Introduction to Time Series

Benjamin S. Skrainka

March 10, 2016

Objectives

Today's objectives:

- Define key time series concepts
- Use graphical tools to analyze time series data
- Use Box-Jenkins work-flow to estimate an ARIMA model
- Use Python's StatsModels to train and evaluate ARIMA models
- Describe Exponential Smoothing (ETS) model

Caveat: we focus on forecasting the mean and not quantiles.

Agenda

Today's agenda:

1. Define key time series concepts and properties
2. The ARIMA model: concepts & terminology
3. Estimating ARIMA models using Box-Jenkins
4. Practical advice
5. Describe ETS model

References

A couple helpful references, arranged by increasing difficulty:

- Hyndman & Athanasopoulos: Forecasting: principles and practice
- Enders: Applied Econometric Time Series
- Hamilton: Time Series Analysis
- