Prediction, Heuristics, and Volatility

Chris Georges

Hamilton College

WEHIA July 2024

1. Introduction

- Explore the consequences of the **prediction technologies** available to trader/investors for market volatility and instability.
- Focus on the role of **regularization** in forecasting for driving **empirical regularities** in financial market data
- Traders are in a **competitive battle** to uncover **fleeting predictable structure** and exploit it before others do so. They face:
  - access to a **wealth of data**
  - model **uncertainty**
  - **non-stationary** environment
  - **over-regularization** of the forecasting model will potentially leave predictable structure on the table.
  - **under-regularization** will potentially cause them to chase spurious trends.
- In our **model**, **dynamic L1 regularization** can generate empirical regularities and can mimic two type models endogenously.
- Further, **ML methods** that forecast well on exogenous data may have little empirical content in a **self referential context**.

Some Background

In a world of deep complexity/uncertainty + big data + competition

- Tension between predictability and overfitting (e.g., D'amour et al., (2020), Nagel (2021))
- Simple **heuristics** (e.g., Anufriev and Hommes (2012)) may outperform more complicated prediction methods
  - Effectively negotiate bias-variance tradeoff (e.g., Gigerenzer and Brighton (2009), Hansen (2020))
  - Interpretability (for risk management, regulatory compliance, marketing – e.g., Hansen (2020))
  - Systemic efficiency (e.g., Dosi et al. (2020))
- **ML** is explicitly designed to navigate this tension – has had increasing success in TS forecasting
  - early on, complex methods were dominated by simpler statistical methods or model averaging - e.g., M4 (Makridakis et al., 2018; Slawek, 2020).
  - substantially more success recently – e.g., NNs, LightGBM
    - ★ M5, M6 competitions (Makridakis et al., 2022, 2023)
    - ★ Gu et al. (2020), Ryll and Seidens (2019), Kelly and Xiu (2023)

- **ML** essentially entertains **wide variety of predictors** and highly **non-linear relationships** with target variables, and uses **regularization** and **out of sample testing** to mitigate overfitting.
- **Regularization** constrains the **complexity** of the forecasting model, often by adding a **complexity penalty**
- Still will generally have **overfitting**
  - driven by purely random patterns in the data
  - as well as propagation via trader behavior
  - aggravated if relaxation of degree of caution
- Below, **endogenous selection of degree of regularization can be a powerful driver of volatility patterns**.
  - and can mimic two type model, via endogenous selection
- Note that success in forecasting in e.g., M6 is on **exogenous data**. Here we are interested in empirical validity in **self referential environments**.

2. Thought Experiment: Fitting and Overfitting an Exogenous Time Series

- **Universal approximation theorem** (Hornik et al. (1989), Cybenko (1989)): implies that a feed forward NN with enough nodes in a single hidden layer can (over)fit any training data to arbitrary precision. Below we provide an **illustration** on purely artificial data.
- **Consider purely exogenous artificial data:** $x_t = x^* + \epsilon_t$, where x^* is a constant (below set to 10.5), and ϵ is a mean zero dividend shock (below uniform on (-0.5,0.5)) .
- **For reference**, the **RE** prediction in this case is that x will be $x^* = 10.5$ in every period. If x^* was unknown, it could be consistently estimated with the sample mean.
- **Now consider prediction with a single layer feed forward NN for which the predictors are p lags of x_t.** Increasing the number of nodes in NN or increasing the number of available predictors p will increase the fit in the training sample, thus overfitting the time series.

Figure 1: Fitted and target values of exogenous artificial x series from neural network with **5 predictors** and **2 nodes** in a single hidden layer. Here the MAD from the RE forecast is 0.033, and the MAE is approx. 0.194. For reference the corresponding MAE under the RE prediction for the 200 period training set would be approx. 0.191.

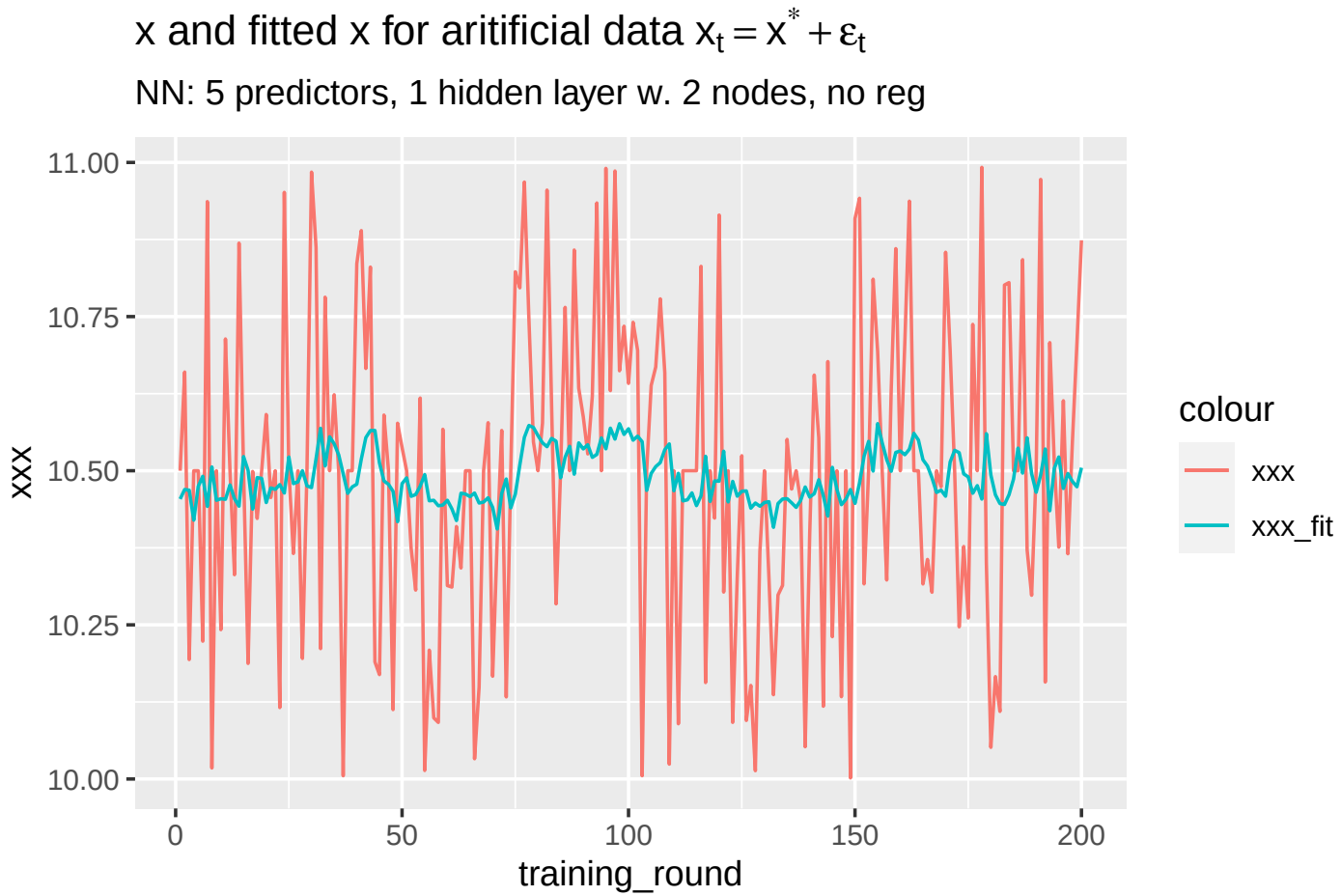


Figure 2: Fitted and target values of exogenous artificial x series from neural network with **5 predictors** and **10 nodes** in a single hidden layer. Here the MAD from the RE forecast is 0.059, and the MAE is approx. 0.177

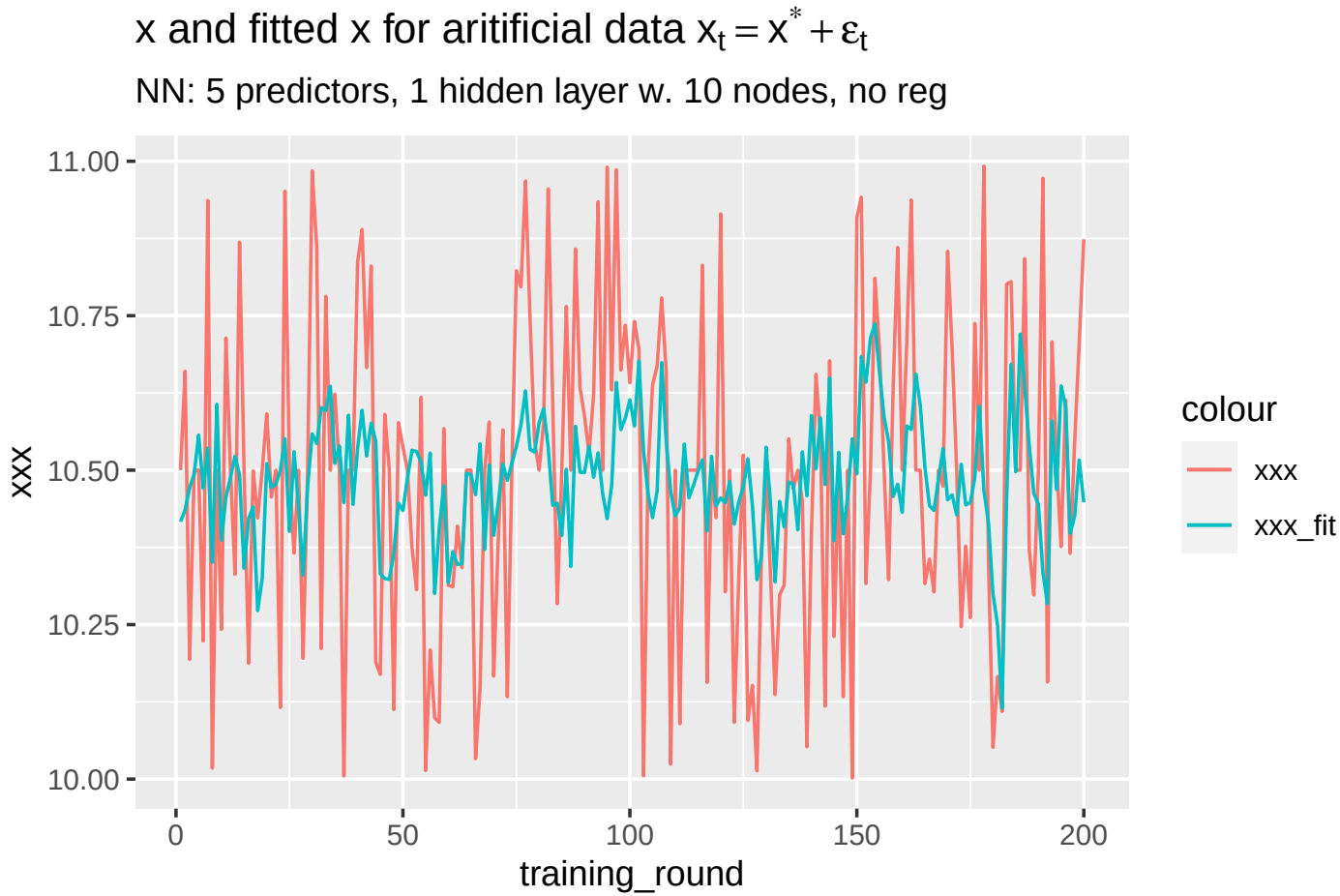


Figure 3: Fitted and target values of exogenous artificial x series from neural network with **5 predictors** and **100 nodes** in a single hidden layer. Here the MAD from the RE forecast is 0.161, and the MAE is approx. 0.101

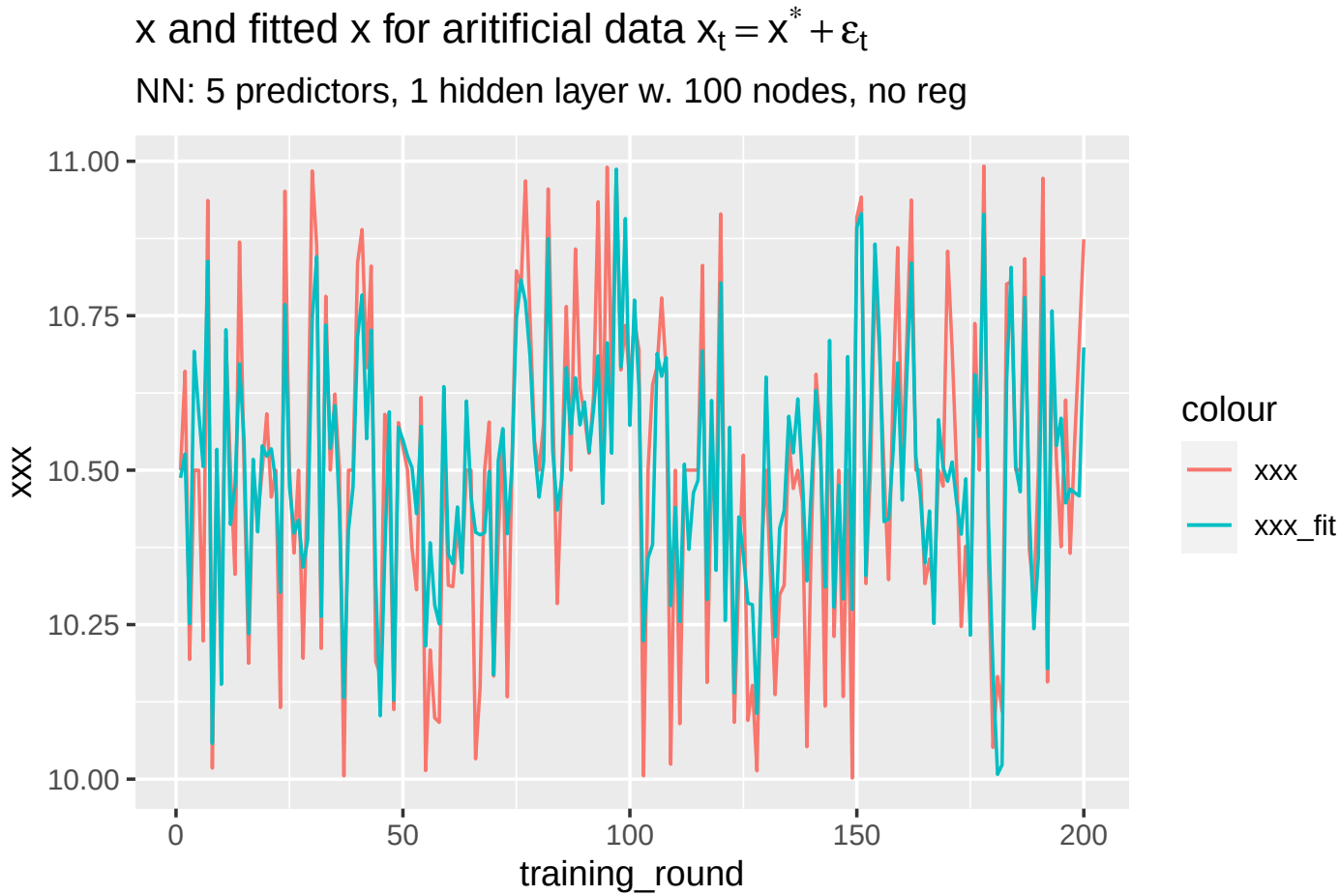
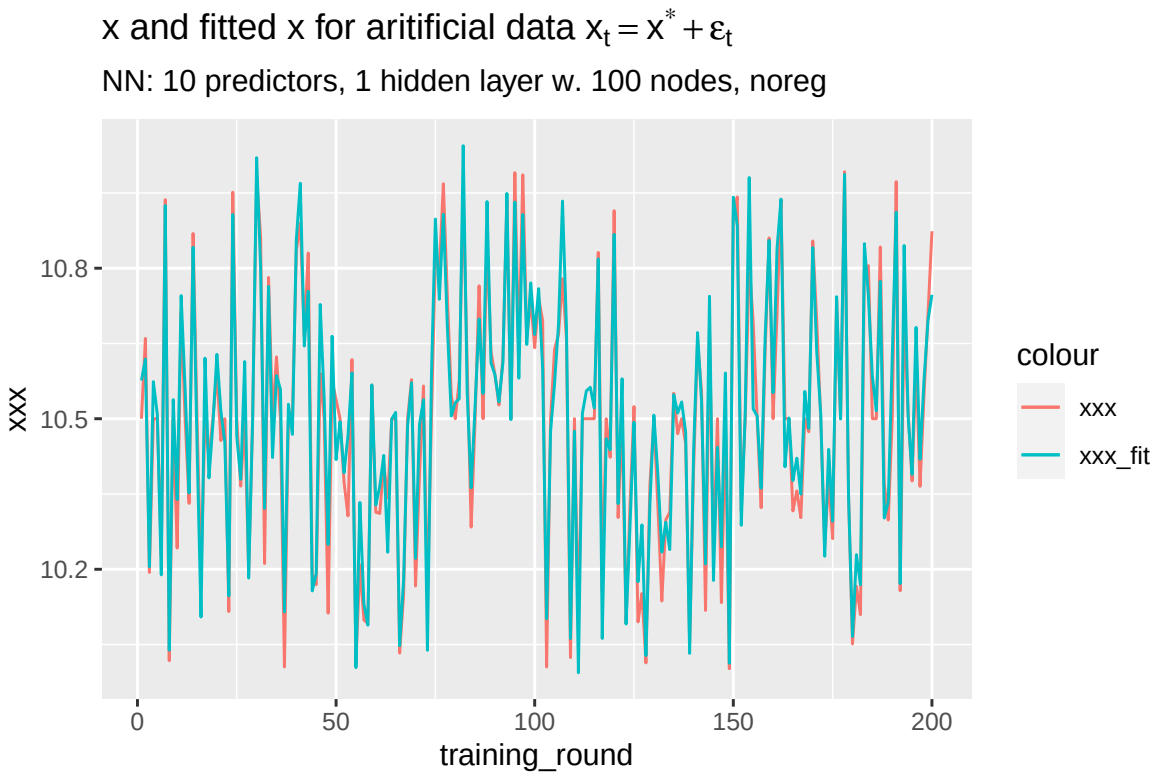


Figure 4: Fitted and target values of exogenous artificial x series from neural network with **10 predictors** and **100 nodes** in a single hidden layer. Here the MAD from the RE forecast is 0.191, and the MAE is approx. 0.048. Fitted network has 1100 kernel weights with Frobenius norm approx. 7.679.



So the NN is prone to extreme overfitting, ...unless mitigated via regularization.

Figure 5: Fitted and target values of exogenous artificial x series from neural network with **10 predictors** and **100 nodes** in a single hidden layer. **L1 regularization 0.005**. Here the MAD from the RE forecast is 0.035, and the MAE is approx. 0.189. Frobenius norm of kernel weights approx. 0.426.

