ABIDES – Agent Based Interactive Discrete Event Simulation

- https://github.com/jpmorganchase/abides-jpmc-public
- NASDAQ-like exchange agent that mains limit order book
- FIFO order matching
- Gives a selection of market agents
  - **abides-jpmc-public**/abides-markets/abides_markets/agents/
  - Can write your own (rule-based or learning)
- Agents are combined into configurations that comprise the markets
  - **abides-jpmc-public**/abides-markets/abides_markets/**configs**/
  - Can write your own

Noise agents

- Arrive to the market randomly, trade on demand
- **abides-jpmc-public**/abidesmarkets/abides_markets/agents/**noise_agent.py**
-

```
bid, bid_vol, ask, ask_vol = self.get_known_bid_ask(self.symbol)

if self.order_size_model is not None:
    self.size = self.order_size_model.sample(random_state=self.random_state)

if self.size > 0:
    if buy_indicator == 1 and ask:
        self.place_limit_order(self.symbol, self.size, Side.BID, ask)
    elif not buy_indicator and bid:
        self.place_limit_order(self.symbol, self.size, Side.ASK, bid)
```

Value Agents

- Introduce the concept of **fundamental** price – represents the knowledge of the outside world (earnings, macro events, etc.)
  - Can be any time series – e.g., mean reverting, historical
- **Value** agents act upon the knowledge of fundamental – buy if asset is cheap relative to fundamental and sell if it is expensive
- **abides-jpmc-public**/abides-markets/abides_markets/agents/**value_agent.py**

Value Agents

```python
mid = int((ask + bid) / 2)
spread = abs(ask - bid)

if self.random_state.rand() < self.percent_aggr:
    adjust_int = 0
else:
    adjust_int = self.random_state.randint(
        0, min(9223372036854775807 - 1, self.depth_spread * spread)
    )
    # adjustment to the limit price, allowed to post inside the spread
    # or deeper in the book as a passive order to maximize surplus

if r_T < mid:
    # fundamental belief that price will go down, place a sell order
    buy = False
    p = (
        bid + adjust_int
    )  # submit a market order to sell, limit order inside the spread or deeper in the book
elif r_T >= mid:
    # fundamental belief that price will go up, buy order
    buy = True
    p = (
        ask - adjust_int
    )  # submit a market order to buy, a limit order inside the spread or deeper in the book
```

Momentum Agents

- Act on observed LOB trends
- **abides-jpmc-public**/abides-markets/abides_markets/agents/examples/**momentum_agent.py**

Momentum Agents

```
if bid and ask:
    self.mid_list.append((bid + ask) / 2)
    if len(self.mid_list) > 20:
        self.avg_20_list.append(
            MomentumAgent.ma(self.mid_list, n=20)[-1].round(2)
        )
    if len(self.mid_list) > 50:
        self.avg_50_list.append(
            MomentumAgent.ma(self.mid_list, n=50)[-1].round(2)
        )
    if len(self.avg_20_list) > 0 and len(self.avg_50_list) > 0:
        if self.order_size_model is not None:
            self.size = self.order_size_model.sample(
                random_state=self.random_state
            )

        if self.size > 0:
            if self.avg_20_list[-1] >= self.avg_50_list[-1]:
                self.place_limit_order(
                    self.symbol,
                    quantity=self.size,
                    side=Side.BID,
                    limit_price=ask,
                )
            else:
                self.place_limit_order(
                    self.symbol,
                    quantity=self.size,
                    side=Side.ASK,
                    limit_price=bid,
                )
```

Market Makers

- Posts on both sides on LOB to satisfy regulatory constraints
- Adjusts for liquidity dropouts
- **abides-jpmc-public**/abides-markets/abides_markets/agents/market_makers/**adaptive_market_maker_agent.py**

```
lowest_bid = highest_bid - ((self.num_ticks - 1) * self.tick_size)
highest_ask = lowest_ask + ((self.num_ticks - 1) * self.tick_size)

bids_to_place = [
    price
    for price in range(lowest_bid, highest_bid + self.tick_size, self.tick_size)
]
asks_to_place = [
    price
    for price in range(lowest_ask, highest_ask + self.tick_size, self.tick_size)
]
```

