# Structural time series grammar over variable blocks


### David Rushing Dewhurst


Consider structural time series models that decompose additively. How can we extend these models to make them more expressive while still maintaining interpretability?

$$y(t) = \varepsilon(t;\sigma) + \sum_k f_k(t;\theta_k)$$

$$y(t) = \varepsilon(t;\sigma) + \sum_k f_k(t;\theta_k) + f_{k'}(t;\theta_{k'})$$

We could add another "building block" term...

...or we could replace static parameters with further time varying components.

$$y(t) = \varepsilon(t;\sigma) + \sum_k f_k(t; g_k(t;\theta_k))$$

$$\varepsilon(t;\sigma) \sim \mathrm{Normal}(0, \sigma^2)$$

charles river analytics

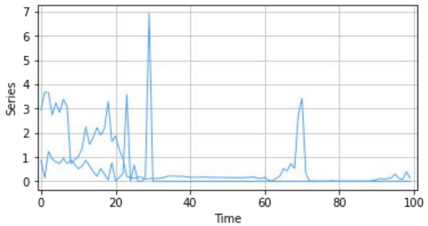
## Implementation

**Modeling**

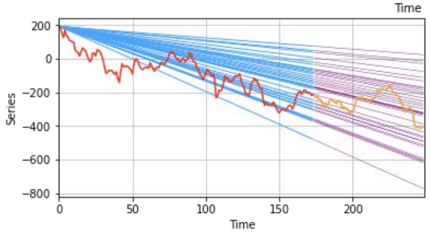
Small library of generative blocks that can be combined into valid sentences of the language generated by grammar ***G*** (with or without changepoint operator -- with changepoint operator these are not causal models)

```
log_vol_1 = sts.AR1(t1=t1, ...)
log_vol_2 = sts.GlobalTrend(t1=t1, ...).cos()
vol = sts.changepoint(log_vol_1.exp(), log_vol_2.exp(), frac=0.6)
price = sts.RandomWalk(t1=t1, loc=0.0, scale=vol, ...).exp()
sample = price()
```

*E.g., stochastic volatility model with changepoint + nontrivial latent structure*




```
with stsb2.effects.ProposalEffect(trend):
    trend.parameter_update(
        a=posterior[trend]['a'],
        b=posterior[trend]['b']
    )
    trend_posterior = trend()
    with stsb2.effects.ForecastEffect(...):
        trend_forecast = trend()
```



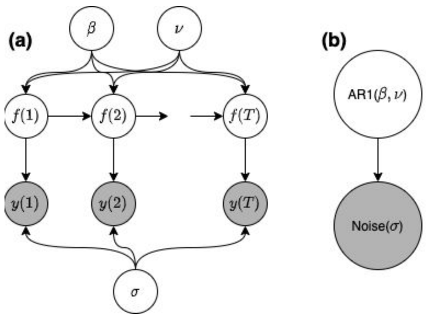

**Inference**

W.I.P. (only proof of concept LF rejection sampling), includes proposal, intervention, and forecast effect handlers for converting *sample(...)* statements into proposal and forecast distributions

Library: https://gitlab.com/daviddewhurst/stsb2; Documentation: https://davidrushingdewhurst.com/stsb2/docs/

# Block structure and grammar

Instead of defining a single sample node for each time slice of a process, define a single sample node that describes both the process that generates the STS component *and* the vector of parameters used in the data-generating process.




This shifts complexity from edge to node space and corresponds to a particular factorization of the model joint likelihood

Define grammar ***G*** over these "blocks" *f* that take parameters *θ* or other blocks as arguments

$$S \to Q \mid Q + S$$
$$Q \to f(p) \mid C(S, f(p)) \mid C(f(p), S)$$
$$p \to \theta \mid (S \mid p, ..., S \mid p)$$

*C(. , .)* is the changepoint operator and inserts a changepoint at a random time by concatenating two models represented by two unique sentences

## Implementation + W.I.P

Non-Markov DGP expressed in single block (same as simple first order Markov model
No differentiability assumption
Objects are stochastic -- can sample from whole STS or from component parts
Explicitly model decomposition is immediately interpretable (compare with GP kernel grammar)

Next steps: implement DSL + compiler to facilitate a) easier model expression and b) model search algorithms (searching for optimal string in language generated by grammar ***G*** subject to some constraints)

```
init series;  # can have multiple calls or single per init
init trend, seasonal, noise;

result prior_pred_samples, posterior_samples, posterior_pred_samples;

set t1 = 100;
set beta = 0.5;
set scale = 0.5;

define trend = GlobalTrend(t1=t1);
define seasonal = GlobalTrend(t1=t1).cos();
define noise = AR1(t1=t1, beta=beta, scale=scale);

# define the model -- this is a seasonal global trend model
define series = trend + seasonal + noise;

# sample from prior
sample series -> prior_predictive -> prior_pred_samples;

# load some data and do inference
file input "./data/some_data.csv";
assign input -> series;
sample series -> posterior -> posterior_samples;
sample series -> posterior_predictive -> posterior_pred_samples;
```

*Possible DSL syntax -- sample operation generates prior / posterior / posterior predictive samples depending on context variable (`prior`, `posterior`, `posterior_predictive`)*

Library: https://gitlab.com/daviddewhurst/stsb2; Documentation: https://davidrushingdewhurst.com/stsb2/docs/