Regime Switching and Technical Trading with Dynamic Bayesian Networks in High-Frequency Stock Markets

Luis Damiano, Brian Peterson, Michael Weylandt

2017-08-21

Contents

This work aims at ... .

The authors acknowledge Google for financial support via the Google Summer of Code 2017 program.

---

1 Hierarchical Hidden Markov Models

The Hierarchical Hidden Markov Model (HHMM) is a recursive hierarchical generalization of the HMM that provides a systematic unsupervised approach for complex multi-scale structure. The model is motivated by the multiplicity of length scales and the different stochastic levels (recursive nature) present in some sequences. Additionally, it infers correlated observations over long periods via higher levels of hierarchy.

The model structure is fairly general and allows an arbitrary number of activations of its submodels. The multi-resolution structure is handled by temporal experts[1] of different time scales.

1.1 Model specification

HHMM are structured multi-level stochastic processes that generalize HHM by making each of the hidden states an autonomous probabilistic model. There are two kinds of states: internal states are HHMM that

[1] In Machine Learning terminology, a problem is divided into homogeneous regions addressed by an expert submodel. A gating network or function decides which expert to use for each input region.

emit sequences by a recursive activation of one of the substates, while production states generate an output symbol according to the probability distribution of the set of output symbols.

Hidden dynamics are lead by transitions. Vertical transitions involve the activation of a substate by an internal state, they may include further vertical transitions to lower level states. Once completed, they return the control to the state that originated the recursive activation chain. Then, a horizontal transition is performed. Its state transition within the same level.

A HHMM can be represented as a standard single level HMM whose states are the production states of the corresponding HHMM with a fully connected structure, i.e. there is a non-zero probability of tranisition from any state to any other state. This equivalent new model lacks the multi-level structure.

Let $z_t^d = i$ be the state of an HHMM at the step t, where $i \in \{1, \ldots, |z^d|\}$ is the state index, $|z^d|$ is the number of possible steps within the d-th level and $d \in \{1, \ldots, D\}$ is the hierarchy index taking values $d = 1$ for the root state, $d = \{2, \ldots, ..., D-1\}$ for the remaining internal states and $d = D$ for the production states.

In addition to its structure, the model is characterized by the state transition probability between the internal states and the output distribution of the production states. For each internal state z_t^d for $d \in \{1, \ldots, D-1\}$, there is a state transition probability matrix $\boldsymbol{A}^d$ with elements $A_{ij}^d = p(z_t^{d+1} = j | z_t^{d+1} = j)$ is the probability of a horizontal transition from the i-th state to the j-th state within the level d. Similarly, there is the initial distribution vector over the substates $\boldsymbol{\pi}^d$ with elements $\pi_j^d = p(z_t^{d+1} = j | z_t^d)$ for $d \in \{1, \ldots, D-1\}$. Finally, each production state z_t^D is parametrized by the output parameter vector $\boldsymbol{\theta}_o^i$ whose form depends on the specification of the observation model $p(\boldsymbol{x}_t | z_t^D = j, \boldsymbol{\theta}_o^j)$ corresponding to the j-th production state.

1.2 Generative model

The root node initiates a stochastic sequence generation. An observation for the first step in the sequence t is generated by drawing at random one of the possible substates according to the initial state distribution $\boldsymbol{\pi}^1$. To replicate the recursive activation process, for each internal state entered z_t^d one of the substates is randomly chosen according to the corresponding initial probability vector $\boldsymbol{\pi}^d$. When an internal state transitions to a production state $z_t^D = j$, a single observation is generated according to the state output parameter vector $\boldsymbol{\theta}_o^j$. Control returns to the internal state that lead to the current production state z_t^{D-1}, which in turns selects the next state in the same level according to transition matrix $\boldsymbol{A}^{D-1}$.

Save for the top, each level $d \in \{2, \ldots, D\}$ has a final state that terminates the stochastic state activation process and returns the control to the parent state of the whole hierarchy. The generation of the observation sequence is completed when control of all the recursive activations returns to the root state.

1.3 Parameter estimation

The parameters of the models are $\boldsymbol{\theta} = \left\{ \left\{ \boldsymbol{A}^d \right\}_{d \in \{1,\ldots,D-1\}}, \left\{ \boldsymbol{\pi}^d \right\}_{d \in \{1,\ldots,D-1\}}, \{\boldsymbol{\theta}_o\} \right\}$. The form of $\boldsymbol{\theta}_o$ depends on the specification of the production states. We refer the read to **fine1998hierarchical** for a detailed treatment of estimation and inference procedures.

---

2 Regime Switching and Technical Trading with Dynamic Bayesian Networks in High-Frequency Stock Markets

2.1 Preamble

...

2.2 Feature extraction

2.2.1 Input series

Tick series are a sequence of triples $\{y_k\}$ with $y_k = (t_k, p_k, v_k)$, where $t_k \leq t_{k+1}$ is the time stamp in seconds, p_k is the trade price and v_k is the trade volume. The sequence is ordered by the occurence of trades. There can be more than one trade within a second.

Following **tayal2009regime**, who in turns drew inspiration from the technical analysis techniques proposed by **ord2008secret**, we derive a zig-zag sequence that captures the bid-ask bounce $\{z_k\}$ with $z_k = (i_n, j_n, e_n, \phi_n)$, where $i_n \leq i_j$ are indices to the tick series representing the starting and ending point of the extrema, $e_n = p_k \; \forall \; k : i_n \leq k \leq j_n$ is the price at the local extrema, and ϕ_m measures the average volume per second during the zig-zag leg ending at e_n:

$$\phi_n = \frac{1}{t_{j_n} - t_{i_{n-1}} + 1} \sum_{k=i_{n-1}}^{j_n} v_k.$$

We note that $p_{i_n} < e_n < p_{j_n+1}$ for local maxima and $p_{i_n} > e_n > p_{j_n+1}$ for local minima. The average volume, which includes the end-point extrema, is normalized by $t_{j_n} - t_{i_{n-1}} + 1$ to avoid division by zero when the zig-zag leg occurs within the same time period. Most importantly, we underline that the n-th zig-zag point z_n is realized only after observing the $(j_n + 1)$-th tick point y_{j_n+1}. Failing to consider the one tick lag between leg completion and the time of detection would cause look-ahead bias in the out of sample forecasts.

2.2.2 Processing rules

Discrete features are created based on the zig-zag series $\{z_n\}$. We first create an auxiliary series $\{O_n\}$ with $O_n = (f_n^0, f_n^1, f_n^2)$, where f_n^0 represents the direction of the zig-zag, f_n^1 indicates the existence of a trend and f_n^2 indicates whether average volume increased or decreased.

Formally,

$$f_n^0 = \begin{cases} +1 & \text{if } e_n \text{ is a local maximum (positive zig-zag leg)} \\ -1 & \text{if } e_n \text{ is a local minimum (negative zig-zag leg),} \end{cases}$$

and

$$f_n^1 = \begin{cases} +1 & \text{if } e_{n-4} < e_{n-2} < e_n \wedge e_{n-3} < e_{n-1} \text{ (up-trend)} \\ -1 & \text{if } e_{n-4} > e_{n-2} > e_n \wedge e_{n-3} > e_{n-1} \text{ (down-trend)} \\ 0 & \text{otherwise (no trend).} \end{cases}$$

For the third indicator function, we compute the average volume ratios,

$$\nu_n^1 = \frac{\phi_n}{\phi_{n-1}}, \quad \nu_n^2 = \frac{\phi_n}{\phi_{n-2}}, \quad \nu_n^1 = \frac{\phi_{n-1}}{\phi_{n-2}},$$

we transform the ratios into a discrete variable using an arbitrary threshold α,

$$\tilde{\nu}_n^j = \begin{cases} +1 & \text{if } \nu_n^j - 1 > \alpha \\ -1 & \text{if } 1 - \nu_n^j > \alpha \\ 0 & \text{if } |\nu_n^j - 1| \leq \alpha, \end{cases}$$

and we finallly define

$$f_n^2 = \begin{cases} +1 & \text{if } \tilde{\nu}_n^1 = 1, \tilde{\nu}_n^2 > -1, \tilde{\nu}_n^3 < 1 \text{ (volume strengthens)} \\ -1 & \text{if } \tilde{\nu}_n^1 = -1, \tilde{\nu}_n^2 < -1, \tilde{\nu}_n^3 > -1 \text{ (volume weakens)} \\ 0 & \text{otherwise (volume is indeterminant).} \end{cases}$$

The features or legs $\boldsymbol{D} = \{D_1, \ldots, D_9\}$, $\boldsymbol{U} = \{U_1, \ldots, U_9\}$ are then created using the following table.

Feature	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8	U_9
Zig-zag direction	Upwards+1	Upwards+1	Upwards+1	Upwards+1	Upwards+1	Upwards+1	Upwards+1	Upwards+1	Upwards+1
Price trend	Up-trend+1	Down-trend-1	Up-trend+1	No trend0	No trend0	No trend0	Down-trend-1	Up-trend+1	Down-trend-1
Change in volume	Strengthens+1	Strengthens+1	Indeterminate0	Strengthens+1	Indeterminate0	Weakens-1	Indeterminate0	Weakens-1	Weakens-1
Market state	Bullish	Bullish	Bullish	Bullish	Local volatility	Bearish	Bearish	Bearish	Bearish

Table 2: Summary of feature space.

Feature	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9
Zig-zag direction	Downwards-1	Downwards-1	Downwards-1	Downwards-1	Downwards-1	Downwards-1	Downwards-1	Downwards-1	Downwards-1
Price trend	Up-trend+1	Down-trend-1	Up-trend+1	No trend0	No trend0	No trend0	Down-trend-1	Up-trend+1	Down-trend-1
Change in volume	Weakens-1	Weakens-1	Indeterminate0	Weakens-1	Indeterminate0	Strengthens+1	Indeterminate0	Strengthens+1	Strengthens+1
Market state	Bullish	Bullish	Bullish	Bullish	Local volatility	Bearish	Bearish	Bearish	Bearish

Plot here showing some example

Defining appropiate rules that capture trade volume dynamics and identify trends in volume despite high-frequency noise is the most challenging aspect of this design. **wisebourt2011hierarchical** proposes a modification for the feature extraction procedure. By computing the spread between the Volume Weighted Average Prices of the bid and the ask, he designs a book imbalance metric that describes the state of the order book at any given moment in time. In turns, **sandoval2015computational** applies wavelets over two simple-smoothed exponential distance-weighted average volume series to measure trade volume concentration in both sides of the book.

2.3 Model

We adhere to the methodology proposed in the original work as much as possible. We set up a HHMM to learn the sequence of discrete features extracted from a high-frequency time series of stock prices and traded volume. The figure below summarises the model structure in the form of a Dynamic Bayesian Network.

The graph starts with a root node z^0 that has two top-level children z_1^1 and z_2^1 representing bullish markets (or runs) and bearish markets (or reversals). The specifications do not pose any constraints to determine beforehand which node takes each of the possible two meanings. In consequence, latent states need to be

labeled after the learning stage based on sample characteristics such as mean returns. Althought the original author does not mention this possibility, prior information, like parameter ordering, could be embedded to break symmetry and mitigate eventual identification issues.

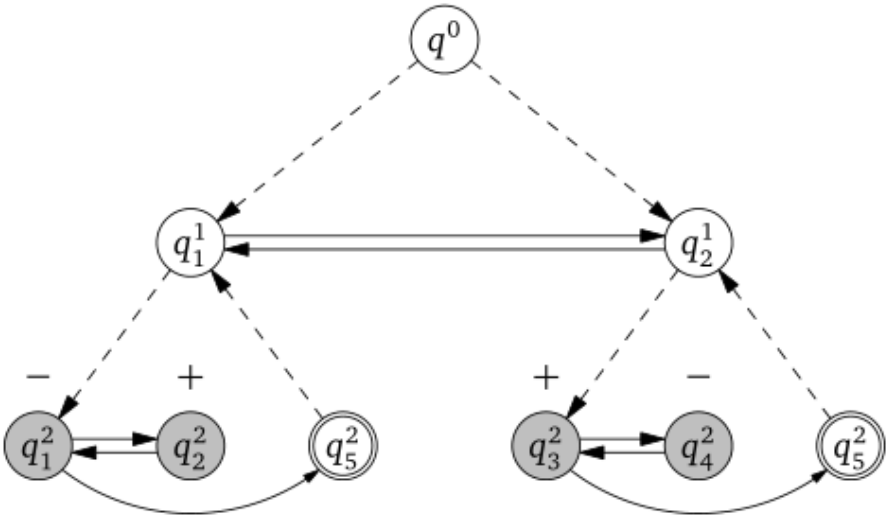


Figure 4.6: Hierachical hidden Markov model for price and volume analysis. q_1^1 and q_2^1 are top level states representing runs or reversals. q_1^2 and q_4^2 represent negative zig-zag legs, while q_2^2 and q_3^2 represent positive zig-zag legs. These are production nodes, filled in gray, that emit an observation symbol according to some distribution. Transitions enforce the alternating sequence of positive and negative legs. q_5^2 is the termination nodes (note, there are two of them) at which point control is returned back to the parent node in layer 1.

Figure 1: Graph from Tayal (2009). Will make my own figure when time comes. q_1^1 is equivalent to z_1^1 in our notation.

Each top node activates a probabilistic HMM with two latent states for negative and positive zig-zag legs. The sub-model activated by the node z_1^1 always starts with the node z_1^2 producing an observation from the distribution of negative zig-zag $\boldsymbol{D} = \{D_1, \ldots, D_9\}$. Next, it transitions to (a) the node z_2^2 producing a positive zig-zag leg $\boldsymbol{U} = \{U_1, \ldots, U_9\}$, (b) the end node which may, in turn, transition back to z_1^1 emitting a new negative zig-zag leg, or (c) the end node and switches to the second sub-model, landing on node z_3^2 and producing a positive zig-zag leg. Restricting transitions to this limited set of movements force alternation between positive and negative zig-zag legs, thus guaranteeing that all possible observation sequences are well behaved. The sub-model belonging to z_2^1 has similar but symmetrical behaviour. These two inner models are conditionally independent, an advantage we will take for computational time.

Theoretically, all HHMM can be expressed as an equivalent HMM with possibly sparse initial probability vector and transition matrix. Although learning from this representation may prove less efficient in terms of computational complexity, the relatively simple structure and many restrictions of the model under study make estimation and inference feasible. The equivalent "expanded" HMM has the following initial probability vector

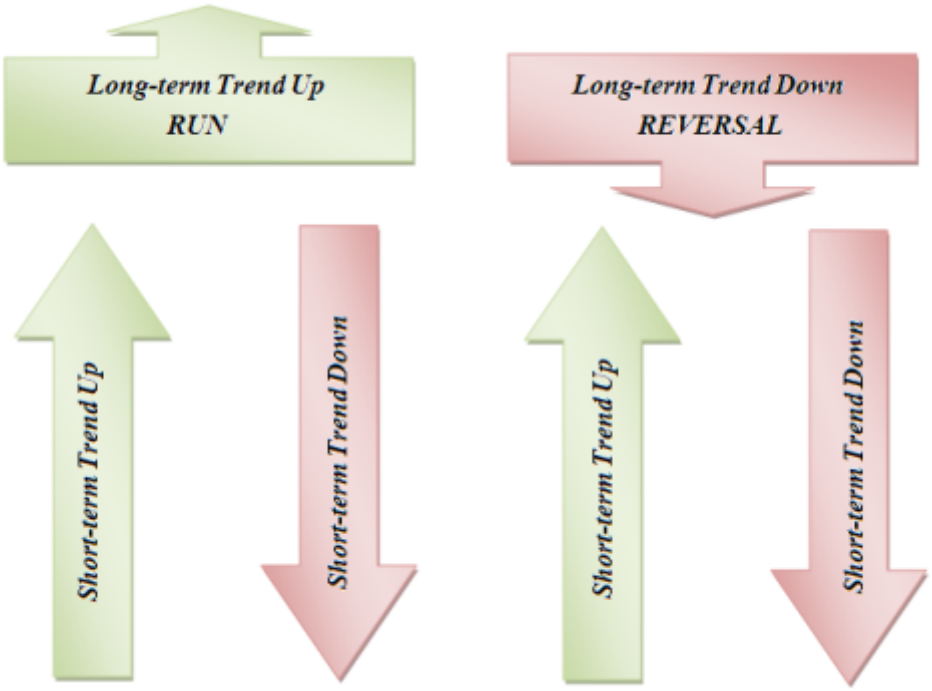


Figure 2: Graph from Wisebourt (2011). Will make my own figure when time comes. Explain why that positive zig-zags are short-term trend up, while top nodes are long term, etc.

$$\boldsymbol{\pi} = \begin{bmatrix} \pi_1 & 0 & 1-\pi_1 & 0 \end{bmatrix}$$

and transition matrix

$$\boldsymbol{A} = \begin{bmatrix} p_{11} & p_{12} & 1-p_{11}-p_{12} & 0 \\ 1 & 0 & 0 & 0 \\ p_{31} & 0 & p_{33} & 1-p_{31}-p_{33} \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

where each element p_{ij} represents the probability of transitioning from the hidden state i (row) to j (column) in one step. The matrix is sparse with zeros representing nodes with no direct connections. Since the initial probability vector and the rows of the transition matrix must sum to 1, hidden dynamics are governed by only five free parameters.

Does this representation "lose" the hierarchical interpretation of the states? Do transition probabilities summarize all possible paths from one node to the other? In this case I believe there's a 1-to-1 map from the HHMM arrows to the HMM matrix.

Production nodes emit one observation from the finite sets of possible outputs $\boldsymbol{O}_D$ (for z_1^2 and z_4^2) and $\boldsymbol{O}_U$ (for z_2^2 and z_3^2). Conditional probability distributions are given by output probability vectors of length L_U and L_D subject to sum-to-one constraints:

$$\boldsymbol{B}^1 = p(\boldsymbol{x}_t|z_1^2) = \left[b_{D_1}^1, \ldots, b_{D_9}^1\right], \boldsymbol{B}^2 = p(\boldsymbol{x}_t|z_2^2) = \left[b_{U_1}^2, \ldots, b_{U_9}^2\right], \boldsymbol{B}^3 = p(\boldsymbol{x}_t|z_3^2) = \left[b_{U_1}^3, \ldots, b_{U_9}^3\right], \boldsymbol{B}^4 = p(\boldsymbol{x}_t|z_4^2) = \left[b_{D_1}^4, \ldots,\right.$$

The observation model has 32 free parameters.

On the whole, the parameter vector for the HMM representation reduces to

$$\boldsymbol{\theta} = (\pi_1, p_{11}, p_{12}, p_{31}, p_{33}, b^1_{D_l}, b^2_{U_l}, b^3_{U_l}, b^4_{D_l}),$$

with $l \in \{1, \ldots, L-1\}$.

2.4 The sampler

...

2.5 Dataset

The original work presents results for both simulated and real data. The latter is based on historical high-frequency time series for the 60 stocks listed in the S&P/TSE60 index. The dataset consists of all 22 business days of May 2007. The author excludes three days due to significant errors without disclosing the exact dates. We confirm that our results are consistent with the original work by focusing on GoldCorp Inc (TSE:G), the only series described exhaustively in the original work.

2.6 Feature extraction

Note that we do not model prices directly. Instead, non-linear transforms are applied on the price and traded volume high-frequency series to produce the sequence of observations fed to the proposed model.

...

2.7 Methodology

Model parameters are estimated on a rolling window with five days each. Since top nodes are symmetrical, states are labeled ex-post based on the order of the in-sample mean of the percentage change in the initial and final price before the top level state switch. The state with larger and smaller returns are marked as bullish (a run) and bearish (a reversal) respectively.

After learning and labelling, two out of sample inference procedures are run on the sixth day. First, offline smoothing infers the hidden state at time t based on the full evidence of the sixth day. Although this quantity is not useful for trading because of its look-ahead bias, it provides an upper bound benchmark for the model. Second, online filtering is used as a trading rule.

Most of the diagnostics are based on trade returns. For the l-th top-level state switch, the percentage return is defined as

$$R_l = \frac{p^e_l - p^s_l}{p^s_l},$$

where p^s_l and p^e_l are the price at the start and end of the switch.

The information content of learning the top-level state is assessed by comparing the unconditional empirical distribution of trade returns versus their empirical distribution conditioned on the top-level state. Additionaly, regime return characteristics are validated: mean trade returns are expected to be higher for bullish regimes compared to bearish regimes, and they are expected to be positive and negative for runs and reversals

respectively. In the original work, most of these analysis are run both in-sample and out-of-sample. We focus on out-of-sample results only.

Finally, a trading strategy is tested. After a zig-zag leg is completed, the trading system buys one unit every time the top-level state switches to bullish (a run) and sells one unit every time it switches to bearish (a reveral). Trades are executed one tick after the zig-zag leg is observed to ensure there is no lock-ahead bias.

2.8 GoldCorp Inc (TSE:G)

We present an in-depth study of one stock to assess the strengths and weaknesses of the model. Using the tick-by-tick series of GoldCorp Inc (TSE:G), we split our dataset in training (2007-05-01 to 2007-05-07 - five trading days and ticks) and test (2007-05-08 - ticks) sets. Next, we run our procedure in a walking forward fashion for this stock as well as others and present some summary statistics for the performance of the strategy.