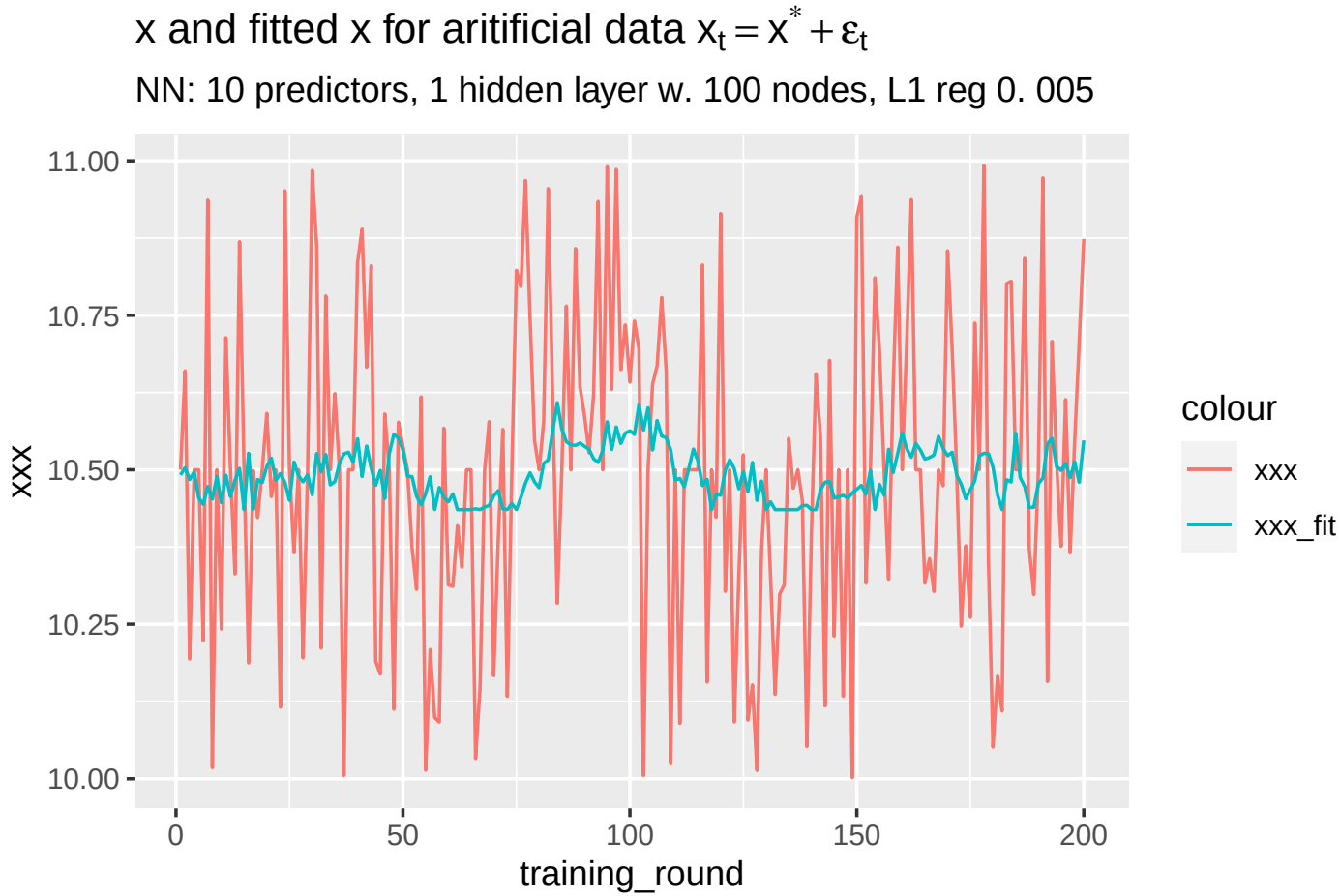
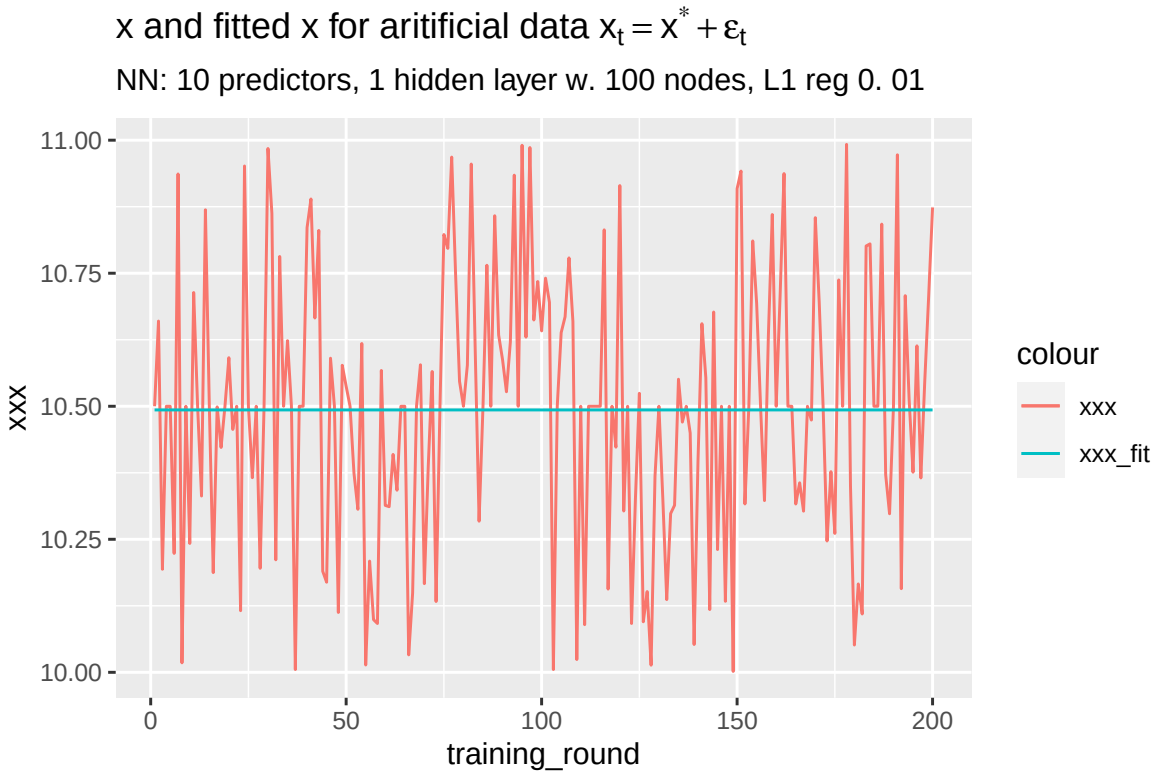


Figure 6: Fitted and target values of exogenous artificial x series from neural network with **10 predictors** and **100 nodes** in a single hidden layer. **L1 regularization 0.01**. Here the MAD from the RE forecast is 0.009, and the MAE is approx. 0.192. Frobenius norm of kernel weights approx. 0.006.



So with a high enough regularization penalty, the NN selects the simplicity heuristic.

- So here we can see the **tension** between the desire to uncover predictable structure and the ease of overfitting (fitting noise).
  - With sufficient regularization (e.g., a large enough penalty in the loss function) the NN will select the **simplicity heuristic**
  - But this would also rule out the possibility of exploiting **predictable structure**, if such structure were to exist.
- We can also see that **with a given level of regularization intensity** (L1 penalty), random patterns in the data can still pull the prediction away from x^*.
- **Further**, as we will see below, if forecasters select this regularization intensity dynamically (e.g., by CV), this **intensity will vary over time** and itself be driven by patterns in the data. I.e., there will be episodes in which the data appears to be relatively more predictable, and so traders will jump to exploit these fleeting windows or "pockets of predictability" (Farmer et al, 2023, Chinco et al, 2019), setting off further **volatility**.

3. Model Environment

- Simple (toy) workhorse market model with stationary fundamentals. Extends Georges and Pereira (2021).
- The agents are chartists, forecast returns, uncertain about fundamentals and the beliefs of the other traders – willing to entertain a wide range of patterns
  - A strong **simplicity heuristic** is consistent with RE (fundamentals)
  - They can in principle learn the simplicity heuristic, but have to fight **overfitting** the available data
  - Agents dynamically select of **L1 regularization intensity** by CV

Market Structure and Price Determination

- **Two assets**:
  - a stock with stochastic dividend $d_t = \overline{d} + \varepsilon_t$ and price P_t
  - a bond with known fixed return r.
- A risk neutral trader i would be **indifferent** between holding the stock or the bond **if her forecast satisfied**

$$P_t = \frac{F_t^i[P_{t+1} + d_{t+1}]}{1 + r} \tag{1}$$

- **Arbitrage:** assume the market price in period t satisfies (1) for an average risk neutral trader:

$$P_t = \frac{\overline{F}_t[P_{t+1} + d_{t+1}]}{1 + r} \tag{2}$$

Prediction Architecture for Agents

- The agents know r but do not know the dividend process and must **forecast** $x_{t+1} \equiv (P_{t+1} + d_{t+1})$.
- We give the agents a **common prediction technology:**
  - A modeling famework
  - An estimation method: may include model selection, regularization, cross validation
- **Heterogeneity:** estimation is asynchronous, each period some subset of the population re-estimates, so fitted models are heterogeneous
- Agents forecast each period given their current fitted models

4. Example Prediction Architecture: Polynomial Regression

- **Example Modeling Framework:** All agents use forecast rules with a common **polynomial autoregressive** functional form,

$$F_t^i[x_{t+1}] = \text{Poly}(x_t, x_{t-1}, x_{t-2}, ...) \tag{3}$$

  with coefficients $a = (a_{0t}^i, a_{1t}^i, ...)$ that vary across agents i and time t.

- **Specific Example:** quadratic AR(3) rule,

$$F_t^i[x_{t+1}] = a_{0t}^i + a_{1t}^i \cdot x_t + a_{2t}^i \cdot x_{t-1} + a_{3t}^i \cdot x_{t-1}^2 + a_{4t}^i \cdot x_{t-2} \tag{4}$$
$$+ a_{5t}^i \cdot x_{t-2}^2 + a_{6t}^i \cdot x_{t-1} \cdot x_{t-2}$$

- **Example Estimation Methods** for this case:
  - Georges and Pereira (2021) focus on comparing:
    - ⋆ **OLS** ("least squares learning")
    - ⋆ **LASSO** ("penalized regression")
  - LASSO is similar to OLS but with a complexity penalty added to the loss function. The LASSO coefficient estimates $(a_{0t}^L, a_{1t}^L, \ldots)$ minimize

$$\sum_{k=1}^{M}(x_{t-k} - F[x_{t-k}])^2 + \lambda_t \cdot \sum_{j=1}^{p} |a_{jt}| \tag{5}$$

    where p is the number of regressors (predictors) in the regression.[1]
  - **LASSO**'s L1 regularization forces some or all slope parameter estimates to zero, generating a **sparse** and relatively **interpretable** fitted model. If all are estimated to be zero we have our **simplicity heuristic**.

[1] $p = 6$ in (4).

- Given polynomial AR forecasting model, Lasso with k-fold CV to dynamically select the penalty hyperparameter λ_t (G&P, 2021):
  - generates simple **interpretable** fitted prediction models
  - substantially **mitigates overfitting** by the agents, and so
  - **reduces volatility and instability** in the market
  - however, does not entirely eliminate overfitting – survival of apparent **pockets of predictability**, as empirically documented by e.g., Chinco, Clark-Joseph, and Ye (2019) and Farmer, Schmidt, and Timmermann (2023).
  - **agents spend much of their time selecting** the minimal **simplicity heuristic** -just averaging recent values of x – (here consistent with RE), **but periodically become active chartists**.
  - this is a **strong mechanism** for driving **volatility clustering** (with slow decay) and **fat tails** – indeed often too strong
  - **Akin to** the switching in **two type models**. Here all agents are chartists, and there is a continuum of chartist types, but chartists frequently **select a corner solution** where they behave like fundamentalists.
  - Further investigation (Georges, 2022) highlights the **importance of the endogenous penalty** under CV **in driving this intermittency** between periods of stasis and pockets of predictability and volatility.

Figure 7: **LASSO:** AR(3) Quadratic ($p = 6$)(Rule (4)). Prices, returns, and penalties. for 2,500 rounds of a representative run with LASSO updating and **CV** selection of λ_t.

