept)
  - res('y') = high frequency estimate
  - res('y_lf') = low frequency data
  - res('indicator') = high frequency indicators
  - res('y_dt') = high frequency estimate: standard deviation
  - res('y_up') = high frequency estimate: sd + sigma
  - res('y_lo') = high frequency estimate: sd - sigma
  - res('resid') = high frequency residuals
  - res('resid_lf') = low frequency residuals
  - res('beta') = estimated model parameters
  - res('sd') = estimated model parameters: standard deviation
  - res('tstat') = estimated model parameters: t ratios
  - res('aic') = Information criterion: AIC
  - res('bic') = Information criterion: BIC
  - res('namey') = Name of the low frequency disaggregated data
  - res('namex') = Name of the indicators

DESCRIPTION
Performs Temporal disaggregation using the Chow-Lin method (low level function that works only with vectors and matrices).

Example
[y,res]=chowlin1(Y,x,grocer_ta,grocer_divfq,grocer_delta,grocer_typemin)

Example taken from function chowlin.

denton _____Multivariate temporal disaggregation with transversal constraint

CALLING SEQUENCE
[y,res]=denton(Y,x,z,d,arg1,…,argn)

PARAMETERS

INPUT:
* Y = a (NxM) matrix, a list of M vectors or ts or a string vector representing names such objects
  ---> M series of low frequency data with N observations
* x = a (nxM) matrix, a list of M vectors or ts or a string vector representing names such objects
  ---> M series of high frequency data with n observations
* z = a (nx1) vector or a ts ---> high frequency transversal constraint
* d = objective function to be minimized: volatility of
  - - d=0 ---> levels
  - - d=1 ---> first differences
  - - d=2 ---> second differences
* arg1,…,argn = options to denton:
  . the string 'divfq=n' where n is the number of high frequency data points for each low frequency data point
  (default: recovered from the data)
  . the string 'ta=n' where n is the aggregation type:
  n=-1 (default) ---> sum (flow)
  n=0 ---> average (index)
  n=i ---> i th element (stock) ---> interpolation

OUTPUT:
* y = the disaggregated variable
* res = a results tlist with
  - res('meth') = 'Multivariate Denton';
  - res('ta') = Type of disaggregation
  - res('nobs_lf') = Number of low frequency data
  - res('nobs_hf') = Number of high frequency data
  - res('pred') = Number of extrapolations (=0 in this case)
  - res('s') = Frequency conversion
  - res('diff') = Degree of differencing
  - res('y') = High frequency estimate
  - res('y_lf') = low frequency data
  - res('indicator') = high frequency indicators
  - res('tanvsersal') = data for the transversal constraint
  - res('namey') = Name of the low frequency aggregate
  - res('namex') = Name of the high frequency indicators
  - res('namez') = Name of the high frequency transversal constraint

DESCRIPTION
Performs multivariate temporal disaggregation using Denton method (high level function with vectors, matrices or ts and the possibility of default parameters).

Example
[y,res]=denton('Y','x','z',1,'ta=-1','divfq=4');

Provides the quarterly (since divfq=4) disaggregation of Spain's annual exports of Goods (Y) by Denton method, using Spain's Registered exports of goods deflated by unit value index (x) as an indicator. The vector of constraints is supposed to be called z. Annual data are supposed to be the sum of quarterly ones (ta=-1).

denton1_____Multivariate temporal disaggregation with transversal constraint

CALLING SEQUENCE
[y,res]=denton1(Y,x,z,d,ta,s)

PARAMETERS

INPUT:
* Y = (NxM) matrix ---> M series of low frequency data with N observations
* x = (nxM) matrix ---> M series of high frequency data with n observations
* z = (nx1) vector ---> high frequency transversal constraint
* d = objective function to be minimized: volatility of
  - d=0 ---> levels
  - d=1 ---> first differences
  - d=2 ---> second differences
* ta = type of disaggregation
  - ta=-1 ---> sum (flow)
  - ta=0 ---> average (index)
  - ta=i ---> i th element (stock) ---> interpolation
* s = number of high frequency data points for each low frequency data point
  - s= 4 ---> annual to quarterly
  - s=12 ---> annual to monthly
  - s= 3 ---> quarterly to monthly

