State-space models and the Kalman filter

Riccardo (Jack) Lucchetti

UNIVPM/DISES

13-12-2018

Let's start from an unobservable stochastic column vector with r elements $\boldsymbol{\xi}_1$, which is generated according to the following rule

$$\boldsymbol{\xi}_1 = \mathbf{F} \cdot \boldsymbol{\xi}_0 + \mathbf{v},$$

where $\boldsymbol{\xi}_0$ is a known vector of constants and $\mathbf{v} \sim R\left[0, \mathbf{Q}\right]$.
Our best guess on $\boldsymbol{\xi}_1$ is $\widetilde{\boldsymbol{\xi}_1} \equiv \mathbf{F}\boldsymbol{\xi}_0$.

Let's start from an unobservable stochastic column vector with r elements $\boldsymbol{\xi}_1$, which is generated according to the following rule

$$\xi_1 = \mathbf{F} \cdot \xi_0 + \mathbf{v},$$

where ξ_0 is a known vector of constants and $\mathbf{v} \sim R\left[0, \mathbf{Q}\right]$.
Our best guess on ξ_1 is $\widetilde{\xi_1} \equiv \mathbf{F}\xi_0$.

Now enter a third observable vector with n elements, $\mathbf{y}$, of which we know the following:

$$\mathbf{y} = \mathbf{H}'\xi_1 + \mathbf{w}$$

where $\mathbf{w} \sim R\left[0, \mathbf{R}\right]$ ($\mathbf{R}$ is also known). For now, let's assume for simplicity that $\mathbf{v}$ and $\mathbf{w}$ are uncorrelated.

.

The GLS solution. Let's organise what we know in the following system of equations:

$$\begin{bmatrix} \widetilde{\xi}_1 \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ \mathbf{H}' \end{bmatrix} \xi_1 + \begin{bmatrix} -\mathbf{v} \\ \mathbf{w} \end{bmatrix}$$

.

The GLS solution. Let's organise what we know in the following system of equations:

$$\begin{bmatrix} \widetilde{\xi}_1 \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ \mathbf{H}' \end{bmatrix} \xi_1 + \begin{bmatrix} -\mathbf{v} \\ \mathbf{w} \end{bmatrix}$$

The covariance matrix for the "disturbance term" is known, so we can use GLS and compute

$$\begin{aligned} \widehat{\xi}_1 &= \left(\begin{bmatrix} \mathbf{I} & \mathbf{H} \end{bmatrix} \begin{bmatrix} \mathbf{Q} & 0 \\ 0 & \mathbf{R} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{I} \\ \mathbf{H}' \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{I} & \mathbf{H} \end{bmatrix} \begin{bmatrix} \mathbf{Q} & 0 \\ 0 & \mathbf{R} \end{bmatrix}^{-1} \begin{bmatrix} \widetilde{\xi}_1 \\ \mathbf{y} \end{bmatrix} = \\ &= \left[\mathbf{Q}^{-1} + \mathbf{H}\mathbf{R}^{-1}\mathbf{H}' \right]^{-1} \left[\mathbf{Q}^{-1} \widetilde{\xi}_1 + \mathbf{H}\mathbf{R}^{-1}\mathbf{y} \right] = \\ &= \left(\left[\mathbf{Q}^{-1} + \mathbf{H}\mathbf{R}^{-1}\mathbf{H}' \right]^{-1} \mathbf{Q}^{-1} \right) \widetilde{\xi}_1 + \left(\left[\mathbf{Q}^{-1} + \mathbf{H}\mathbf{R}^{-1}\mathbf{H}' \right]^{-1} \mathbf{H}\mathbf{R}^{-1} \right) \mathbf{y} \end{aligned} \tag{1}$$

(matrix-weighted average: nice)

Note: from GLS theory, we also know that

$$V(\widehat{\xi}_1) = \left(\mathbf{Q}^{-1} + \mathbf{H}\mathbf{R}^{-1}\mathbf{H}'\right)^{-1}.$$