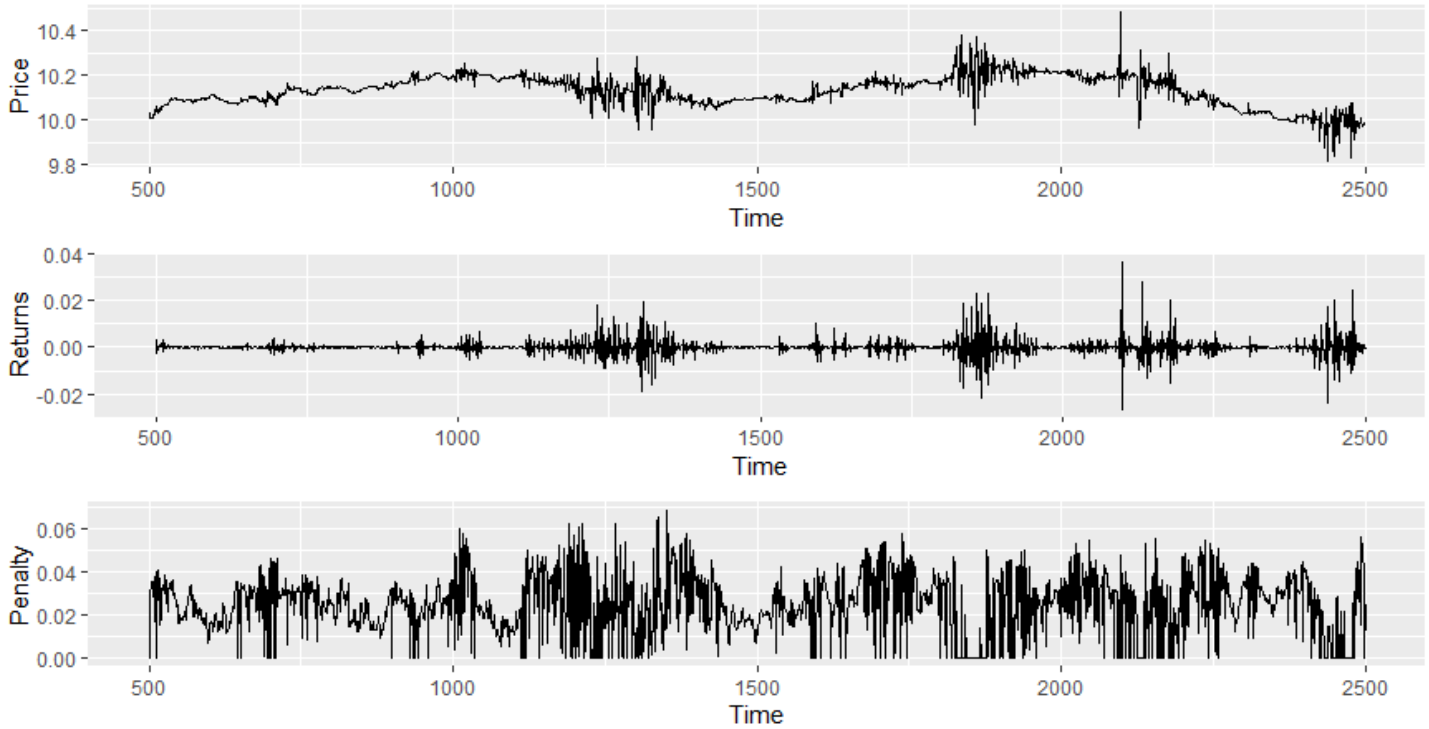


Figure 8: **Optimal agent rule parameters** ($\overline{a}_0$,...,$\overline{a}_6$) – run in Figure 7.

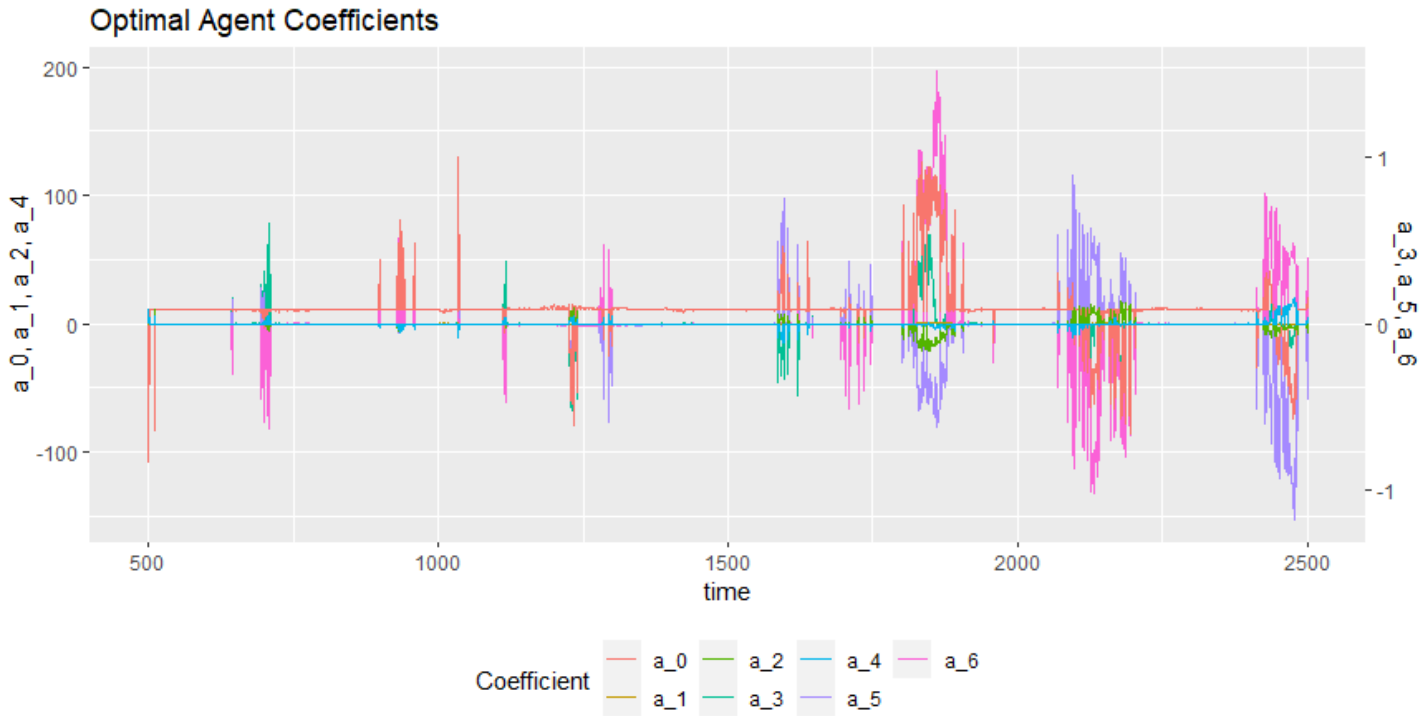
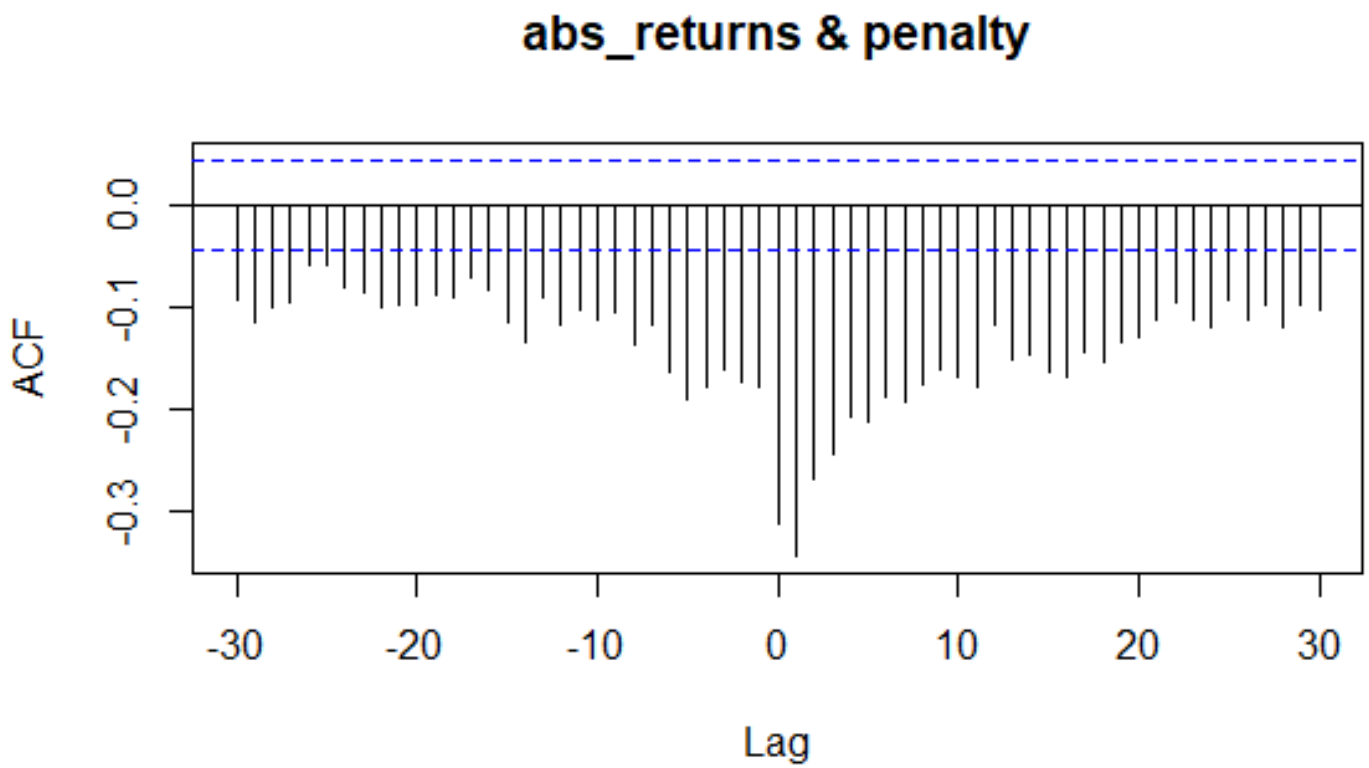


Figure 9: **Cross Correlation Function**: absolute returns and penalty parameter. Run in Figure 1



Compare to the same model with agents using unpenalized **OLS updating rather than Lasso**:

Figure 10: **OLS Updating, AR(3) Quadratic ($p = 6$)**: Same as Figure 1 except OLS estimation without penalty.