OUTPUT:
* y = the disaggregated variable
* res = a results tlist with
  - res('meth') = 'Multivariate Denton';
  - res('ta') = Type of disaggregation
  - res('nobs_lf') = Number of low frequency data
  - res('nobs_hf') = Number of high frequency data
  - res('pred') = Number of extrapolations (=0 in this case)
  - res('s') = Frequency conversion
  - res('diff') = Degree of differencing
  - res('y') = High frequency estimate

- res('y_lf') = low frequency data
- res('indicator') = high frequency indicators
- res('tanvsersal') = data for the transversal constraint

DESCRIPTION
Performs multivariate temporal disaggregation using Denton method (low level function with vectors, and matrices only).

Example
[y,res]=denton1(Y,x,z,grocer_d,grocer_ta,grocer_divfq)

Example taken from function denton.

difonzo_____Multivariate temporal disaggregation with transversal constraint

CALLING SEQUENCE
[y,res]=difonzo(Y,x,z, arg1,…,argn)

PARAMETERS

INPUT:
* Y = list of N vectors or ts or (NxM) matrix
  ---> M series of low frequency data with N observations
* x = (nxM) real or string matrix or list
  ---> M series of high frequency data with n observations
* z = (nx1) vector or ts ---> high frequency transversal constraint
* d = objective function to be minimized: volatility of
  - d=0 ---> levels
  - d=1 ---> first differences
  - d=2 ---> second differences
* arg1, …, argn = options to difonzo:
  . the string 'divfq=n' where n is the number of high frequency data points for each low frequency data points (default: recoverd from the data)
  . the string 'ta=n' where n is the aggregation type:
    n=-1 (default) ---> sum (flow)
    n=0 ---> average (index)
    n=i ---> i th element (stock) ---> interpolation
  . typemod = model for the high frequency innvations
    typemod='wn' ---> multivariate white noise (default)
    typemod='rw' ---> multivariate random walk

OUTPUT:
* y = the disaggregated variable
* res= a result tlist with:
  - res('meth') = 'Multivariate di Fonzo'
  - res('ta') = type of disaggregation
  - res('nobs_lf') = nobs of low frequency data

- res('nobs_hf') = nobs of high-frequency data
- res('pred') = number of extrapolations
- res('s') = frequency conversion between low and high freq
- res('y') = high frequency estimate
- res('y_lf') = low frequency data
- res('indicator') = high frequency indicators
- res('transversal') = high frequency indicators
- res('y_dt') = high frequency estimate: standard deviation
- res('resid') = high frequency residuals
- res('resid_U') = low frequency residuals
- res('beta') = estimated model parameters
- res('sd') = standard deviation of the estimated model parameters
- res('namey') = Name of the low frequency aggregate
- res('namex') = Name of the high frequency indicators
- res('namez') = Name of the high frequency transversal constraint

DESCRIPTION
Performs multivariate temporal disaggregation using Di Fonzo's method (high level function with vectors, matrices or ts and the possibility of default parameters).

Example
difonzo(Y,x,z,'ta=-1','typemod=rw');

Example taken from function difonzo_d. Provides the disaggregation of Spain's annual exports of Goods (Y) by Denton method, using Spain's Registered exports of goods deflated by unit value index (x) as an indicator. The vector of constraints is supposed to be called z. Annual data are supposed to be the sum of low frequency ones (ta=-1).

difonzo1_____Multivariate temporal disaggregation with transversal constraint

CALLING SEQUENCE
[y,res]=difonzo1(Y,x,z,ta,s,typemod)

PARAMETERS

INPUT:
* Y = Nx1 ---> vector of low frequency data
* x = nxp ---> matrix of high frequency indicators
  (without intercept)
* ta = type of disaggregation:
  - ta = -1 ---> sum (flow)
  - ta = 0 ---> average (index)
  - ta = k ---> k th element ---> interpolation
* s = number of high frequency data points for each low frequency data point:
  - s= 4 ---> annual to quarterly
  - s=12 ---> annual to monthly
  - s= 3 ---> quarterly to monthly