2.8.1 Data exploration

```
##    user  system elapsed
##    9.58    0.14   10.87
```

We center our attention on the sequence of trades, disregarding valuable information from the bid and ask series. Future research may employ such information to improve model predictability. We start by extracting the features using the procedure detailed above. We set the threshold for the change in volume indicator variable in $\alpha = 0.25$ as suggested by the author of the original work. In-sample dataset reduced from trade records to 8612 zig-zags.

Apart from the zig-zags themselves, local extrema are interesting on their own. Although we could not gain insight by visually inspecting other intermediate indicators such as the trend f_1, further research may find value in them.

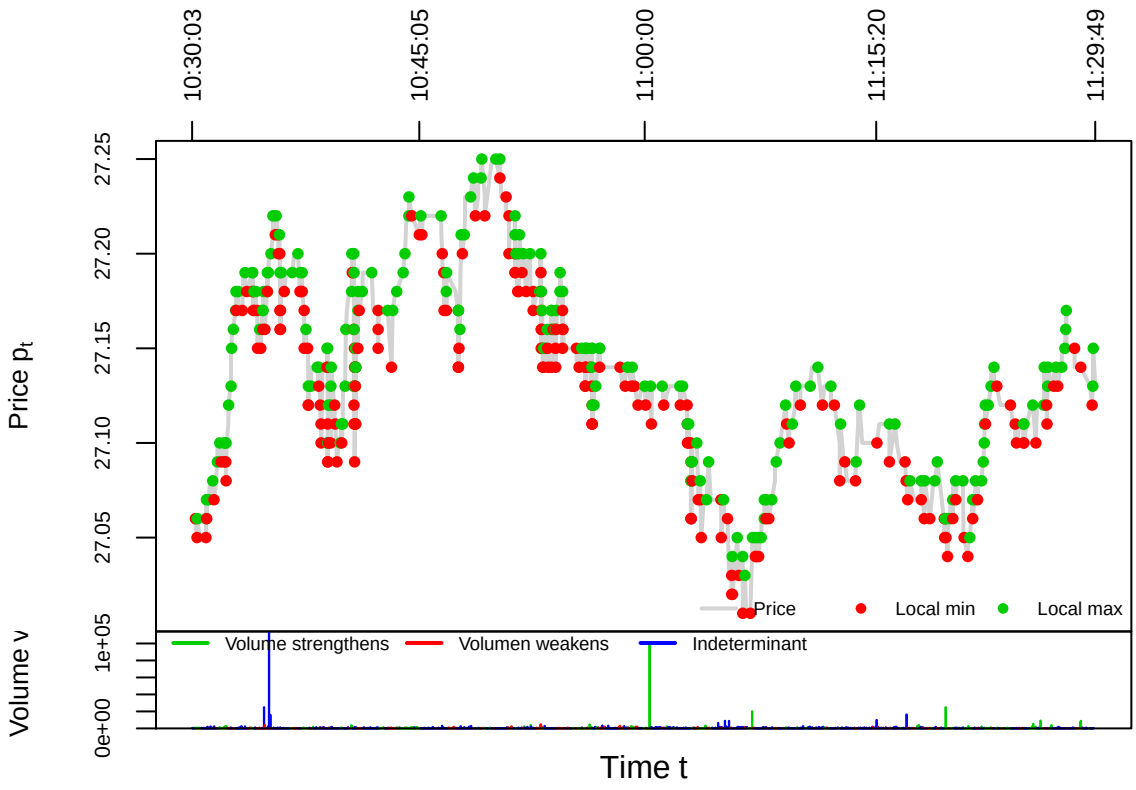


Figure 3: Local extrema detected in TSE:G 2007-05-02 10:30:00/2007-05-02 11:30:00.

Price
Bull 1
Bull 2
Bull 3
Bull 4
Bull 5
Bull 6
Bull 7
Bull 8
Bull 9
Bear 1
Bear 2
Bear 3
Bear 4
Bear 5
Bear 6
Bear 7
Bear 8
Bear 9
Volume strengthens
Volumen weakens
Indeterminant
Time t
Price p_t
Volume v
10:30:03
10:45:05
11:00:00
11:15:20
11:29:49
27.25
27.20
27.15
27.10
27.05
0e+00
1e+05

2.8.2 Estimation

```
##           [,1] [,2]      [,3] [,4]
## [1,] 0.5528861    0 0.4471139    0
```

```
##           [,1]       [,2]      [,3]       [,4]
## [1,] 0.0000000 0.09973768 0.9002623 0.00000000
## [2,] 1.0000000 0.00000000 0.0000000 0.00000000
## [3,] 0.9526642 0.00000000 0.0000000 0.04733579
## [4,] 0.0000000 0.00000000 1.0000000 0.00000000
```

```
##             [,1]        [,2]        [,3]       [,4]      [,5]       [,6]
## [1,] 0.001514451 0.014445768 0.001574060 0.03281536 0.6208348 0.27890734
## [2,] 0.001191884 0.027998759 0.001014878 0.42063382 0.4221437 0.02013664
## [3,] 0.008661794 0.001698369 0.040878628 0.01764925 0.5692420 0.33560239
## [4,] 0.033281095 0.001015034 0.077988802 0.41536518 0.4067408 0.02062659
##              [,7]         [,8]        [,9]
## [1,] 0.0292644387 0.0012753896 0.018640726
## [2,] 0.0850429364 0.0007934046 0.019409951
## [3,] 0.0018515861 0.0134007039 0.002150055
## [4,] 0.0009502016 0.0375767906 0.001054304
```

2.8.3 Convergence

Mixing and convergence to a stationary distribution is very good. Although re-estating the original Hierarchical Hidden Markov Model into a Hidden Markov Model may increase time and memory complexity, the new model becomes far easier to program and convergence.

```
##                   mean      se_mean          sd          50%    n_eff
## p_1k[1]    0.544425885 1.728500e-02 0.273299839 0.5528861210 250.0000
## p_1k[2]    0.000000000 0.000000e+00 0.000000000 0.0000000000 250.0000
## p_1k[3]    0.455574115 1.728500e-02 0.273299839 0.4471138790 250.0000
## p_1k[4]    0.000000000 0.000000e+00 0.000000000 0.0000000000 250.0000
## A_ij[1,1]  0.000000000 0.000000e+00 0.000000000 0.0000000000 250.0000
## A_ij[1,2]  0.116011032 4.821659e-03 0.076237117 0.0997376774 250.0000
## A_ij[1,3]  0.883988968 4.821659e-03 0.076237117 0.9002623226 250.0000
## A_ij[1,4]  0.000000000 0.000000e+00 0.000000000 0.0000000000 250.0000
## A_ij[2,1]  1.000000000 0.000000e+00 0.000000000 1.0000000000 250.0000
## A_ij[2,2]  0.000000000 0.000000e+00 0.000000000 0.0000000000 250.0000
## A_ij[2,3]  0.000000000 0.000000e+00 0.000000000 0.0000000000 250.0000
## A_ij[2,4]  0.000000000 0.000000e+00 0.000000000 0.0000000000 250.0000
## A_ij[3,1]  0.933523145 3.945215e-03 0.060709291 0.9526642124 236.7931
## A_ij[3,2]  0.000000000 0.000000e+00 0.000000000 0.0000000000 250.0000
## A_ij[3,3]  0.000000000 0.000000e+00 0.000000000 0.0000000000 250.0000
## A_ij[3,4]  0.066476855 3.945215e-03 0.060709291 0.0473357876 236.7931
## A_ij[4,1]  0.000000000 0.000000e+00 0.000000000 0.0000000000 250.0000
## A_ij[4,2]  0.000000000 0.000000e+00 0.000000000 0.0000000000 250.0000
## A_ij[4,3]  1.000000000 0.000000e+00 0.000000000 1.0000000000 250.0000
## A_ij[4,4]  0.000000000 0.000000e+00 0.000000000 0.0000000000 250.0000
## phi_k[1,1] 0.002264531 1.437605e-04 0.002273054 0.0015144514 250.0000
## phi_k[1,2] 0.014659927 2.488175e-04 0.003934151 0.0144457678 250.0000
## phi_k[1,3] 0.002435013 1.519305e-04 0.002402232 0.0015740596 250.0000
## phi_k[1,4] 0.035867068 2.432798e-03 0.025044556 0.0328153583 105.9777
## phi_k[1,5] 0.619439915 2.893640e-03 0.031649064 0.6208348144 119.6280
## phi_k[1,6] 0.275651300 1.794260e-03 0.023670594 0.2789073392 174.0393
```

```
## phi_k[1,7] 0.029084082 2.903215e-04 0.004590386 0.0292644387 250.0000
## phi_k[1,8] 0.001787321 1.051107e-04 0.001661945 0.0012753896 250.0000
## phi_k[1,9] 0.018810843 2.405108e-04 0.003802810 0.0186407262 250.0000
## phi_k[2,1] 0.001670825 1.355221e-04 0.001681130 0.0011918838 153.8801
## phi_k[2,2] 0.027481165 4.217307e-04 0.005904408 0.0279987592 196.0118
## phi_k[2,3] 0.001368584 7.894749e-05 0.001248269 0.0010148780 250.0000
## phi_k[2,4] 0.417344280 1.975689e-03 0.026964871 0.4206338153 186.2772
## phi_k[2,5] 0.422213394 2.310259e-03 0.031378309 0.4221436789 184.4750
## phi_k[2,6] 0.024870607 1.641134e-03 0.021459940 0.0201366407 170.9893
## phi_k[2,7] 0.084756432 5.428854e-04 0.008583772 0.0850429364 250.0000
## phi_k[2,8] 0.001068778 6.600594e-05 0.001043646 0.0007934046 250.0000
## phi_k[2,9] 0.019225935 3.457387e-04 0.005466609 0.0194099505 250.0000
## phi_k[3,1] 0.009072240 2.126791e-04 0.003362752 0.0086617935 250.0000
## phi_k[3,2] 0.002588232 1.562501e-04 0.002470532 0.0016983691 250.0000
## phi_k[3,3] 0.040909037 3.451759e-04 0.005457710 0.0408786280 250.0000
## phi_k[3,4] 0.024154183 1.493020e-03 0.020826586 0.0176492484 194.5830
## phi_k[3,5] 0.568179404 2.266569e-03 0.032878417 0.5692420072 210.4186
## phi_k[3,6] 0.335872290 1.951416e-03 0.024832657 0.3356023886 161.9373
## phi_k[3,7] 0.002693853 1.754687e-04 0.002774404 0.0018515861 250.0000
## phi_k[3,8] 0.013421031 2.358499e-04 0.003729114 0.0134007039 250.0000
## phi_k[3,9] 0.003109730 2.288338e-04 0.003048185 0.0021500546 177.4362
## phi_k[4,1] 0.033323718 3.549072e-04 0.005611576 0.0332810948 250.0000
## phi_k[4,2] 0.001495993 8.738685e-05 0.001381707 0.0010150340 250.0000
## phi_k[4,3] 0.078802149 5.327542e-04 0.008423583 0.0779888024 250.0000
## phi_k[4,4] 0.414490295 2.334784e-03 0.028133172 0.4153651794 145.1924
## phi_k[4,5] 0.405077802 2.889220e-03 0.030267337 0.4067408299 109.7456
## phi_k[4,6] 0.026302444 1.601537e-03 0.021805009 0.0206265949 185.3695
## phi_k[4,7] 0.001340625 8.412715e-05 0.001330167 0.0009502016 250.0000
## phi_k[4,8] 0.037535986 3.557716e-04 0.005625242 0.0375767906 250.0000
## phi_k[4,9] 0.001630987 1.046606e-04 0.001654830 0.0010543044 250.0000
##                 Rhat
## p_1k[1]    0.9960420
## p_1k[2]          NaN
## p_1k[3]    0.9960420
## p_1k[4]          NaN
## A_ij[1,1]        NaN
## A_ij[1,2]  0.9973961
## A_ij[1,3]  0.9973961
## A_ij[1,4]        NaN
## A_ij[2,1]        NaN
## A_ij[2,2]        NaN
## A_ij[2,3]        NaN
## A_ij[2,4]        NaN
## A_ij[3,1]  1.0060737
## A_ij[3,2]        NaN
## A_ij[3,3]        NaN
## A_ij[3,4]  1.0060737
## A_ij[4,1]        NaN
## A_ij[4,2]        NaN
## A_ij[4,3]        NaN
## A_ij[4,4]        NaN
## phi_k[1,1] 0.9976105
## phi_k[1,2] 1.0014667
## phi_k[1,3] 0.9976205
```

```
## phi_k[1,4] 1.0208381
## phi_k[1,5] 0.9993311
## phi_k[1,6] 1.0023684
## phi_k[1,7] 1.0046338
## phi_k[1,8] 1.0070245
## phi_k[1,9] 0.9967712
## phi_k[2,1] 0.9965369
## phi_k[2,2] 1.0155352
## phi_k[2,3] 1.0004008
## phi_k[2,4] 0.9998513
## phi_k[2,5] 1.0054966
## phi_k[2,6] 0.9963122
## phi_k[2,7] 0.9960477
## phi_k[2,8] 1.0023561
## phi_k[2,9] 0.9972978
## phi_k[3,1] 1.0016432
## phi_k[3,2] 1.0055663
## phi_k[3,3] 0.9960141
## phi_k[3,4] 0.9969357
## phi_k[3,5] 1.0000072
## phi_k[3,6] 0.9988277
## phi_k[3,7] 0.9992319
## phi_k[3,8] 0.9992699
## phi_k[3,9] 0.9962634
## phi_k[4,1] 0.9964343
## phi_k[4,2] 1.0032324
## phi_k[4,3] 0.9970590
## phi_k[4,4] 1.0106517
## phi_k[4,5] 1.0002961
## phi_k[4,6] 0.9981026
## phi_k[4,7] 0.9971450
## phi_k[4,8] 0.9963979
## phi_k[4,9] 0.9963979
```

Comments.

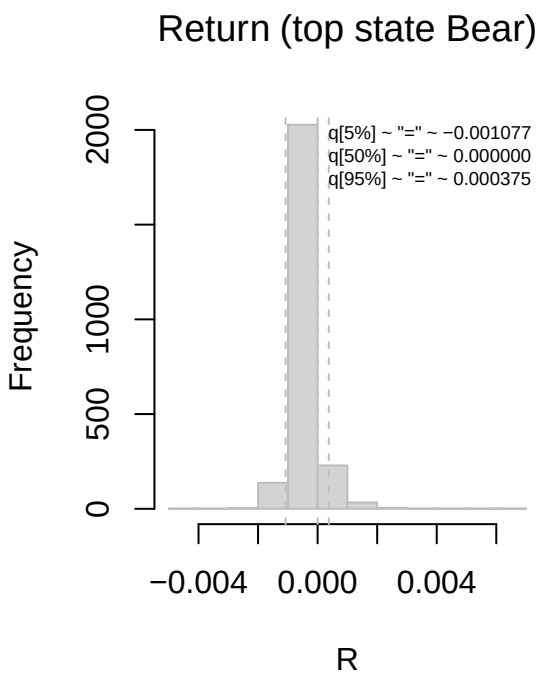
2.8.4 State probability

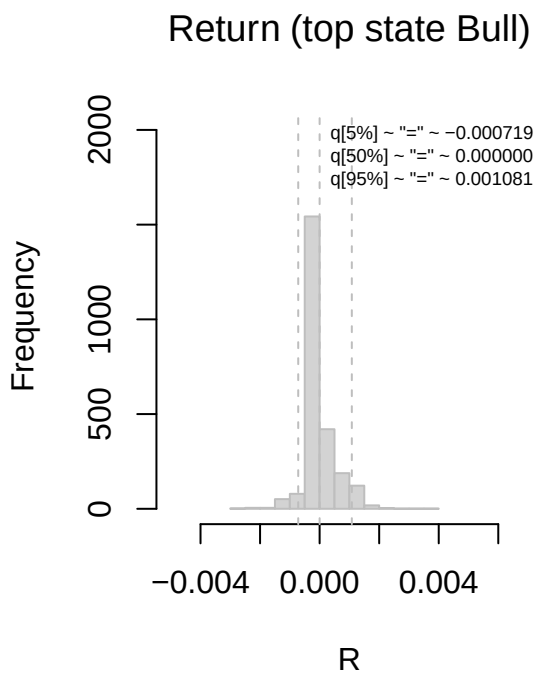
```
## [1] "I identified top-nodes as bears and bulls.\n                Result: Bear = -0.0001% vs Bull = 0
```

Observations are hard classified as belonging to the emission state (one of the four possible bottom-node) with largest forward probability. By the definition of the hierarchical model, they become naturally linked to one of the two possible top states (bears or bulls).