Note: from GLS theory, we also know that

$$V(\widehat{\xi}_1) = \left(\mathbf{Q}^{-1} + \mathbf{H}\mathbf{R}^{-1}\mathbf{H}'\right)^{-1}.$$

Even nicer: try to forecast $\mathbf{y}$ on the basis of $\boldsymbol{\xi}_0$. Best choice is $\widehat{\mathbf{y}} = \mathbf{H}'\widetilde{\xi}_1$. The forecast error $\mathbf{e}$, satisfies $\mathbf{y} = \mathbf{H}'\widetilde{\xi}_1 + \mathbf{e}$. (Covariance matrix for $\mathbf{e}$ is also easy to compute; let's just call it $\boldsymbol{\Sigma}$. Useful later)

Note: from GLS theory, we also know that

$$V(\widehat{\xi_1}) = \left(\mathbf{Q}^{-1} + \mathbf{H}\mathbf{R}^{-1}\mathbf{H}'\right)^{-1}.$$

Even nicer: try to forecast $\mathbf{y}$ on the basis of ξ_0. Best choice is $\widehat{\mathbf{y}} = \mathbf{H}'\widetilde{\xi_1}$. The forecast error $\mathbf{e}$, satisfies $\mathbf{y} = \mathbf{H}'\widetilde{\xi_1} + \mathbf{e}$. (Covariance matrix for $\mathbf{e}$ is also easy to compute; let's just call it $\boldsymbol{\Sigma}$. Useful later)
Now substitute into (1)

$$\left(\left[\mathbf{Q}^{-1} + \mathbf{H}\mathbf{R}^{-1}\mathbf{H}'\right]^{-1}\mathbf{Q}^{-1}\right)\widetilde{\xi_1} + \left(\left[\mathbf{Q}^{-1} + \mathbf{H}\mathbf{R}^{-1}\mathbf{H}'\right]^{-1}\mathbf{H}\mathbf{R}^{-1}\right)\left(\mathbf{H}'\widetilde{\xi_1} + \mathbf{e}\right)$$

and simplify to get

$$\widehat{\xi_1} = \widetilde{\xi_1} + \mathbf{K}\mathbf{e} \tag{2}$$

The "wise-up" algorithm

We could wrap up all of the above into an algorithm whose steps are:

1. given ξ_0, form a first guess of $\xi_1 = \widetilde{\xi}_1 = \mathbf{F}\xi_0$
2. use $\widetilde{\xi}_1$ to predict $\mathbf{y}$ via $\widehat{\mathbf{y}} = \mathbf{H}'\widetilde{\xi}_1$
3. observe $\mathbf{y}$
4. compute the forecast error on $\mathbf{y}$, $\mathbf{e} = \mathbf{y} - \widehat{\mathbf{y}}$
5. use $\mathbf{e}$ to update our guess on ξ_1 via $\widetilde{\xi}_1 + \mathbf{K}\mathbf{e}$.

The "wise-up" algorithm

We could wrap up all of the above into an algorithm whose steps are:

1. given ξ_0, form a first guess of $\xi_1 = \widetilde{\xi}_1 = \mathbf{F}\xi_0$
2. use $\widetilde{\xi}_1$ to predict $\mathbf{y}$ via $\widehat{\mathbf{y}} = \mathbf{H}'\widetilde{\xi}_1$
3. observe $\mathbf{y}$
4. compute the forecast error on $\mathbf{y}$, $\mathbf{e} = \mathbf{y} - \widehat{\mathbf{y}}$
5. use $\mathbf{e}$ to update our guess on ξ_1 via $\widetilde{\xi}_1 + \mathbf{K}\mathbf{e}$.

Steps 1 and 2 don't require $\mathbf{y}$; if $\mathbf{y}$ is costly to acquire you can split the above as:

- forecast $\mathbf{y}$ (1–2)
- once $\mathbf{y}$ is observed, update your estimate of ξ_1 (3–5) via $\mathbf{e}$.