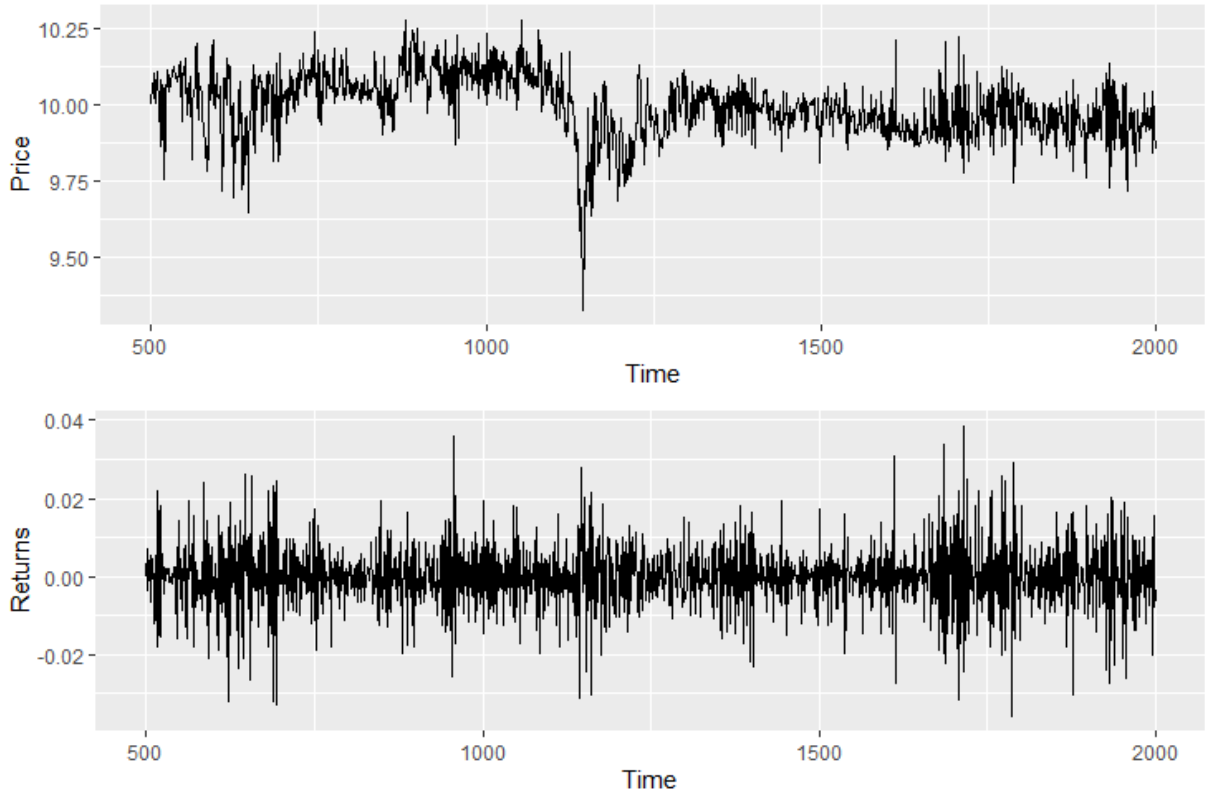


Figure 11: **Optimal Agent rule parameters:** Figure 4.

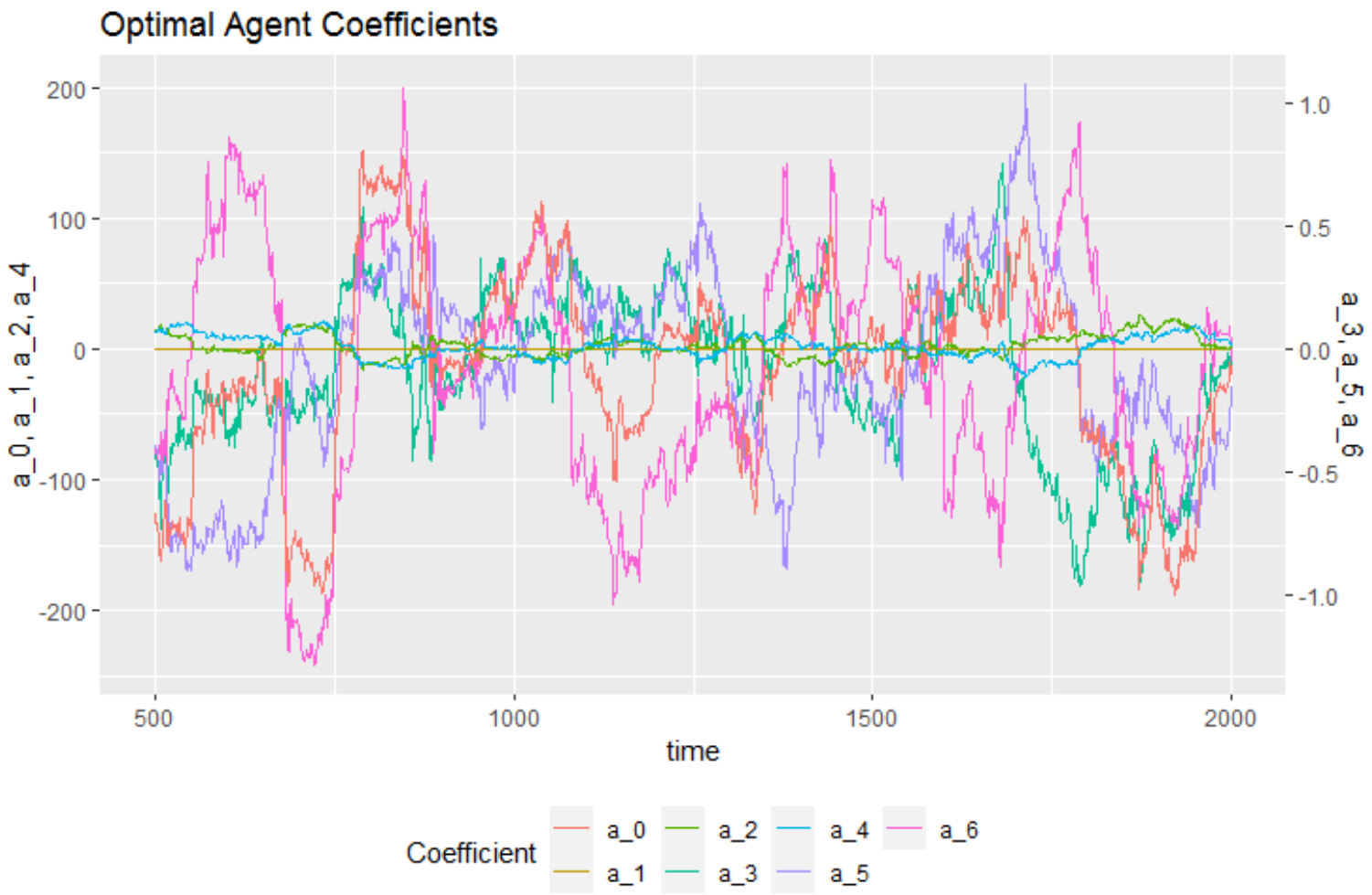
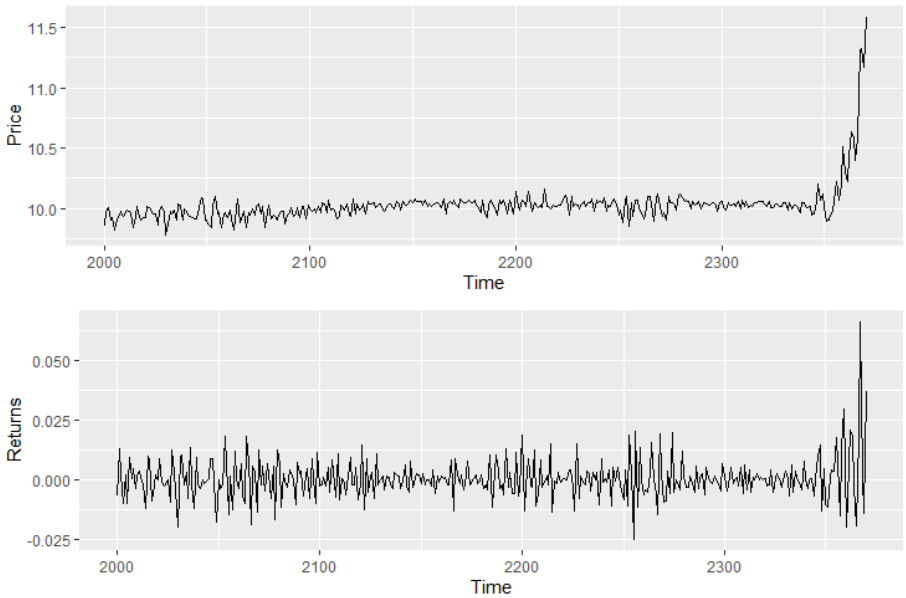
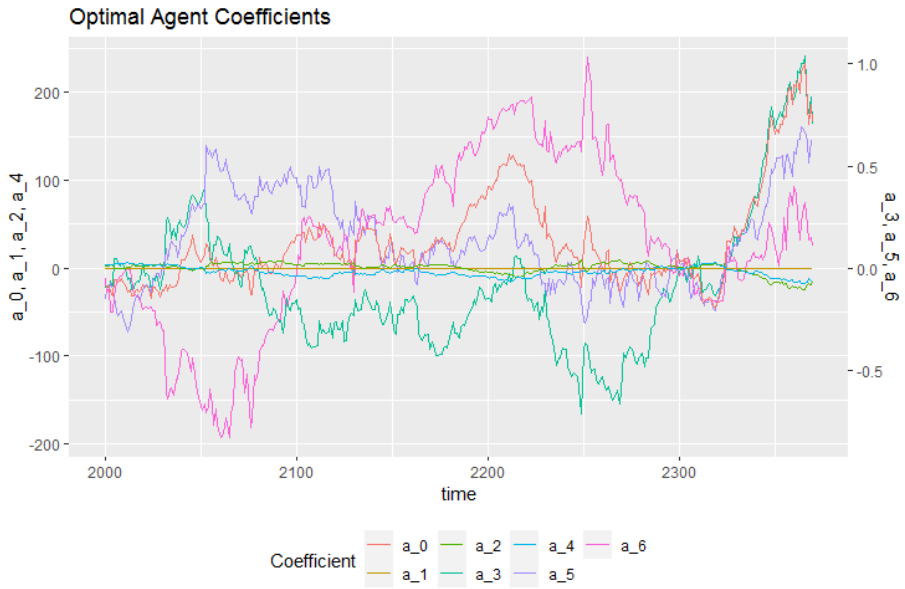


Figure 12: **Bubble formation** in the last run in subsequent simulation rounds.





5. Neural Networks

- The p predictors are now just p lags of x_t (no non-linear pre-processing of the inputs).

Neural Networks:

- Suppose e.g., that the neural network has a single hidden layer with h nodes, and a single output with linear activation.
  - The p predictors are each fed to the h nodes in the hidden layer, linearly combined, these sums nonlinearly transformed by the activation function, and the product combined linearly to produce the single output value, which serves as a prediction of x_{t+1}. Then we have

$$F[x_{t+1}] = b_2 + w_2' g(\mathbf{b_1} + \mathbf{W_1 x}_t) \tag{6}$$

  - Where the b's and w's are coefficients or "weights" with the b's often called "biases." $g()$ is a non-linear activation function (e.g., tanh, or relu). Here b_2 is a scalar, w_2 and $\mathbf{b_1}$ are vectors, and $\mathbf{W_1}$ is an h×p matrix.
  - The weights are **akin to** the coefficients of the **linear regression model**, and indeed, if g() was linear, the entire network would reduce to a linear AR(p) model.

- This simple structure allows for **substantial flexibility** and is akin to Facebook's AR-Net approach (Triebe et al., 2019). Vs. more complex NN frameworks for TS forecasting (convolutional, recurrent/LSTM, transformers).
- Can add hidden layers to get deeper network (deep learning) and add regularization via penalization, dropout, early stopping, etc.
  - We include **L1 complexity penalty selection via CV**.
  - We also **favor the simplicity heuristic** by normalizing the predictors by the theoretical equilibrium mean x^* and σ_d as well as by having training start with the kernel weights distributed around 0 but the output bias initialized at x^*. So training starts in the neighborhood of the RE-consistent simplicity heuristic.