The following table provides some summary statistics for the returns of the observations classified in each state. The structure of the two top nodes are symmetrical, thus they do not have a predetermined label. We label them as run and reversals according to the expected trade returns observed in-sample.

	ret_mean	ret_stdev	ret_skewness.ret	ret_kurtosis.ret	ret_iqrange.25%	ret_iqrange.50%	ret_iqrange.75
Bear	-0.0001185	0.0005063	1.058093	25.931461	-0.000361	0	0.00000
Bull	0.0001176	0.0004938	0.296264	9.274487	0.000000	0	0.00036





Analyse mean, sd, kurtosis, etc.

-0.00
0.00
0.00
0.00
1.06
0.30
25.93
9.27
-0.00
0.00
0.00
0.00
0.00
0.00
8.23
7.35
5.00
4.00

...

2.8.5 Fitted output

Now, look how at how the features are classified:

The model has four production/emission states: 1, 2, 3 and 4. 1 and 3 for negative legs and 2 and 4 for

positive legs. States 1 and 2 belong to the "bull" state while 3 and 4 belong to "bear" state. There can be negative legs in bull states (they're negative but with little volume) and positive legs in bear states (positive but, again, with little volume).

Our sampler, tho, maps each of the 18 possible features into one state. For example, observations from feature 3 (ie U3) are all mapped into state 4. In a sense, this adds very little information since imputation is almost deterministic once the parameters are estimated.

1. Look at forward probabilities (the two columns are repeated since I don't compute smoothed probabilities to improve computational time)
2. Then the distribution of returns inside each "top state" (either bull or bear). These top states should have two children state for positive/negative legs each. States 1 and 2 = Bull (greater mean return), states 3 and 4 = bear (smaller mean return).

...

2.8.6 In-sample summary

...

2.8.7 Forecast

...

2.8.7.1 Walking forward forecasts

...

2.8.8 Trading strategy

...

2.9 Discussion

2.9.1 The statistical model

...

2.9.2 The financial application

...

---

3 Original Computing Environment

```
##
## CXXFLAGS=-O3 -mtune=native -march=native -Wno-unused-variable -Wno-unused-function
```

```
##  setting  value
##  version  R version 3.3.3 (2017-03-06)
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
##  collate  Spanish_Argentina.1252
##  tz       America/Buenos_Aires
##  date     2017-08-21
##
##  package      * version   date       source
##  BH             1.62.0-1  2016-11-19 CRAN (R 3.3.2)
##  colorspace     1.3-2     2016-12-14 CRAN (R 3.3.3)
##  dichromat      2.0-0     2013-01-24 CRAN (R 3.3.2)
##  digest       * 0.6.12    2017-01-27 CRAN (R 3.3.3)
##  ggplot2      * 2.2.1     2016-12-30 CRAN (R 3.3.3)
##  graphics     * 3.3.3     2017-03-06 local
##  grDevices    * 3.3.3     2017-03-06 local
##  grid           3.3.3     2017-03-06 local
##  gridExtra      2.2.1     2016-02-29 CRAN (R 3.3.3)
##  gtable         0.2.0     2016-02-26 CRAN (R 3.3.3)
##  inline         0.3.14    2015-04-13 CRAN (R 3.3.3)
##  labeling       0.3       2014-08-23 CRAN (R 3.3.2)
##  lattice        0.20-34   2016-09-06 CRAN (R 3.3.3)
##  lazyeval       0.2.0     2016-06-12 CRAN (R 3.3.3)
##  magrittr       1.5       2014-11-22 CRAN (R 3.3.3)
##  MASS           7.3-45    2016-04-21 CRAN (R 3.3.3)
##  Matrix         1.2-8     2017-01-20 CRAN (R 3.3.3)
##  methods      * 3.3.3     2017-03-06 local
##  munsell        0.4.3     2016-02-13 CRAN (R 3.3.3)
##  plyr           1.8.4     2016-06-08 CRAN (R 3.3.3)
##  RColorBrewer   1.1-2     2014-12-07 CRAN (R 3.3.2)
##  Rcpp           0.12.10   2017-03-19 CRAN (R 3.3.3)
##  RcppEigen      0.3.2.9.1 2017-03-15 CRAN (R 3.3.3)
##  reshape2       1.4.2     2016-10-22 CRAN (R 3.3.3)
##  rstan        * 2.14.2    2017-03-19 CRAN (R 3.3.3)
##  scales         0.4.1     2016-11-09 CRAN (R 3.3.3)
##  StanHeaders  * 2.14.0-1  2017-01-09 CRAN (R 3.3.3)
##  stats        * 3.3.3     2017-03-06 local
##  stats4         3.3.3     2017-03-06 local
##  stringi        1.1.3     2017-03-21 CRAN (R 3.3.3)
##  stringr        1.2.0     2017-02-18 CRAN (R 3.3.3)
##  tibble         1.3.0     2017-04-01 CRAN (R 3.3.3)
##  tools          3.3.3     2017-03-06 local
##  utils        * 3.3.3     2017-03-06 local
```