The "wise-up" algorithm

We could wrap up all of the above into an algorithm whose steps are:

1. given ξ_0, form a first guess of $\xi_1 = \widetilde{\xi}_1 = \mathbf{F}\xi_0$
2. use $\widetilde{\xi}_1$ to predict $\mathbf{y}$ via $\widehat{\mathbf{y}} = \mathbf{H}'\widetilde{\xi}_1$
3. observe $\mathbf{y}$
4. compute the forecast error on $\mathbf{y}$, $\mathbf{e} = \mathbf{y} - \widehat{\mathbf{y}}$
5. use $\mathbf{e}$ to update our guess on ξ_1 via $\widetilde{\xi}_1 + \mathbf{Ke}$.

Steps 1 and 2 don't require $\mathbf{y}$; if $\mathbf{y}$ is costly to acquire you can split the above as:

- forecast $\mathbf{y}$ (1–2)
- once $\mathbf{y}$ is observed, update your estimate of ξ_1 (3–5) via $\mathbf{e}$.

In fact, you can combine all the steps into one and go straight from ξ_0 to $\widehat{\xi}_1 = \mathbf{F}\xi_0 + \mathbf{Ke}$. $\mathbf{K}$ is called the "gain" matrix.

The "wise-up" algorithm

We could wrap up all of the above into an algorithm whose steps are:

1. given ξ_0, form a first guess of $\xi_1 = \widetilde{\xi}_1 = \mathbf{F}\xi_0$
2. use $\widetilde{\xi}_1$ to predict $\mathbf{y}$ via $\widehat{\mathbf{y}} = \mathbf{H}'\widetilde{\xi}_1$
3. observe $\mathbf{y}$
4. compute the forecast error on $\mathbf{y}$, $\mathbf{e} = \mathbf{y} - \widehat{\mathbf{y}}$
5. use $\mathbf{e}$ to update our guess on ξ_1 via $\widetilde{\xi}_1 + \mathbf{Ke}$.

Steps 1 and 2 don't require $\mathbf{y}$; if $\mathbf{y}$ is costly to acquire you can split the above as:

- forecast $\mathbf{y}$ (1–2)
- once $\mathbf{y}$ is observed, update your estimate of ξ_1 (3–5) via $\mathbf{e}$.

In fact, you can combine all the steps into one and go straight from ξ_0 to $\widehat{\xi}_1 = \mathbf{F}\xi_0 + \mathbf{Ke}$. $\mathbf{K}$ is called the "gain" matrix.

Let's call this the "wise-up" algorithm.

Generalisation

Now suppose we only observe ξ_0 imperfectly: all we have at the beginning is an unbiased estimate of ξ_0 (say, $\widehat{\xi_0}$) with a known covariance matrix (all would just be a limiting case with $V(\widehat{\xi_0}) = 0$).

Nothing would change, except for messier algebra.

This generalisation, however, paves the way to the possibility of iterating the algorithm above.
This is the intuition behind the celebrated *Kalman Filter*. But first, we need to define what a "state-space model" is.

A linear state-space model can be written as

$$\boldsymbol{\xi}_{t+1} = \mathbf{F}_t \boldsymbol{\xi}_t + \mathbf{v}_t \tag{3}$$

$$\mathbf{y}_t = \mathbf{A}_t' \mathbf{x}_t + \mathbf{H}_t' \boldsymbol{\xi}_t + \mathbf{w}_t \tag{4}$$

A linear state-space model can be written as

$$\boldsymbol{\xi}_{t+1} = \mathbf{F}_t \boldsymbol{\xi}_t + \mathbf{v}_t \tag{3}$$

$$\mathbf{y}_t = \mathbf{A}_t' \mathbf{x}_t + \mathbf{H}_t' \boldsymbol{\xi}_t + \mathbf{w}_t \tag{4}$$

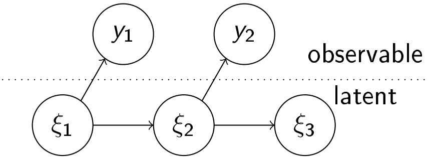


Figure: State space model through time

Usually, the $(r \times 1)$ vector $\mathbf{v}_t$ and the $(n \times 1)$ vector $\mathbf{w}_t$ are assumed to be vector white noise.