Figure 13: **Neural Network:** Returns and Penalties for 2000 period NN run (TF) with CV. Here there is a single hidden layer with 50 nodes, the predictors are 5 lags of x, and the penalties selected by CV can take values 0.001, 0.01, or 0.1.

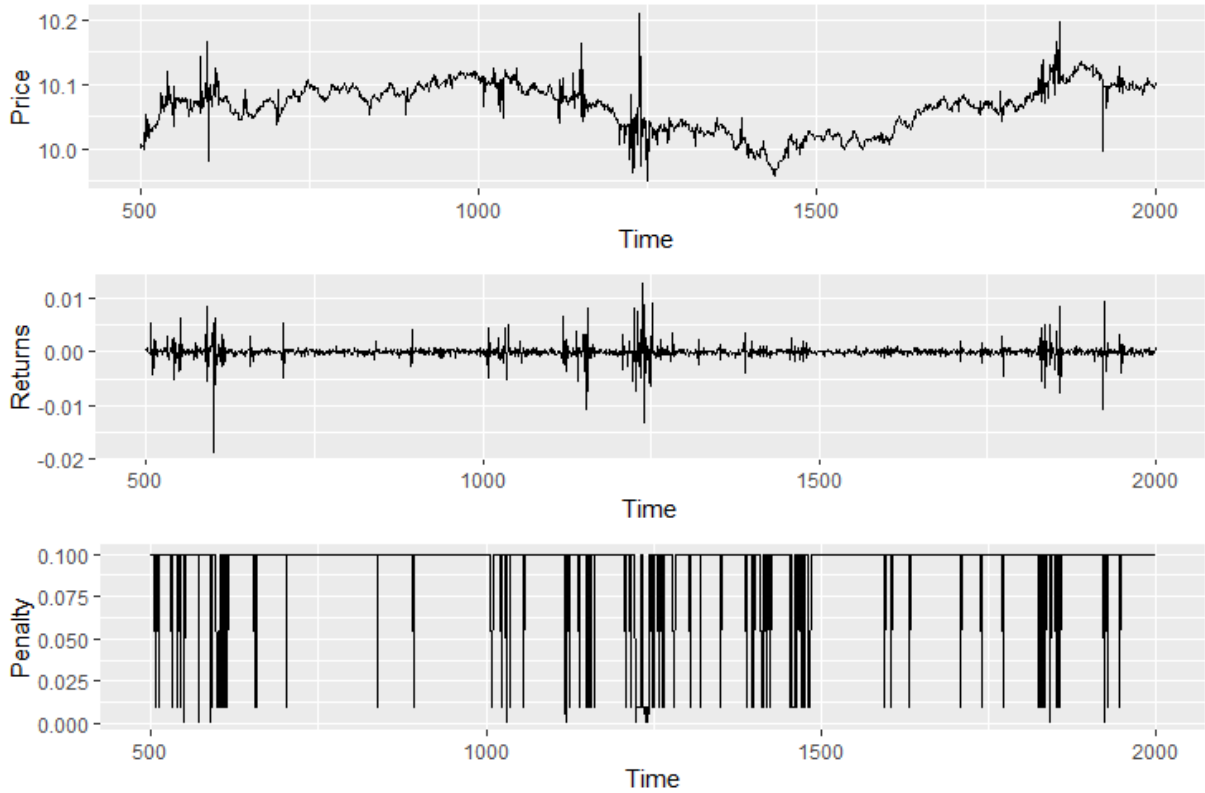
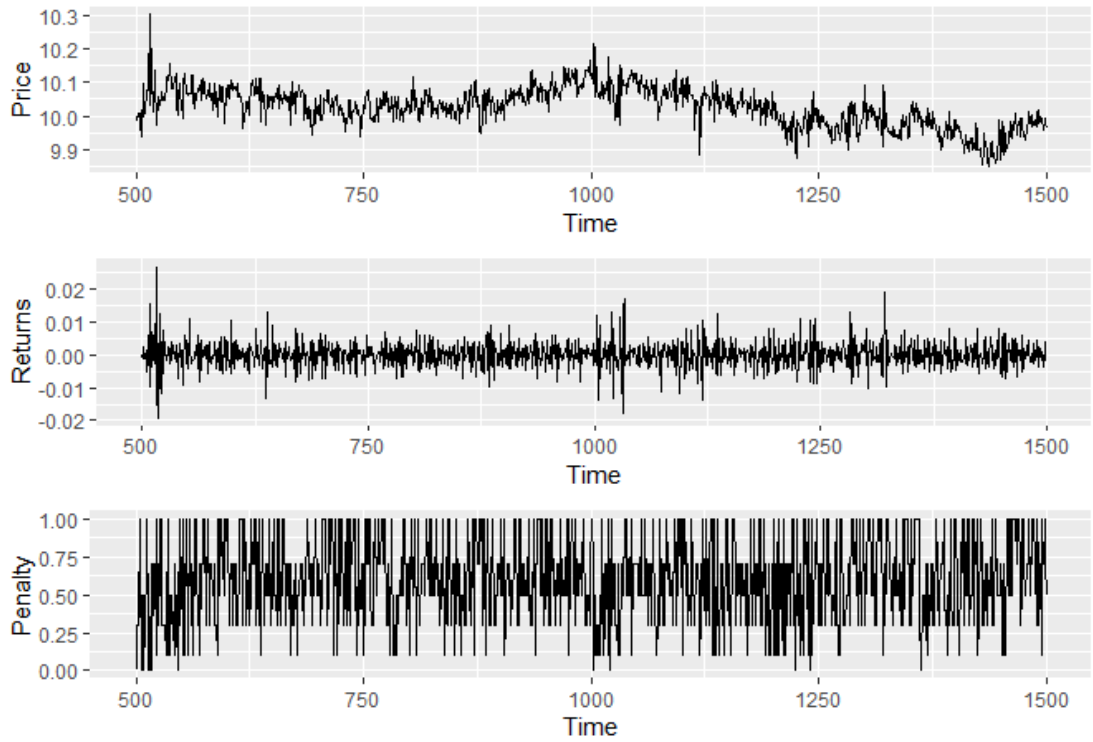


Figure 14: **Neural Network:** Returns and Penalties for 1000 period NN run (TF) with CV on a **broader penalty set**. Single hidden layer with 20 nodes, the predictors are 5 lags of x, and the penalties selected by CV can take values 0, 0.1, 0.3, 0.5, 0.7, 1.0.

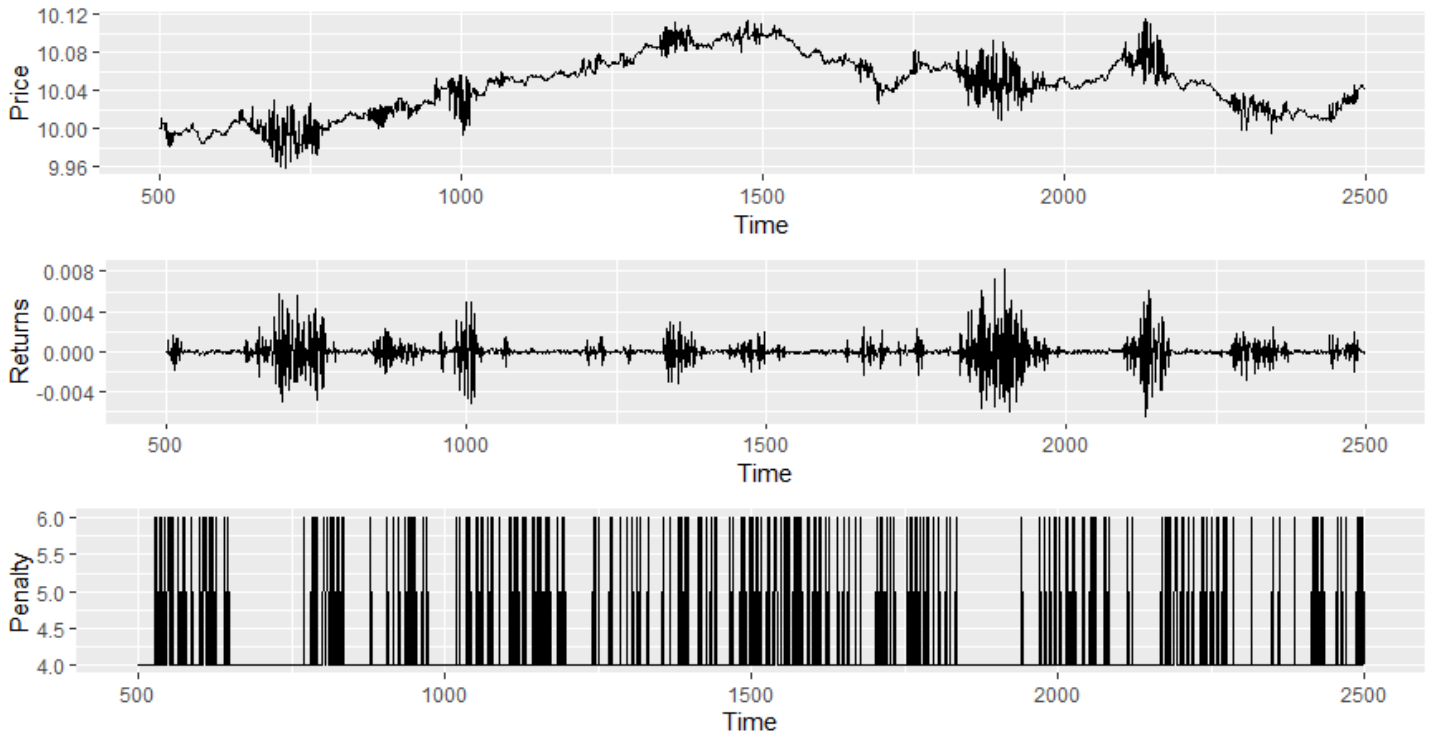


6. Regression Trees, Random Forests, and Gradient Boosting Machines

Regression Trees, Random Forests, and Gradient Boosting Machines:

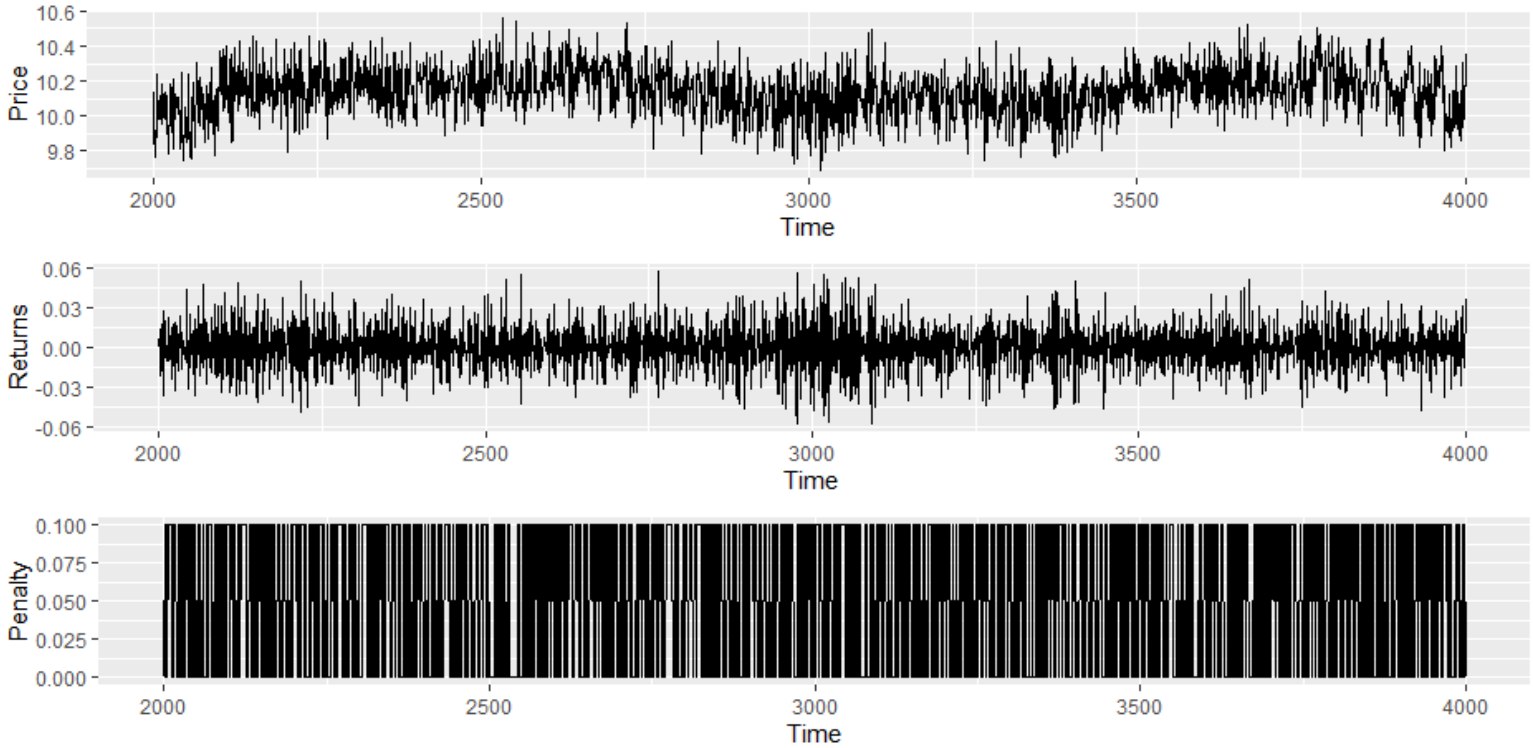
- **Regression Tree:** partition the space of predictors (features) sequentially to minimize a loss function. Prediction for each resulting "leaf" is the average value of the outcome variable in sample at that leaf.
- **Random Forest:** model averaging over a set of regression trees
- **Gradient Boosting Machine:** sequentially train simple trees (weak learners) on the residuals from the last fitting
- Regularization: can again be imposed via penalty selected by CV, by limiting the depth of the trees, etc.
- Advantages: tend to perform reasonably well off the shelf.

Figure 15: **GBM:** Returns and Penalties for 2,500 period LightGBM run with CV. Here the predictors are 5 lags of x, agent memory is 200 periods, and cross validation is over the penalty parameter which can take values of 4 or 6.



Forcing relatively large values of the penalty parameter.

Figure 16: Returns and Penalties for 4,000 period LightGBM run with CV on less extreme penalty set: 0.0, 0.05, 0.1.



And if add larger values of penalty in CV (as in Fig. 14), they tend not to be selected.

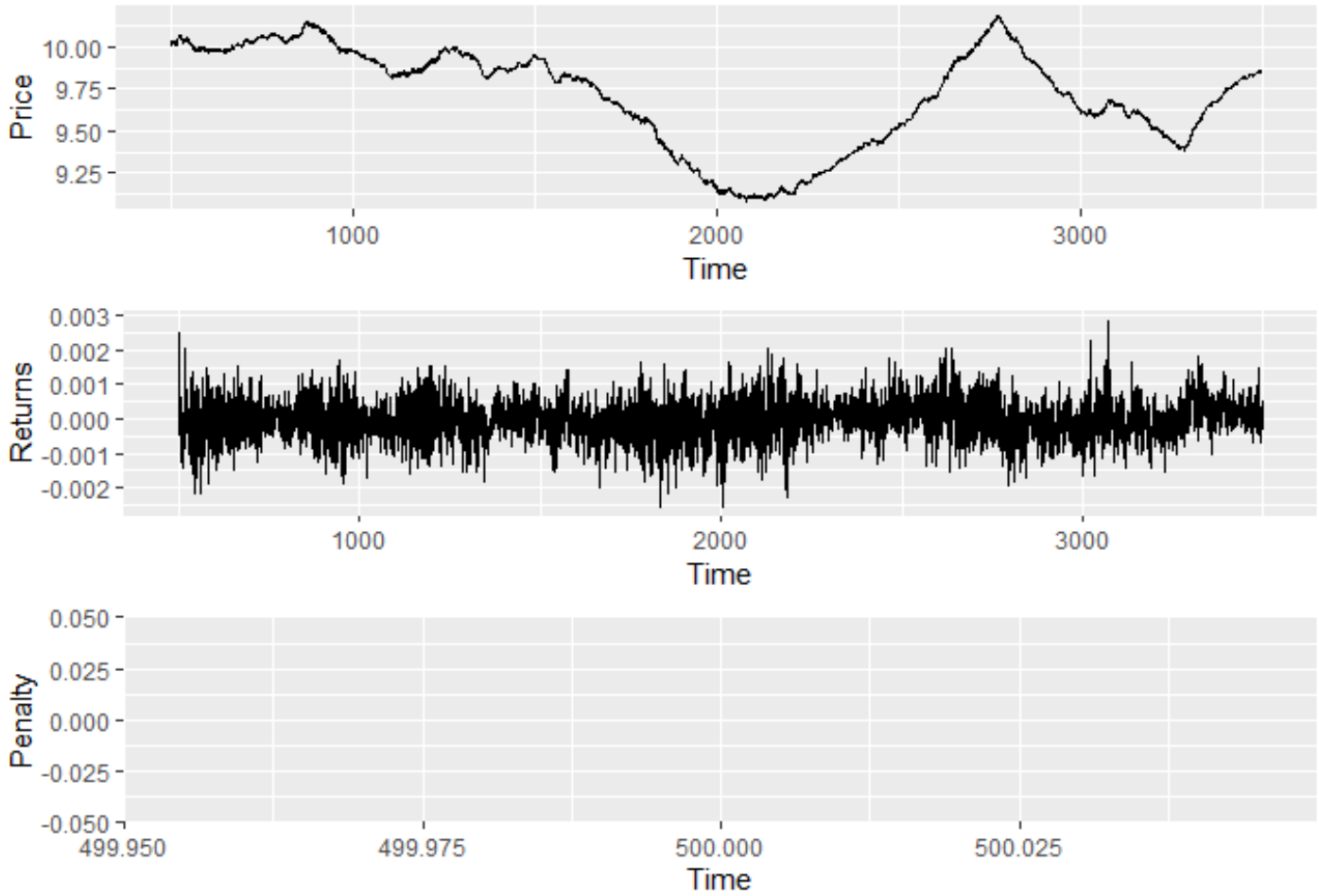
7. 3rd Party ML TS Packages

- **Prophet (Meta):** variable trend + seasonality, L1 regularization
- **TSMixer (Google), TimesNet, PatchTST, N-BEATS, N-HiTS**: NN based (deep learning)
- **TimesFM (Google), TimeGPT (Nixtla), Chronos (Amazon)**:TS foundation models
- **etc...**

Prophet:

- GAM (general additive model) w. flexible trend + seasonality
- bayesian estimation
- trend switch points selection with L1 regularization (sparse priors)
- seasonality by Fourier series, also with L1 regularization
- requires us to change how updating works – keep forecast until reestimate

Figure 17: **Prophet:** Returns for 4000 period Prophet run with L1 regularization but not CV. Agents who refit the model keep the same forecast until they refit.



- Summary from Illustrative Simulations
  - Dynamic regularization (selection of penalty by CV) by agents **can** be a universal driver of fat tails and clustered volatility. Akin to dynamic model selection.
  - Effects vary across ML methods under endogenous dynamics.

8. Calibration

- Combination of manual calibration and Franke and Westerhoff (2011) MSM approach
  - Start with latin hypercube samples.
  - Match moments via Nelder-Mead algorithm to minimize a weighted quadratic distance measure.
  - Moments include measures of autocorrelation, and fat tails of returns as well as long memory in volatility.
  - Calibrated parameters include learning rate, memory, dividend shock variance, bond return.

- Some Details:
  - 9 Moments:
    - ★ mean of absolute returns (volatility)
    - ★ first order AC of raw returns
    - ★ ACs of absolute returns for lags 1,5, 10, 25, 50, 100) (long memory in volatility)
    - ★ Hill estimator for tail index of absolute returns (fat tails of returns)
  - Empirical moments based on returns from daily S&P prices from 1991-2021 (approx 7700 days)
  - Distance measure: $J = (m^{sim} - m^{emp})' W (m^{sim} - m^{emp})$
    - ★ where m is 9x1 and W is a 9x9 weighting matrix: inverse of estimated variance-covariance matrix of m^{emp} by bootstrap
  - Calibrated parameters include:
    - ★ learning rate
    - ★ memory
    - ★ dividend shock variance
    - ★ ML algorithm specific parameters

- Some Preliminary Calibration Results:
  - Lasso with CV tends to favor excessive fat tails and persistence in volatility.
  - Dynamic penalty selection matters for this. Lasso without CV and OLS underperform on these measures.
  - Tree methods and Prophet tend to underperform on these measures generally: both for no CV or (for tree methods) CV with broad allowed penalty range. Weak capacity for extrapolating trends out of sample.
  - Some weak convergence properties – particularly with Prophet (e.g., at higher pupdate). Also for OLS, NN and Lasso at low memory – not reflected in J.