* delta = the increment used to evaluate the derivative
* typemin estimation method:
  typemin='wls' ---> weighted least squares
  typemin='llike' ---> maximum likelihood

OUTPUT:
* y = the disaggregated variable
* res= a result tlist with:
- res('meth') = 'Multivariate di Fonzo'
- res('ta') = type of disaggregation
- res('nobs_lf') = nobs of low frequency data
- res('nobs_hf') = nobs of high-frequency data
- res('pred') = number of extrapolations
- res('s') = frequency conversion between low and high freq
- res('y') = high frequency estimate
- res('y_lf') = low frequency data
- res('indicator') = high frequency indicators
- res('transversal') = high frequency indicators
- res('y_dt') = high frequency estimate: standard deviation
- res('resid') = high frequency residuals
- res('resid_U') = low frequency residuals
- res('beta') = estimated model parameters
- res('sd') = standard deviation of the estimated model parameters

DESCRIPTION
Performs multivariate temporal disaggregation using Di Fonzo's method (low level function with vectors, and matrices only).

Example
[y,res]=difonzo1(Y,x,z,grocer_ta,grocer_divfq,grocer_typemod)

Example taken from function difonzo.

explo_agreg_____explosion of variables for a disaggregation estimation

CALLING SEQUENCE
[Y,x,namendo,namexos,s]=explo_agreg(ta,namey, arg1,…,argn)

PARAMETERS

INPUT:
* ta = type of disaggregation:
- ta = -1 ---> sum (flow)
- ta = 0 ---> average (index)
- ta = k ---> k th element ---> interpolation
* namey = low frequency data
* arg1,…argn = a collection of ts, vectors, matrices or strings or lists of such objects

---
OUTPUT:
* Y = (Nx1) real vector of low frequency data
* x = (Nxp) real matrix of high frequency indicators (without intercept)
* nameY = a string name of low frequency data
* x = (nxp) string matrix of names for the high frequency indicators
* s = number of high frequency data points for each low frequency data points:
  - s= 4 ---> annual to quarterly
  - s=12 ---> annual to monthly
  - s= 3 ---> quarterly to monthly

DESCRIPTION
Generates matrices associated to the exogenous and endogenous variables of a disaggregation problem, the associated names and the conversion from low to high frequency parameter.

Example
[Y,x,grocer_namendo,grocer_namexos,grocer_divfq]=explo_agreg(grocer_ta,grocer_namey,varargin(:))

Example taken from function chowlin.

fernandez_____Temporal disaggregation using the Fernandez method

CALLING SEQUENCE
[y,res]=difonzo(Y,x,z, arg1,…,argn)

PARAMETERS

INPUT:
* Y = list of N vectors or ts or (NxM) matrix
  ---> M series of low frequency data with N observations
* x = (nxM) real or string matrix or list
  ---> M series of high frequency data with n observations
* z = (nx1) vector or ts ---> high frequency transversal constraint
* d = objective function to be minimized: volatility of
  - d=0 ---> levels
  - d=1 ---> first differences
  - d=2 ---> second differences
* arg1, …, argn = options to difonzo:
  . the string 'divfq=n' where n is the number of high frequency data points for each low frequency data points (default: recoverd from the data)
  . the string 'ta=n' where n is the aggregation type:
    n=-1 (default) ---> sum (flow)
    n=0 ---> average (index)
    n=i ---> i th element (stock) ---> interpolation
  . typemod = model for the high frequency innvations
    typemod='wn' ---> multivariate white noise (default)
    typemod='rw' ---> multivariate random walk