Let's start with something simple:

$$y_t = \varphi y_{t-1} + \varepsilon_t + \theta \varepsilon_{t-1};$$

now invert $A(L)$ to give $y_t = C(L)a_t$, where $a_t = A(L)^{-1}\varepsilon_t$, or

$$a_t = \varphi a_{t-1} + \varepsilon_t \tag{5}$$

Therefore,

$$y_t = a_t + \theta a_{t-1}. \tag{6}$$

It's not difficult to see that

- the only thing we observe is the scalar y_t; therefore, $\mathbf{y}_t = y_t$.
- To express y_t, you need *both* a_t and its lag; therefore, $\boldsymbol{\xi}_t$ must contain at least a_t and a_{t-1}. Using (6),

$$y_t = [1 \quad \theta] \begin{bmatrix} a_t \\ a_{t-1} \end{bmatrix}$$

- $\boldsymbol{\xi}_{t+1}$ can be written as a function of itself lagged, and a white-noise process:

$$\begin{bmatrix} a_{t+1} \\ a_t \end{bmatrix} = \begin{bmatrix} \varphi & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a_t \\ a_{t-1} \end{bmatrix} + \begin{bmatrix} \varepsilon_{t+1} \\ 0 \end{bmatrix}$$

So the ARMA(1,1) model can be written in state-space form this way:

$$\mathbf{H} = \begin{bmatrix} 1 \\ \theta \end{bmatrix} \qquad \mathbf{F} = \begin{bmatrix} \varphi & 0 \\ 1 & 0 \end{bmatrix} \qquad \mathbf{Q} = \sigma^2 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

Not difficult to generalise to ARMAs of any order.
State-space representations are, in general, not unique. For example, there are other ways to express an ARMA(1,1):

$$\mathbf{H} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \mathbf{F} = \begin{bmatrix} \varphi & \theta \\ 0 & 0 \end{bmatrix} \qquad \mathbf{Q} = \sigma^2 \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},$$

or

$$\mathbf{H} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \mathbf{F} = \begin{bmatrix} \varphi & 0 \\ 1 & 0 \end{bmatrix} \qquad \mathbf{Q} = \sigma^2 \begin{bmatrix} 1 & \theta \\ \theta & \theta^2 \end{bmatrix}.$$

The best one to use depends on several factors, not least computational efficiency.

The Kalman recursions

Essentially, a clever way to apply the “wise-up” algorithm to state-space models.

The Kalman recursions

Essentially, a clever way to apply the "wise-up" algorithm to state-space models.

In practice, we perform two "passes":

1. the "forward" pass (which is the Kalman filter proper),
2. the "backwards" pass, also known as "smoothing".

The Kalman recursions

Essentially, a clever way to apply the "wise-up" algorithm to state-space models.

In practice, we perform two "passes":

1. the "forward" pass (which is the Kalman filter proper),
2. the "backwards" pass, also known as "smoothing".

The forward pass is mostly used in estimating the parameters of the models.

The Kalman recursions

Essentially, a clever way to apply the "wise-up" algorithm to state-space models.

In practice, we perform two "passes":

1. the "forward" pass (which is the Kalman filter proper),
2. the "backwards" pass, also known as "smoothing".

The forward pass is mostly used in estimating the parameters of the models.

Smoothing is used, instead, to reconstruct the historical time path of the state vector.

We start from a preliminary estimate of the state at time 0: a random variable with known mean and variance:

$$\boldsymbol{\xi}_0 \sim R(\tilde{\boldsymbol{\xi}}_0, \mathbf{P}_0)$$

We start from a preliminary estimate of the state at time 0: a random variable with known mean and variance:

$$\xi_0 \sim R(\tilde{\xi}_0, \mathbf{P}_0)$$

We may (a) guess the value of $\mathbf{y}_0$ via equation (4) and of ξ_1 via equation (3) (b) when $\mathbf{y}_1$ becomes available, update our guess on ξ_1 wia the "wise-up" algorithm.