- Preliminary Calibration Findings:

Table 1: Selected Statistical Features of Returns: Empirical (S&P) and Simulations

	ACF 1 Abs Return	ACF 5 Abs Return	ACF 10 Abs Return	ACF 25 Abs Return	ACF 50 Abs Return	ACF 100 Abs Return	Hill Estimator Abs Ret	J Value
S&P	0.2714	0.3216	0.2765	0.1854	0.1292	0.0788	2.7448	—
OLS1	0.3825	0.14005	0.1208	0.0917	-0.0219	-0.0210	3.597	337.6
OLS2	0.3892	0.2438	0.24215	0.1525	0.0427	0.0134	3.479	180.5
Lasso1	0.5843	0.4564	0.4132	0.3520	0.2795	0.1968	2.134	235.1
Lasso2	0.4993	0.3627	0.3408	0.2190	0.0392	-0.0317	3.076	223.9
Lasso3	0.5514	0.4480	0.3856	0.2146	0.0618	-0.0381	2.1565	196.7
NN2	0.3600	0.1078	0.1271	0.075	0.0451	0.0246	3.3682	241.1
LGBM1	0.3867	0.0836	0.0659	0.0876	0.0536	0.0306	4.333	599.0
Prophet2	0.0899	0.1289	0.1139	0.0816	0.0819	0.1398	3.779	339.3

By method and method parameters. Based on 5000 round runs except NN 2000 rds.
OLS2: fewer predictors; Lasso2 fixed penalty (no CV); Lasso3 lower memory

- A few observations:
  - Overall fits are somewhat weak due to first order AC and overall volatility. Weights put low value on Hill. J values are quite sensitive to random seeds.
  - We are particularly interested in fat tails (Hill) and slow decay of volatility (long memory). Lasso tends to generate too much fat tailedness (under CV), OLS (no CV), LightGBM (under CV) and Prophet (no CV) too little. NN still tbd!
  - Dynamic selection of penalty **can** act as universal driver of intermittency, clustered volatility, fat tails. **However** it may be too strong (Lasso) or may not be chosen by the agents via automated CV (LGBM). Behavioral question.
  - A method (e.g., GBTs or NNs) that does well at forecasting **exogenous** financial time series (M5, M6, Gu, etc.) may have little empirical content in an **endogenous** setting with market feedback. E.g., tree methods are poor at extrapolating forecasts outside of the historical training data, so complex dynamics may not emerge in the market. Similarly with Prophet which relies heavily on trend and stable calendar effects.
  - OLS and Prophet exhibit weak convergence properties and tend to drift far from the stationary equilibrium.

Figure 18: **S&P500:** Daily prices for S&P500 1991-2021

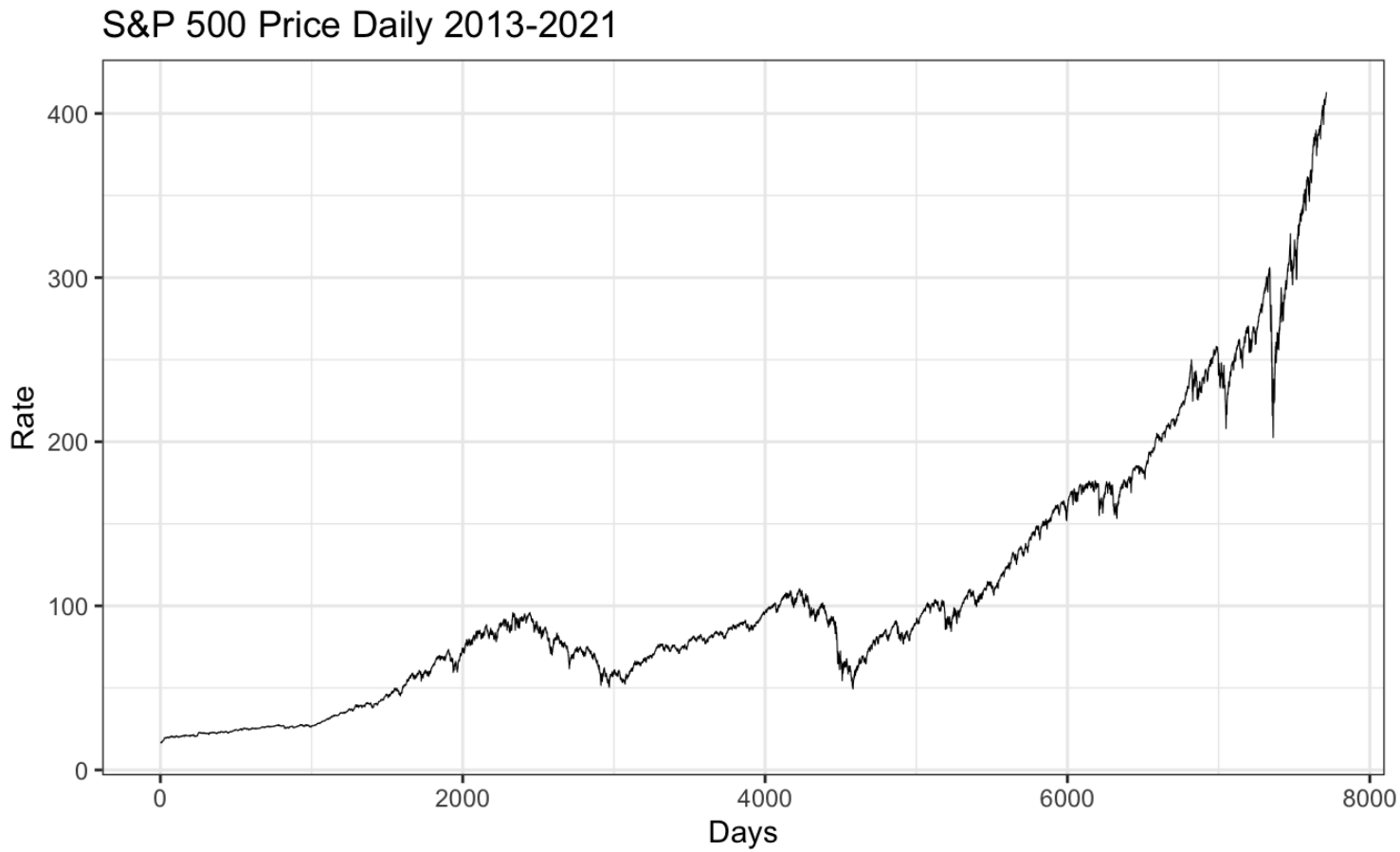
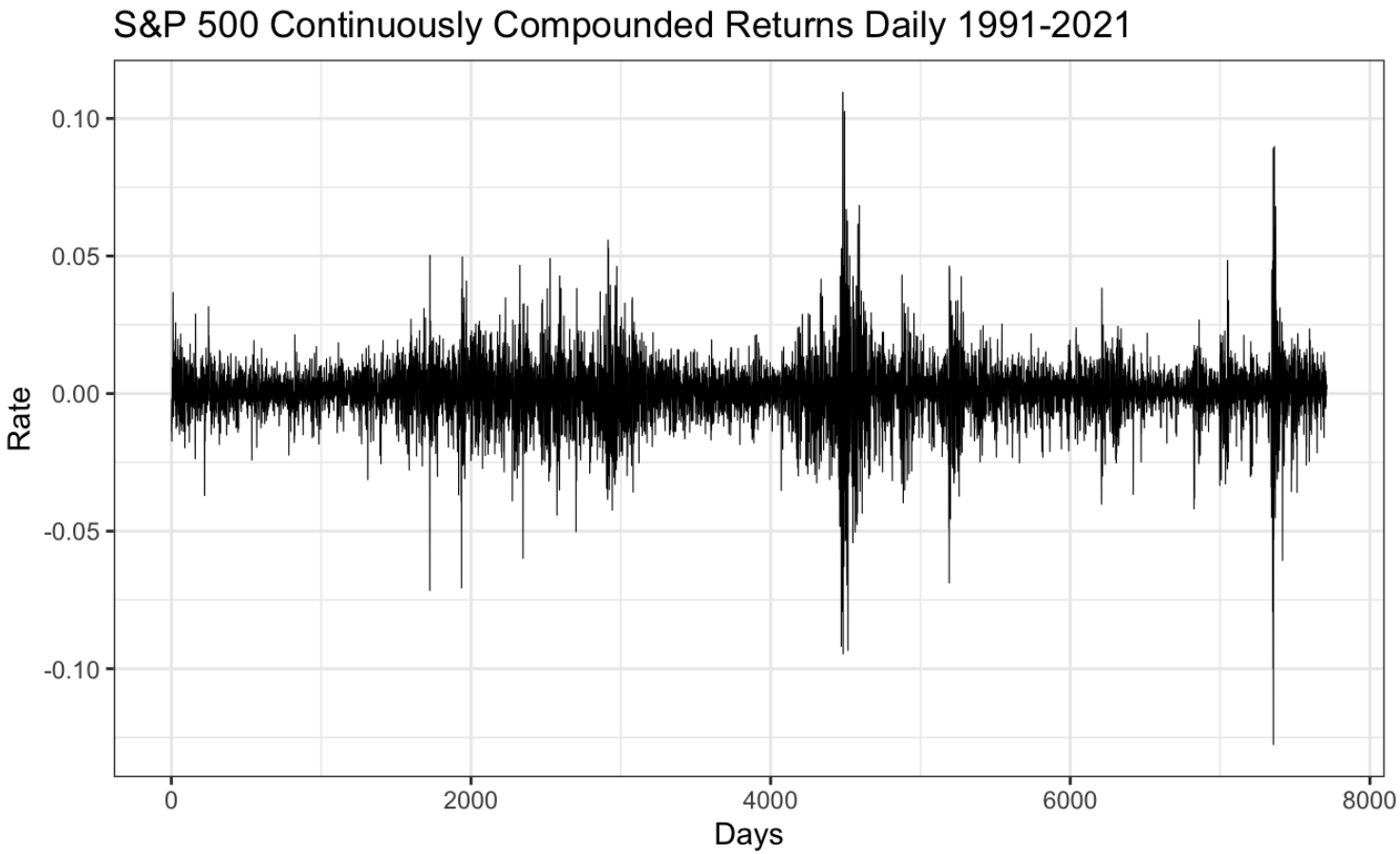


Figure 19: **S&P500:** Daily returns for S&P500 1991-2021



OLS1 spec has 2 lags 2 powers pupdate=.5 memory=100 lags=powers=2 shockrange=0.5; 5000 rds for all of these **OLS2** has lags=powers=1 (better fit); **Lasso1** has memory=100 pupdate=.2 shockrange 1.5 (need large shock range to get mean abs rets large enough); **Lasso2** drops CV on penalty (uses an average value); **Lasso3** has mem50 pup0.1 shkrg0.85 **LGBM** has CV on penalties (0,.01,.1,1,3) - 3 rarely gets selected and isn't large enough to make much diff, if add 5, never selected, so never get much intermittency unless force CV on large values of penalty **NN2** 20 nodes, m100, pup0.2, shkrg0.5, CV 0 0.01, 0.02, 0.03, 0.05, 0.07, 0.1 only, 2000 rds only, DF, would select larger CV penalties if allowed but already well above necessary **Prophet2** m200, pup0.1, shockrange 0.85 DF