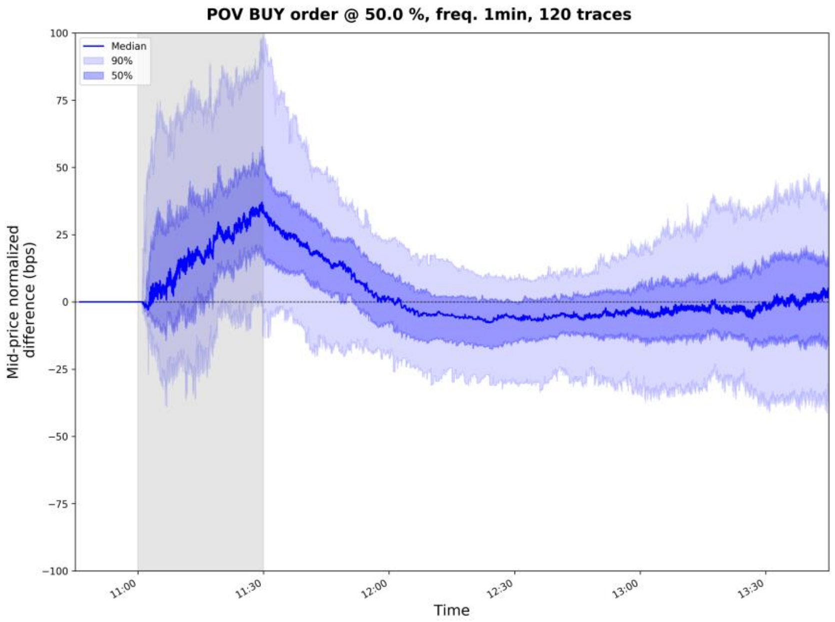
Market Configurations

- **abides-jpmc-public**/abides-markets/abides_markets/**configs/**
- Examples:
  - RMSC03: 1 Exchange Agent, 1 POV Market Maker Agent, 100 Value Agents, 25 Momentum Agents, 5000 Noise Agents
  - RMSC04: 1 Exchange Agent, 2 Market Maker Agents, 102 Value Agents, 12 Momentum Agents, 1000 Noise Agents
- Simulated a market that corresponds to a configuration:

```
$ abides abides-markets/abides_markets/configs/rmsc04.py --end_time "10:00:00"
```

Market Impact Configuration

- Multi-agent simulations allow to simulate **market impact** of trading:
  - **Execution** agent is trading by lifting limit order book layers; **momentum** agents amplify the impact of execution; **value** agents push price back towards the fundamental



Notebook Example

- **abides-jpmc-public/notebooks/demo_ABIDES-Markets.ipynb**

- Each agent maintains a log
  - Exchange agent - markets log (i.e., midprices)

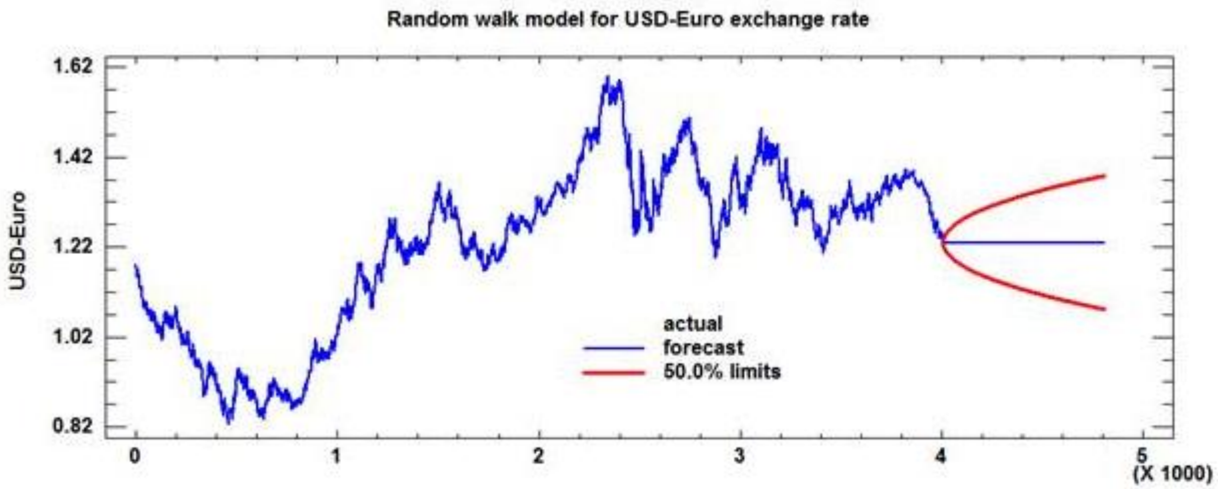
Gym

- https://github.com/jpmorganchase/abides-jpmc-public/tree/main/abides-gym

Time Series Models and Applications in Financial Markets

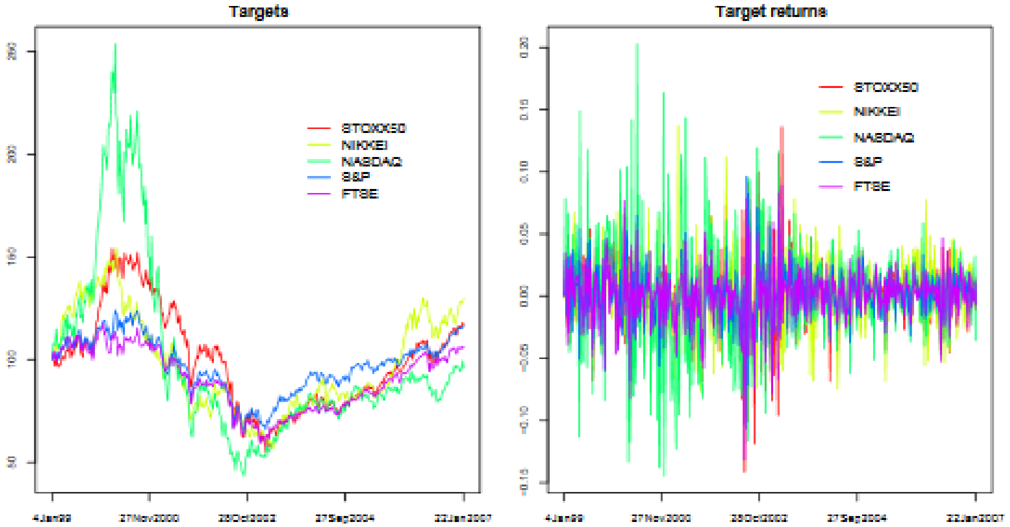
What are time series models?

- **Time series** is a time-oriented or chronological sequence of observations on a variable of interest
- Time series models employ the statistical properties of **historical** data to specify a formal model and then estimate the unknown parameters of this model



Stationary Time Series

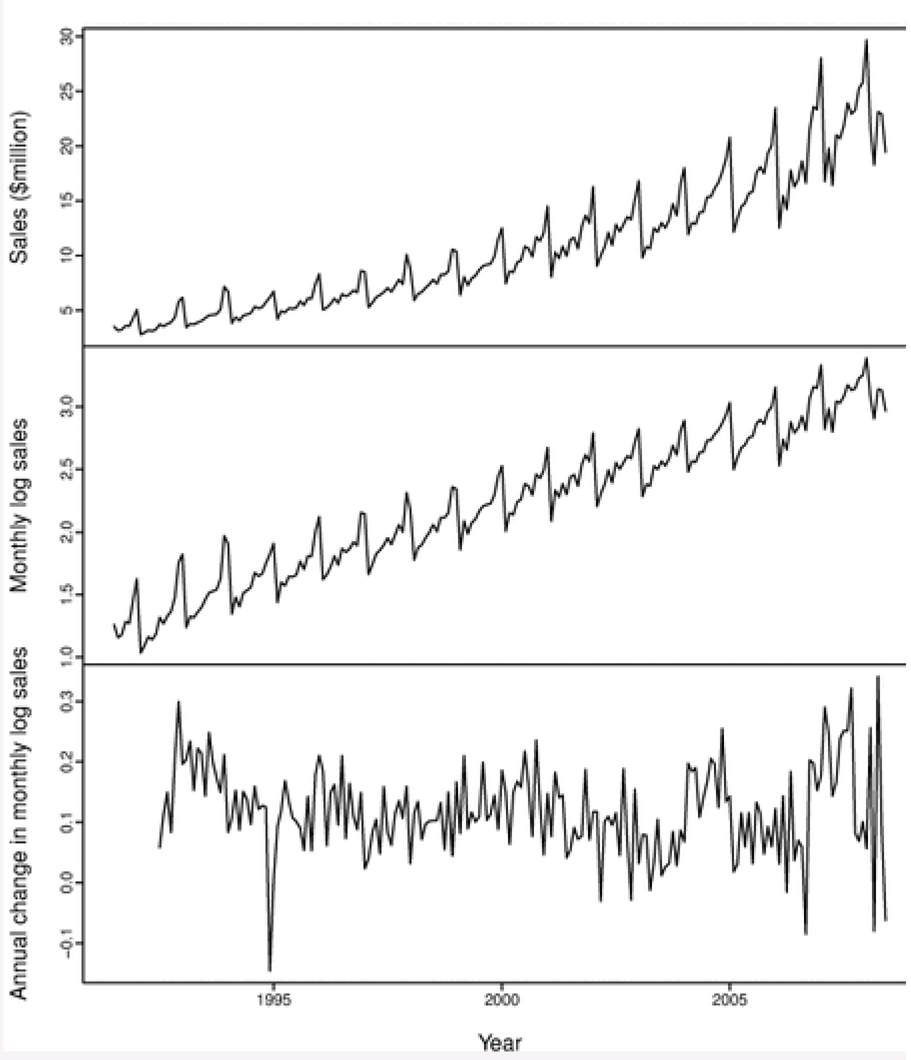
- Time series is called **stationary** if its properties are not affected by a change in the time origin, i.e. if joint probability distribution is exactly the same for $y_t, y_{t+1},\ldots, y_{t+n}$ and $y_{t+k}, y_{t+k+1},\ldots, y_{t+k+n}$
- A **stationary** time series is one whose statistical properties such as mean, variance, autocorrelation, etc. are all constant over time



- What if time series is **not stationary**?
- Non-linear trends
  - Data transformations (*log* y_t, $sqrt(y_t)$, y_t^2)
- Linear (polynomial) trends
  - Finance: forecast returns
  - Differencing ($x_t = at+b+\varepsilon_t$, $x_t - x_{t-1} \sim \varepsilon_t - \varepsilon_{t-1} + a$)
- Seasonal trends
  - Example: forecast of energy derivative prices based on weather patterns
  - Fit trigonometric model (eg., $A+B\ Cos\ wt+C\ Sin\ wt$)
  - Deseason (eg. $y_t = S_t + T_t + \varepsilon_t$, S_t- seasonal component, T_t – linear component, ε_t- noise)