We start from a preliminary estimate of the state at time 0: a random variable with known mean and variance:

$$\xi_0 \sim R(\tilde{\xi}_0, \mathbf{P}_0)$$

We may (a) guess the value of $\mathbf{y}_0$ via equation (4) and of ξ_1 via equation (3) (b) when $\mathbf{y}_1$ becomes available, update our guess on ξ_1 wia the "wise-up" algorithm.
So we have

1. a forecast error $\mathbf{e}_1$ with its own covariance matrix $\mathbf{\Sigma}_1$;
2. a new starting point $\tilde{\xi}_1$ with its own covariance matrix $\mathbf{P}_1$;

which means that we can repeat the algorithm from ξ_1, so to obtain $\mathbf{e}_2$ and $\mathbf{\Sigma}_2$, and then on and on, until our data end, and we stop at ξ_T, $\mathbf{e}_T$ and $\mathbf{\Sigma}_T$.

The formulae for performing the above are generalisations of (2) and (3) that can be found in specialised books.

The formulae for performing the above are generalisations of (2) and (3) that can be found in specialised books.
They're messier because:

1. The state matrices can be time-varying;
2. the disturbances $\mathbf{w}_t$ and $\mathbf{v}_t$ may not be uncorrelated, and
3. initialisation for the algorithm may not be obvious: notably, sometimes you may want to endow ξ_0 with an "infinite" covariance matrix (this case is known as the "diffuse prior" case).

One last thing on the filtering pass: if the parameters of the system are the true ones, filtering gives you the best guess of the state vector $\boldsymbol{\xi}_t$, given the information up to time $t-1$:

$$\widehat{\boldsymbol{\xi}}_t = E(\boldsymbol{\xi}_t | \Im_{t-1})$$

One last thing on the filtering pass: if the parameters of the system are the true ones, filtering gives you the best guess of the state vector $\boldsymbol{\xi}_t$, given the information up to time $t-1$:

$$\widehat{\boldsymbol{\xi}}_t = E(\boldsymbol{\xi}_t | \Im_{t-1})$$

Therefore, the one-step-ahead prediction errors give you

$$\mathbf{e}_t = \mathbf{y}_t - E(\mathbf{y}_t | \Im_{t-1})$$

and by construction $E(\mathbf{e}_t | \Im_{t-1}) = 0$ (martingale difference sequence).

One last thing on the filtering pass: if the parameters of the system are the true ones, filtering gives you the best guess of the state vector $\boldsymbol{\xi}_t$, given the information up to time $t-1$:

$$\widehat{\boldsymbol{\xi}_t} = E(\boldsymbol{\xi}_t | \Im_{t-1})$$

Therefore, the one-step-ahead prediction errors give you

$$\mathbf{e}_t = \mathbf{y}_t - E(\mathbf{y}_t | \Im_{t-1})$$

and by construction $E(\mathbf{e}_t | \Im_{t-1}) = 0$ (martingale difference sequence).

Therefore, you can calculate the Gaussian log-density for each of them:

$$\ell_t = \text{const} \; - \frac{1}{2} \ln |\boldsymbol{\Sigma}_t| - \frac{1}{2} \mathbf{e}_t' \boldsymbol{\Sigma}_t^{-1} \mathbf{e}_t \tag{7}$$

and $\sum_t \ell_t$ may be interpreted as a log-likelihood. This is the main idea behind the usage of the Kalman Filter as an inferential tool.

Once you have the log-likelihood for a given set of system matrices, you can either

- maximize it (as a rule, numerically) and go for classical (frequentist) inference methios
- go Bayesian, use it to update your priors and get your posteriors, which is what DSGE people do.

Smoothing

Basic idea:
There are several smoothing techniques. What people usually do in econom(etr)ics is *fixed-interval* smoothing.

Smoothing

Basic idea:

There are several smoothing techniques. What people usually do in econom(etr)ics is *fixed-interval* smoothing.

Basically, you use the "wise-up" algorithm backwards (algebra is complicated, though) and start from $\hat{\boldsymbol{\xi}}_T$ and use $\mathbf{e}_{T-1}, \mathbf{e}_{T-2}, \ldots$ to work your way backwards to $\boldsymbol{\xi}_0$.