OUTPUT:
* y = the disaggregated variable
* res= a result tlist with:
  - res('meth') = 'Multivariate di Fonzo'
  - res('ta') = type of disaggregation
  - res('nobs_lf') = nobs of low frequency data
  - res('nobs_hf') = nobs of high-frequency data
  - res('pred') = number of extrapolations
  - res('s') = frequency conversion between low and high freq
  - res('y') = high frequency estimate
  - res('y_lf') = low frequency data
  - res('indicator') = high frequency indicators
  - res('transversal') = high frequency indicators
  - res('y_dt') = high frequency estimate: standard deviation
  - res('resid') = high frequency residuals
  - res('resid_U') = low frequency residuals
  - res('beta') = estimated model parameters
  - res('sd') = standard deviation of the estimated model parameters
  - res('namey') = Name of the low frequency aggregate
  - res('namex') = Name of the high frequency indicators
  - res('namez') = Name of the high frequency transversal constraint
  - res('prests') = a boolean indicating the presence or absence of a time series in the regression
  - res('bounds') = if there is a timeseries in the regression, the bounds of the regression

DESCRIPTION
Performs temporal disaggregation using the Fernandez method (high level function with vectors, matrices or ts and the possibility of default parameters).

Example
load('SCI\macros\grocer\db\xesp.dat'); [y,res] = fernandez(Y,x,'ta=-1');

Example taken from function fernandez_d. Provides the disaggregation of Spain's annual exports of Goods (Y) by Fernandez method, using Spain's Registered exports of goods deflated by unit value index (x) as an indicator. Annual data are supposed to be the sum of low frequency ones (ta=-1).

fernandez1_____Temporal disaggregation using the Fernandez method

CALLING SEQUENCE
[y,res]=fernandez1(Y,x,ta,s)

PARAMETERS

INPUT:
* Y = Nx1 ---> vector of low frequency data
* x = nxp ---> matrix of high frequency indicators (without intercept)
* ta = type of disaggregation:

- ta = -1 ---> sum (flow)
- ta = 0 ---> average (index)
- ta = k ---> k th element ---> interpolation

* s = number of high frequency data points for each low frequency data points:
- s= 4 ---> annual to quarterly
- s=12 ---> annual to monthly
- s= 3 ---> quarterly to monthly

OUTPUT:
* y = the disaggregated variable
* res= a result tlist with:
- res('meth') = 'Multivariate di Fonzo'
- res('ta') = type of disaggregation
- res('nobs_lf') = nobs of low frequency data
- res('nobs_hf') = nobs of high-frequency data
- res('pred') = number of extrapolations
- res('s') = frequency conversion between low and high freq
- res('y') = high frequency estimate
- res('y_lf') = low frequency data
- res('indicator') = high frequency indicators
- res('transversal') = high frequency indicators
- res('y_dt') = high frequency estimate: standard deviation
- res('resid') = high frequency residuals
- res('resid_U') = low frequency residuals
- res('beta') = estimated model parameters
- res('sd') = standard deviation of the estimated model parameters

DESCRIPTION
Performs temporal disaggregation using the Fernandez method (low level function with vectors, and matrices only).

Example
[y,res]=fernandez1(Y,x,grocer_ta,grocer_divfq)

Example taken from function fernandez.

litterman_____Temporal disaggregation using the Litterman method

CALLING SEQUENCE
[y,res]=litterman(namey, arg1,…,argn)

PARAMETERS

INPUT:
* namey = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quote, representing the low frequency data that must be disaggregated
* arg1,…,argn = arguments which can be:

. a time series
. a real (nx1) vector
. a string equal to the name of a time series or a (nx1) real vector between quotes
. a list of such objects
. the string 'divfq=n' where n is the number of high frequency data points for each low frequency data points (default: recoverd from the data)
. the string 'typemin=xxx' where xxx is the maximisation method (llike -default- or wls)
. the string 'ta=n' where n is the aggregation type:
n=-1 (default) ---> sum (flow)
n=0 ---> average (index)
n=i ---> i th element (stock) ---> interpolation
. the string 'delta=x' where x is the increment used for numerical derivation
---