Example: making time series stationary

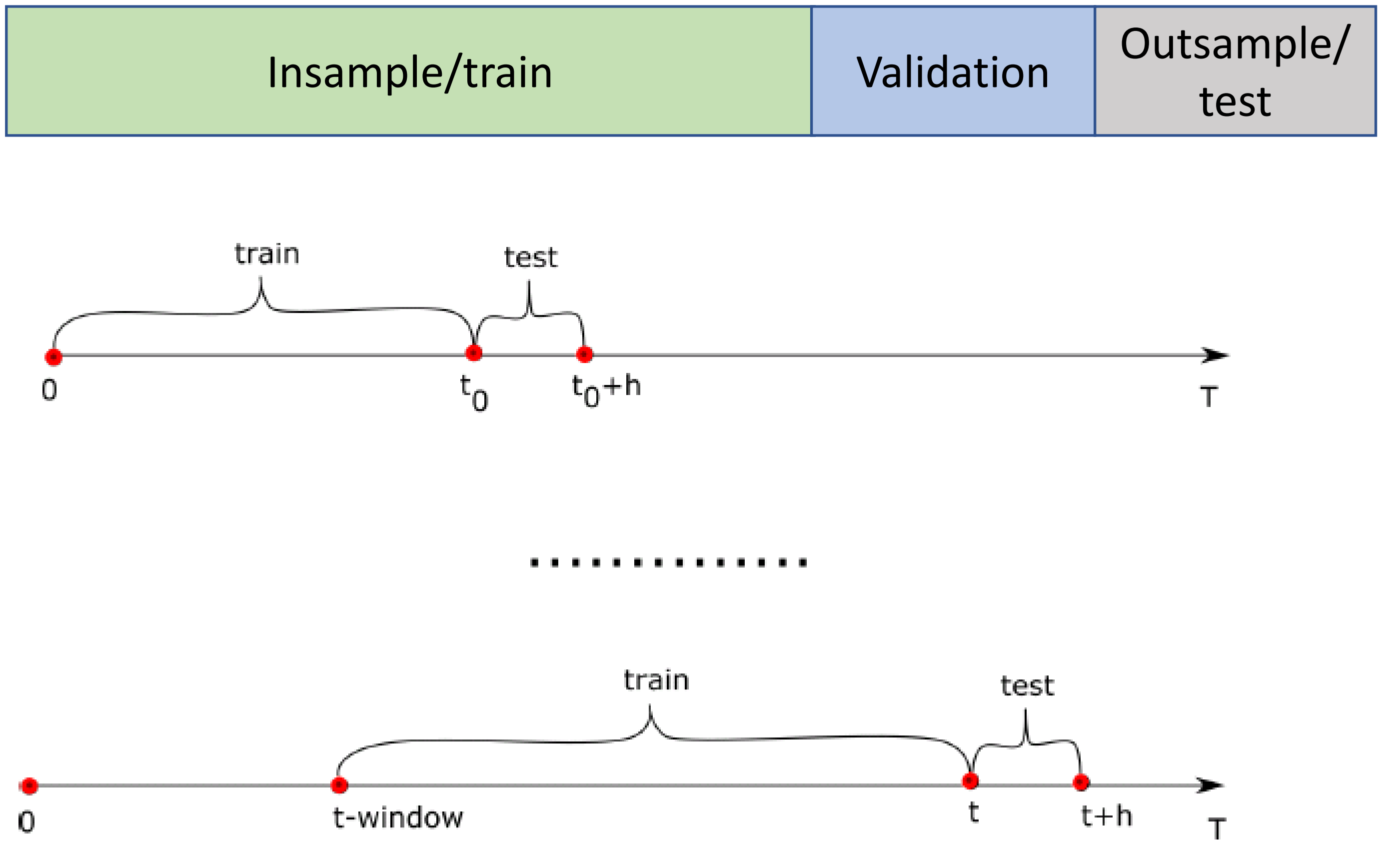
- Example: anti-allergy drug sales
- Nonlinear growth -> log data transformation
- Seasonal pattern and linear trend -> deseasoning and first differencing



Forecasting: General Methodology

- Identify candidate predictor time series x_{1t}, x_{2t}, .. x_{nt}
- Plot predictor time series and visually determine the basic features
  - Outliers
  - Non-linear transformations
  - Trends/seasonality
- Data-splitting
  - **Insample** dataset: train multiple models
  - **Validation** dataset: select the best model
  - **Outsample** dataset: predict and analyze performance
- Forecast time series $\tilde{y}_t$

Practical guidelines



Forecast quality

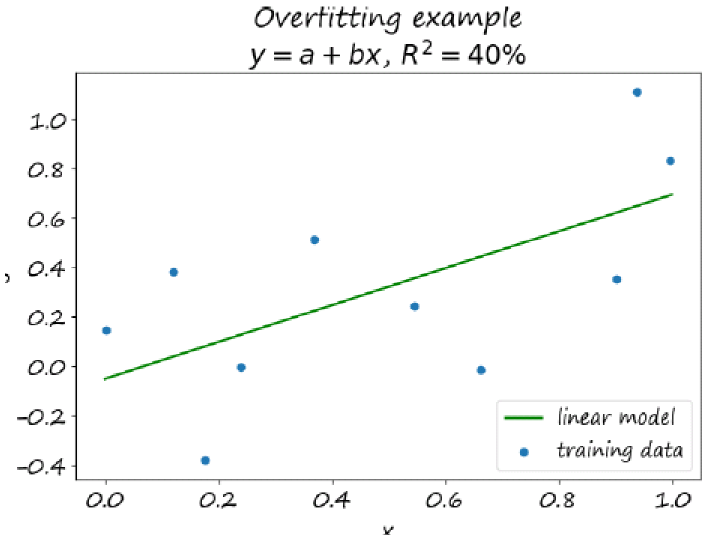
- How to evaluate forecast quality?
  - $\tilde{y}_t = f(x_{1t}, x_{2t}, .., x_{nt})$ – forecast
  - $MSE = \sum_{t=1}^{T}(y_t - \tilde{y}_t)^2$ - mean squared error
    - linear regression fit minimizes MSE
  - $R^2 = 1 - \frac{MSE}{\sum_{t=1}^{T}(y_t - \hat{y})^2}$
    - $\hat{y} = \frac{1}{T}\sum_{t=1}^{T} y_t$
    - how well predictor variables explain variance in y_t
    - will increase on insample data when the number of predictor variables is large (**overfitting**)