Smoothing

Basic idea:
There are several smoothing techniques. What people usually do in econom(etr)ics is *fixed-interval* smoothing.
Basically, you use the "wise-up" algorithm backwards (algebra is complicated, though) and start from $\hat{\xi}_T$ and use $\mathbf{e}_{T-1}, \mathbf{e}_{T-2}, \ldots$ to work your way backwards to ξ_0.
Again, see the books for details.

Random walk plus noise

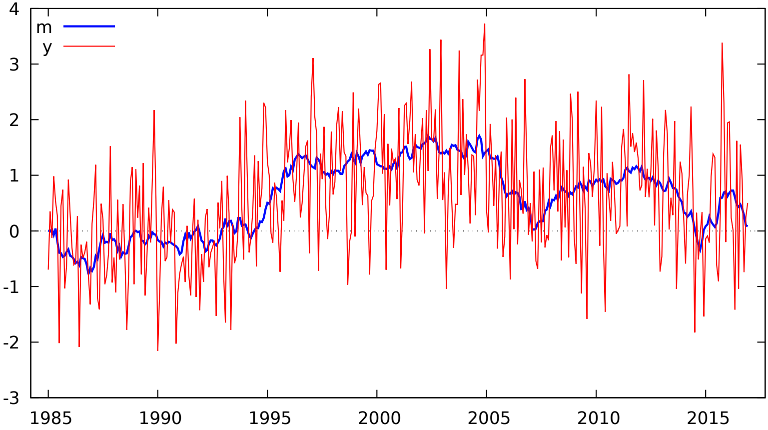
Let us generate a "random walk plus noise" artificial process

$$\begin{aligned} m_t &= m_{t-1} + u_t \\ y_t &= m_t + \varepsilon_t \end{aligned}$$

where $V(u_t) = 0.01$ and $V(\varepsilon_t) = 1$; gretl code follows

```
nulldata 384
set seed 71218
setobs 12 1985:1

series m = cum(normal()*0.1)
series y = m + normal()
```

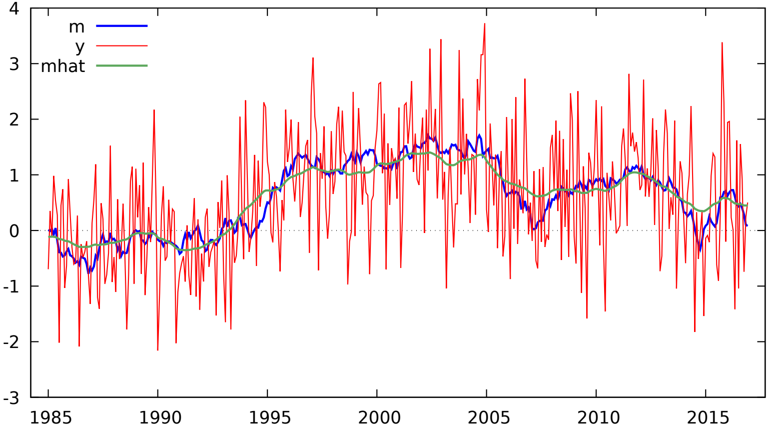


Note that in practice, the only observable series would be y_t (the red line). What we want to do is use the Kalman filter to recover m_t.

```
matrix F = 1
matrix H = 1
matrix Q = 0.01
mod = ksetup(y, F, H, Q)
mod.diffuse = 1
mod.obsvar = 1

err = ksmooth(&mod)
series mhat = mod.state
```

