# DYNAMIC MODELS

As was presented in class and is visible from presentation, this part of our course is not particularly demanding as far as STATA is concerned. Nevertheless, below you can find 'translations' from normal language to STATA programming (should anyone need that).

### Unrestricted approach

In the unrestricted approach, model estimation is no different to standard OLS. STATA 8 automatically generates lagged values if we use an appropriate `l.number` lag operator in the estimation command - we do not need to be creating these series before using `generate` or `gen` commands. In other words, a standard syntax for first four lags should be constructed as follows (`regress` command is equivalent to `reg` in STATA):

```
reg y x1 l.x1 l2.x1 l3.x1 l4.x1 x2 x3 ...
```

Interestingly, one can also apply difference operators automatically. That is, by typing for example `d4.x1` we tell STATA to compute $x_t - x_{t-4}$ for every $t$. This is, by the way, very useful to easily get inflation time series from price level observations. Just as with the lag operators, if used with `regress` or any other estimation command, differences do not need to be generated, as it happens automatically.

### Arithmetic lags

To perform arithmetic lags, owing to the fact that we are applying some algebraic transformations, we need to perform some `generate` and `store` operations. Using the examples from the slides, we need to type:

```
gen z=5*x + 4*l.x + 3*l2.x + 2*l3.x + l4.x.
```

Sometimes, especially in the older STATA versions, with many lags it is better to create each lag separately and only then sum them up. Normally, however, there should be no problems.

After this step, to finally do the regression we need, we standardly apply `reg` command to the new generated variable:

```
regress y z
```

The estimator by $z$ coefficient will tell us exactly the $\gamma$ value and thus the descending order (the slope of this line).

## Polynomial lags (a.k.a. Almon lags)

Polynomial lags, in terms of performing STATA operations, are not particularly different from the arithmetic lags. The main difference is that we need to generate more 'new' variables to be able to finally run the regression. Sticking to the example laid out in the slides, let's specify a model:

$$\begin{aligned} y_t = \quad & \alpha + \gamma_0 x_t + (\gamma_0 + \gamma_1 + \gamma_2)x_{t-1} + (\gamma_0 + 2\gamma_1 + 4\gamma_2)x_{t-2} + \\ & (\gamma_0 + 3\gamma_1 + 9\gamma_2)x_{t-3} + (\gamma_0 + 4\gamma_1 + 16\gamma_2)x_{t-4} + e_t. \end{aligned}$$

On this model, we perform necessary algebraic operations to retain the actual values of $z_0$, $z_1$ and $z_2$ as follows from the transformation (see the slides for details).

Having done that, we are ready to perform `generate` on these variables, following the derived relations. In particular:

```
gen z0=x + l.x + l2.x + l3.x + l4.x
gen z1=x + l.x + 2*l2.x + 3*l3.x + 4*l4.x
gen z2=x + l.x + 4*l2.x + 9*l3.x + 16*l4.x.
```

These new, generated variables may now be used in the actual regression with the interpretation of the estimated parameters as indicated in the initial equation above. This holds, because we actually estimate $y_t = \alpha + \gamma_0 z_0 + \gamma_1 z_1 + \gamma_2 z_2 + e_t$.

Please, note that all we do is just algebraic operations and thus we **do not** influence the error term in any way. That is exactly why this is still 'the same' error term.

## Geometric lags and Koyck transformation

Koyck transformation is used to allow for infinite lags (these are the data, who 'decide' about the number of lags - 'they' do that implicitly, because we are not really considering the number of lags at all). There is a trade-off to it, but rather marginally painful. In addition, Koyck transformation may be improved to allow for some other (less consistent) fading out patterns.

Assume you wanted *a priori* allow for an infinite number of lags in your equation. You obviously cannot do that in the unrestricted approach, as we would loose all the data. In the structured approach, however, if we decide to impose on data geometric fading out, we might actually get away with 'infinity' due to the handy property of convergence in geometric series: $\beta_i = \beta * \phi^i < \infty$ and this is true for all $|\phi| < 1$. Thus, no longer fearing explosion, armed in a fading out mechanism and willing to accept the risk of misspecified descending pattern, we can start applying the Koyck transformation.

The rule of thumb designed by Koyck tells the following:

1. Take the original equation of the form $y_t = \alpha + \beta_0 x_t + \beta_1 x_{t-1} + \beta_2 x_{t-2} ... + e_t$ and transform it using the substitution of $\beta_i = \beta * \phi^i$ to obtain:

   $y_t = \alpha + \beta(x_t + \phi x_{t-1} + \phi^2 x_{t-2} + \phi^3 x_{t-3} + ...) + e_t$

2. Lag everything once and multiply by $\phi$.

   $\phi y_{t-1} = \phi\alpha + \phi\beta(x_{t-1} + \phi x_{t-2} + \phi^2 x_{t-3} + \phi^3 x_{t-4} + ...) + \phi e_{t-1}$

3. Substract one from the other and see what happens :)...

   $y_t - \phi y_{t-1} = \alpha(1 - \phi) + \beta x_t + (e_t - \phi e_{t-1})$

   $y_t = \alpha(1 - \phi) + \phi y_{t-1} + \beta x_t + (e_t - \phi e_{t-1})$

   $y_t = \alpha(1 - \phi) + \phi y_{t-1} + \beta x_t + v_t$

Voila, we obtained a form we can easily run in STATA and which *de facto* IS a infinite lag form. However, there is one problem. If we run OLS on this equation, the estimators will not be consistent as $v_t$ is by definition dependent on $y_{t-1}$. Thus, OLS will produce totally unreliable results. Gladly, there is solution to this provided by two-stage-least-squares estimation.

To define a step-by-step procedure, one should first instrument for $y_{t-1}$ and only then run a regression on these fitted values (among other). An instrument, *i.e.* a variable that is well correlated. It does not need to be a causal relationship, it even does not have to be a very reasonable relationship. It is just enough that the instrument will be well contemporaneously correlated with $y_{t-1}$, while at the same time being orthogonal to $e_t$. Gladly, we have a perfect candidate to this role - that is $x_{t-1}$. It is thus enough that we run the following commands in STATA:

```
regress l.y l.x
predict xb
est store fittedy
regress y l.fittedy x
```

and this is it. One can also perform a 2SLS procedure on STATA (the results would be exactly the same). This way we obtain the method of consistently estimating infinite lag effects under the Koyck transformation.

## Adaptive expectations and partial adjustment

Although Koyck transformation combined with 2SLS allows for infinite lags at a rather negligible cost of geometric fading out, sometimes and for some data this may be an inappropriate

approach. For instance, in adaptive expectations models (like SAS-DAD or Lucas models in macroeconomics) one should allow for adaptive (that is error-correcting) expectations in the dynamic form. We may also want to allow for gradual adjustment (for instance in the case of inventories, when it is costly to change the *status quo*).

Both of these approaches are easily incorporated into dynamic modelling and result in testable forms similar to Koyck transformation with consistent 2SLS estimators. To see how it works you may try to consider a model where $y_t = \alpha + \beta\tilde{x}_t + e_t$, where tilda denotes expectations. Let's assume the expectation mechanism to be based on the past observation and the error of past observation: $\tilde{x}_t - \tilde{x}_{t-1} = \lambda(x_{t-1} - \tilde{x}_{t-1})$. The other case (partial adjustment) differs from the original case in $y_t - y_{t-1} = \lambda(\tilde{y}_t - y_{t-1})$, where $\lambda$ denotes the fraction by which adjustment occurs.

Try to obtain the testable forms of the two problems and code that into STATA commands step-by-step. This is the assignment this time :).