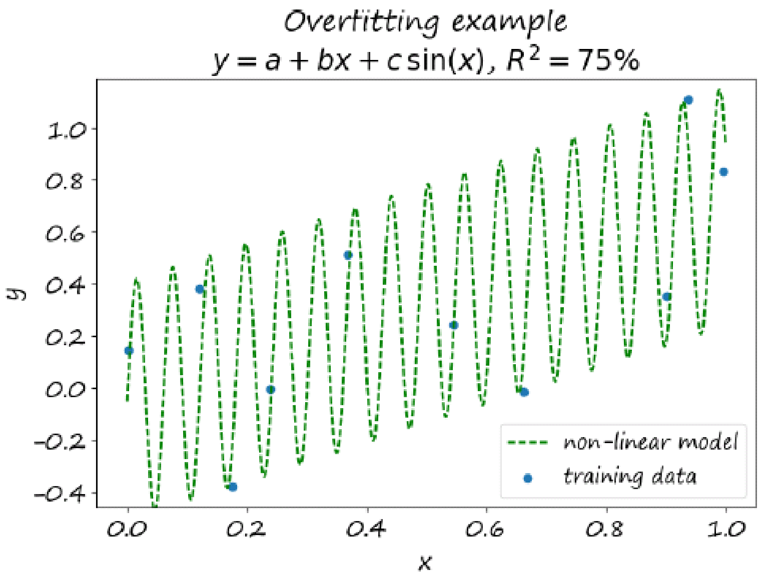
Overfitting

- Overfitting ~ fitting the noise
- Underfitting ~ not capturing enough complexity of the data
- General rules:
  - choose forecast of the simplest possible functional form
  - choose forecast with the smallest number of predictor variables
- Good value of R^2?
  - It depends!
  - R^2 on insample data – **fit** quality
  - R^2 on outsample data – **prediction** quality
  - Consider comparing MSE to **baseline** model on outsample data

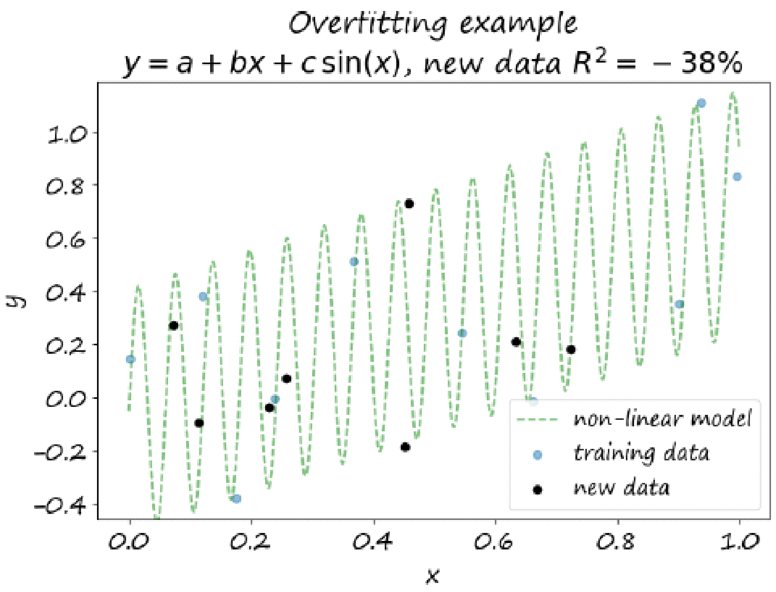
Overfitting

Insample

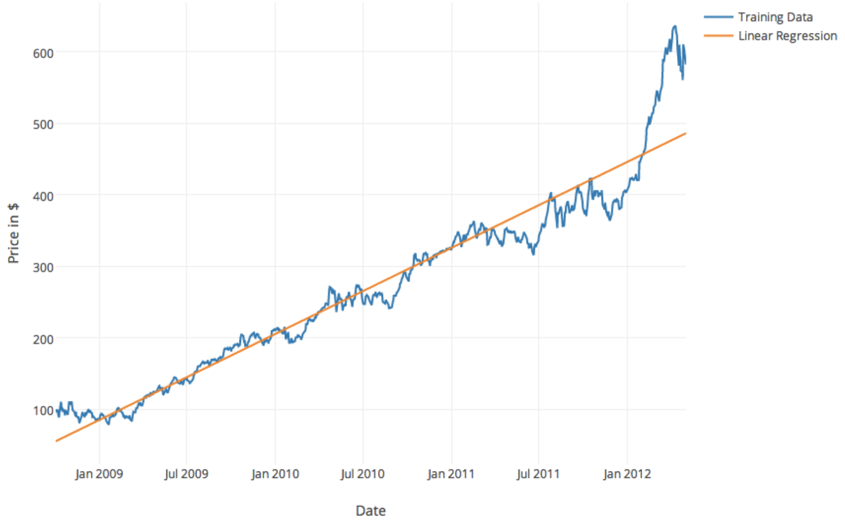




Outsample



Forecasting by linear regression



- Single or multiple predictors
  $Y = a_1X_1 + a_2X_2 + .. + a_nX_n + b + \varepsilon$
- Fit coefficients to a_i to minimize MSE and compute standard errors SE associated with a_i