produces


m
y
mhat
4
3
2
1
0
-1
-2
-3
1985
1990
1995
2000
2005
2010
2015

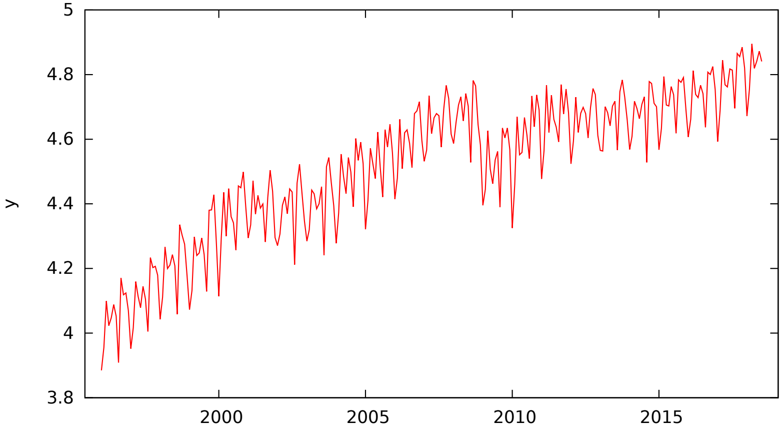
Structural Time Series Models

Invented by the English statistician Andrew Harvey in the 1980s.
His approach resembles what guitarists do with pedal effects: in order to get the features they want, they stack effects on top of one another, until the sound is right.

Model name	Formula
Random walk plus noise	$\mu_t = \mu_{t-1} + \varepsilon_t$ $y_t = \mu_t + u_t$
Local linear trend	$\mu_t = \mu_{t-1} + \beta_{t-1} + \varepsilon_t$ $\beta_t = \beta_{t-1} + \eta_t$ $y_t = \mu_t + u_t$
Trigonometric Cycle	$a_t = \rho\left(\cos\theta \cdot a_{t-1} + \sin\theta \cdot b_{t-1}\right) + \omega_{1,t}$ $b_t = \rho\left(-\sin\theta \cdot a_{t-1} + \cos\theta \cdot b_{t-1}\right) + \omega_{2,t}$ $y_t = a_t$
$\vdots$	$\vdots$

Example: Austrian IP

```
open dbnomics
data IMF/IFS/M.AT.AIPMA_IX --name=Austria_IP
series y = log(Austria_IP)
include StrucTiSM.gfn
mod = STSM_GUImeta(y, 3, null, 0, 1, 1, 0, 1)
```


5
4.8
4.6
4.4
4.2
4
3.8
y
2000
2005
2010
2015

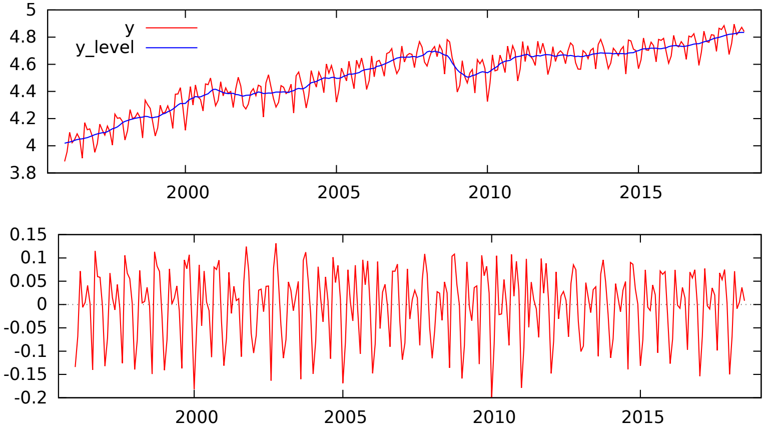
```
Structural model for y, 1996:01 - 2018:07 (T = 271)

                     coefficient   std. error      z       p-value
  ---------------------------------------------------------------
  Irregular          0.0123792     0.00895181    1.383    0.1667
  Trend              0.0131580     0.00144715    9.092    9.69e-20 ***
  Slope              0.000181810   0.000215125   0.8451   0.3980
  Seasonal (dums)    0.0196931     0.00417469    4.717    2.39e-06 ***

Average log-likelihood = 1.51121

Specification:

Stochastic trend, stochastic slope, dummy seasonals (stoch.),
irregular component
```


5
4.8
4.6
4.4
4.2
4
3.8
y
y_level
2000
2005
2010
2015
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
2000
2005
2010
2015