OUTPUT:
* y = the disaggregated variable
* res = a results tlist with:
- res('meth') = 'Litterman'
- res('ta') = type of disaggregation
- res('nobs_lf') = nobs. of low frequency data
- res('nobs_hf') = nobs. of high-frequency data
- res('pred') = number of extrapolations
- res('s') = frequency conversion between low and high freq.
- res('p') = number of regressors (including intercept)
- res('y_lf') = low frequency data
- res('indicator') = high frequency indicators
- res('y') = high frequency estimate
- res('y_dt') = high frequency estimate: standard deviation
- res('y_up') = high frequency estimate: sd + sigma
- res('y_lo') = high frequency estimate: sd - sigma
- res('resid') = high frequency residuals
- res('resid_lf) = low frequency residuals
- res('beta') = estimated model parameters
- res('sd') = estimated model parameters: standard deviation
- res('tstat') = estimated model parameters: t ratios
- res('rho') = innovational parameter
- res('aic') = Information criterion: AIC
- res('bic') = Information criterion: BIC
- res('llike') = Objective function used by the estimation method
- res('typemin') = method of estimation
- res('llike') = Log-likelihood at the estimated parameters
- res('sigma') = Variance at the estimated parameters
- res('namey') = Name of the low frequency aggregate
- res('namex') = Name of the high frequency indicators
- res('prests') = a boolean indicating the presence or absence of a time series in the regression
- res('bounds') = if there is a timeseries in the regression, the bounds of the regression

DESCRIPTION

Performs temporal disaggregation using the Litterman method (high level function with vectors, matrices or ts and the possibility of default parameters).

Example
load('SCI\macros\grocer\db\xesp.dat'); [y,res] = litterman(Y,x,'ta=-1','typemin=wls');

Example taken from function litterman_d. Provides the disaggregation of the Spain's Exports of Goods by the Chow-lin method, using Spain's Registered exports of goods deflated by unit value index as an indicator. The weighted least squares method is used to estimate the autocorrelation parameter. The annual data are the sum of the monthly data (ta=-1).

litterman1_____Temporal disaggregation using the Litterman method

CALLING SEQUENCE
[y,res]=litterman1(Y,x,ta,s,delta,typemin)

PARAMETERS

INPUT:
* Y = Nx1 ---> vector of low frequency data
* x = nxp ---> matrix of high frequency indicators (without intercept)
* ta = type of disaggregation
  ta=-1 ---> sum (flow)
  ta=0 ---> average (index)
  ta=i ---> i th element (stock) ---> interpolation
* s = number of high frequency data points for each low frequency data point
  s= 4 ---> annual to quarterly
  s=12 ---> annual to monthly
  s= 3 ---> quarterly to monthly
* typemin = estimation method:
  typemin='wls' ---> weighted least squares
  typemin='llike' ---> maximum likelihood

OUTPUT:
* y = the disaggregated variable
* res = a results tlist with:
  - res('meth') = 'Litterman'
  - res('ta') = type of disaggregation
  - res('nobs_lf') = nobs. of low frequency data
  - res('nobs_hf') = nobs. of high-frequency data
  - res('pred') = number of extrapolations
  - res('s') = frequency conversion between low and high freq.
  - res('p') = number of regressors (including intercept)

- res('y_lf') = low frequency data
- res('indicator') = high frequency indicators
- res('y') = high frequency estimate
- res('y_dt') = high frequency estimate: standard deviation
- res('y_up') = high frequency estimate: sd + sigma
- res('y_lo') = high frequency estimate: sd - sigma
- res('resid') = high frequency residuals
- res('resid_lf) = low frequency residuals
- res('beta') = estimated model parameters
- res('sd') = estimated model parameters: standard deviation
- res('tstat') = estimated model parameters: t ratios
- res('rho') = innovational parameter
- res('aic') = Information criterion: AIC
- res('bic') = Information criterion: BIC
- res('llike') = Objective function used by the estimation method
- res('typemin') = method of estimation
- res('llike') = Log-likelihood at the estimated parameters
- res('sigma') = Variance at the estimated parameters

DESCRIPTION
Performs temporal disaggregation using the Litterman method (low level function with vectors, and matrices only).

Example
[y,res]=litterman1(Y,x,grocer_ta,grocer_divfq,grocer_delta,grocer_typemin)

Example taken from function litterman.