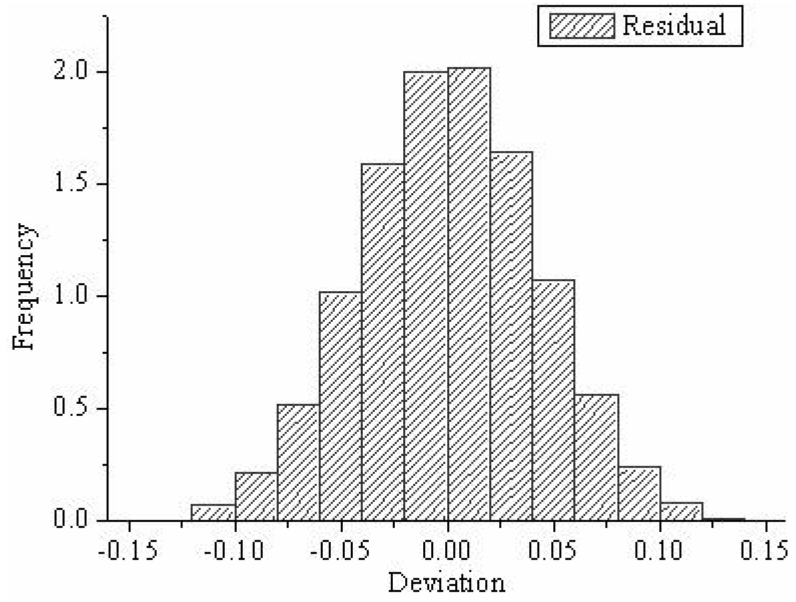
- **Are all *n* predictors useful in predicting the response?**

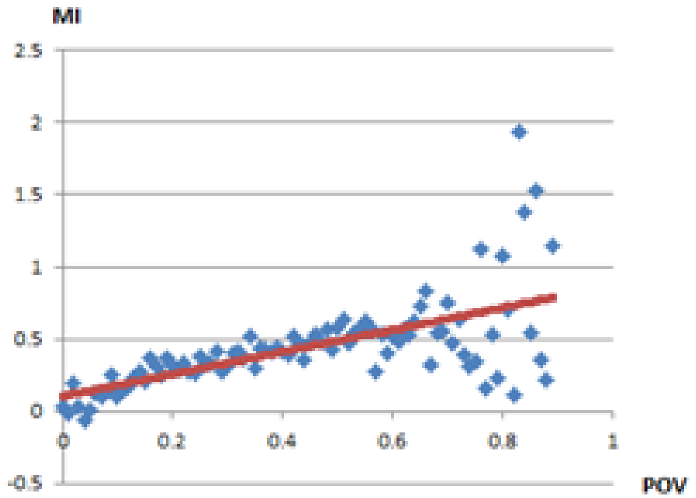
1. Variable selection
   - Forward selection – add predictors that maximize R^2
   - Backward selection – remove predictors with largest *p*-value
     - *t*-value = $a_i/SE(a_i)$ – number of standard deviation that a_i is away from 0
     - *p*-value = probability of observing a value equal to *t*-value or larger given $a_i=0$
     - remove variables with high *p*-values
   - LASSO family
2. Collinearity
   - Two or more predictor variables are closely related to each other
     - Combine variables (eg., credit score + credit limit = credit worthiness)
   - Variance inflation factor $VIF(i)= 1/(1-R^2_{Xi|X-i})$
     - Regress X_i onto other predictors
     - When VIF(i) high (>=10):
       - Drop variable *i*

Potential problems

Check **residuals** (ideally normal i.i.d.):

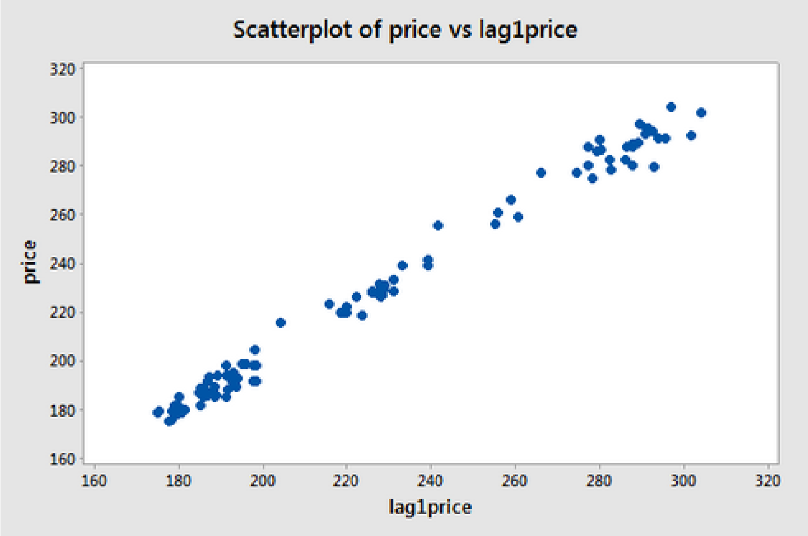
- Non-linearity of response-predictor relationships
  - Log(X), sqrt(X), X^2, sin(X) etc.
  - Neural networks
- Correlation of error terms
  - Very common in time series data (residuals will look like time series themselves)
  - Autoregressive process (AR(n))
- Non-constant variance of error terms
  - Weighted least squares
    - *i*th response has average of n_i observations with variance δ_i
    - Weight of *i*th response is inversely proportional to δ_i

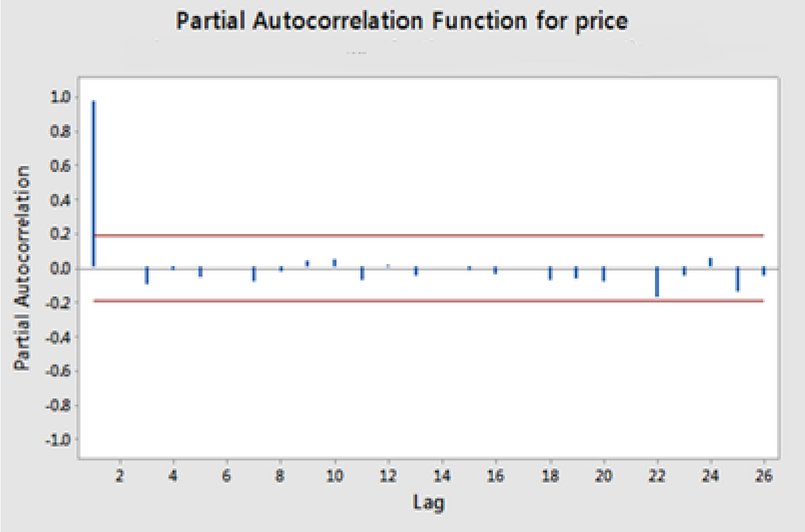




Autoregressive models (AR)

- Example: EOD price forecast, daily volume forecast
- $y_t = a+by_{t-1} + \varepsilon_t$, lag $k=1$
- $y_t = a+by_{t-1} + cy_{t-2}+ \varepsilon_t$, lag $k=2$
- Use partial autocorrelation function (PACF) is useful to determine lag
    - PACF – the amount of correlation with each lag that is not accounted by more recent lags

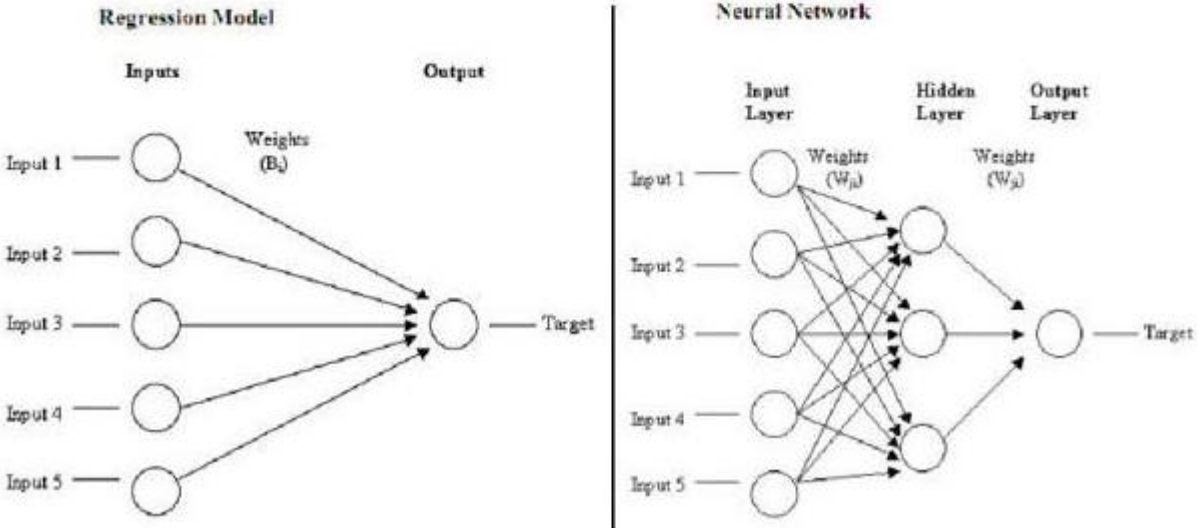




Forecasting by smoothing

- Smoothing is a technique that separates signal from noise
- Examples:
  - Average $\tilde{y}_{T+1} = (y_T + y_{T-1} + \ldots + y_{T-N+1})/N$
  - Median $\tilde{y}_{T+1} = median(y_T, y_{T-1}, \ldots ,y_{T-N+1})$
  - Exponential moving average $\tilde{y}_{T+1} = \lambda\, y_T + (1- \lambda)\, \tilde{y}_T$

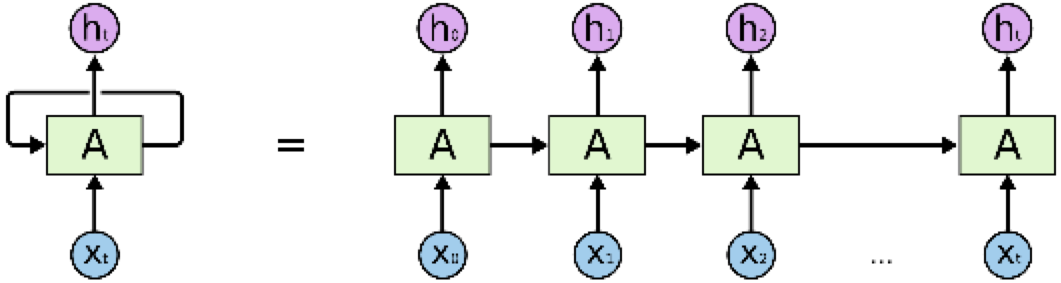
Neural Networks (NNs) and Forecasting

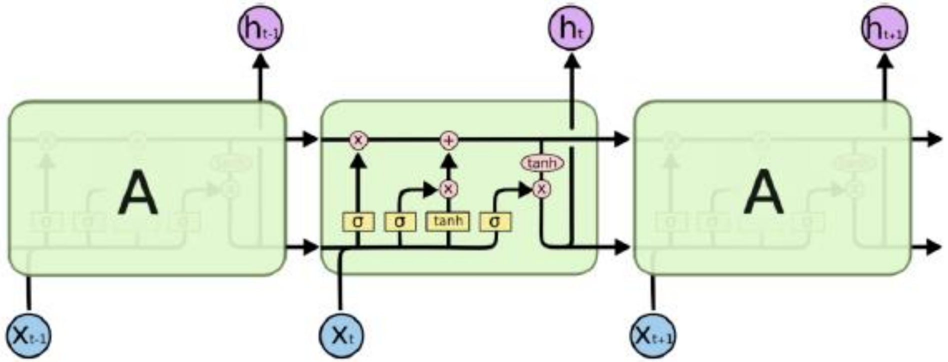


- Advantage:
  - By universal approximation theorem, can theoretically approximate almost any function!
  - Can capture complex non-linear dependencies
  - Scale favorably with large amounts of data
  - Very effective for language and image processing
- Disadvantages:
  - Expensive to train
  - Produces “black box” solution
  - Overfitting issues
- When it comes to time series, can be very effective too, but compare to simpler baseline

NN Architectures for Time Series

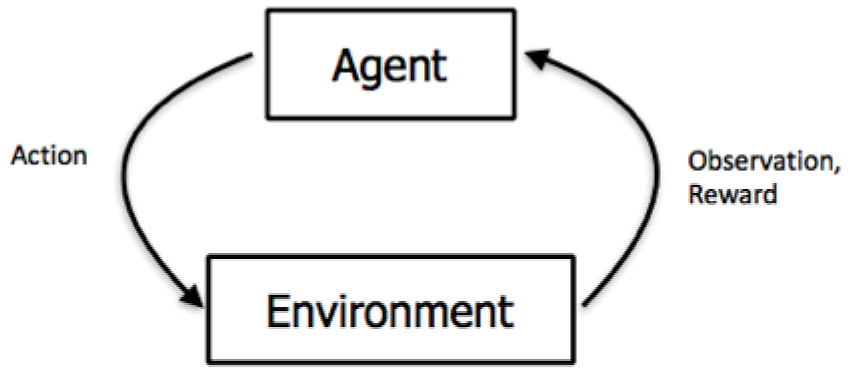
- Time Delay Neural Networks
  - Window of inputs
  - Time lag determined by modeler
  - Number of weights grows linearly with inputs
- Recurrent Neural Networks
  - Contains internal hidden layers to keep track of the past – especially designed for modeling autoregression
  - Use LSTMs (Long Short-Term Memory) – memory cells for modeling lags of unknown duration





Reinforcement Learning (RL)

- Forecast **predicts** future events
  - passive, does not interact with the environment
- RL agent **optimizes** future outcomes
  - active agent



- Environment: Markov Decision Process (MDP)
  - State space: environment and agent states
  - Action space
  - Transition probabilities from one state to another under certain action
    - Model-free vs. model-based
  - Reward for transition from one state to another given certain action
- Objective: maximize future **cumulative** rewards
  - Sequential decision making

Bayesian Methods and Forecasting

- Useful when little or no information is available at the time forecast is required or observations are expensive
  - Online experiment design (drug tests)
- Estimate distribution of **unobserved** data given observed data
- Effective ways to calculate using Gaussian distribution
  - If one needs to estimate μ of normal distribution and б^2 is known
  - Assume normal prior for μ with parameters μ$_o$, б^2_o

$$E(\mu' \mid x) = \frac{\sigma^2\mu + \sigma_0^2 x}{\sigma^2 + \sigma_0^2}$$

$$\mathrm{Var}(\mu' \mid x) = \frac{\sigma^2\sigma_0^2}{\sigma^2 + \sigma_0^2}$$

Bayesian Methods Work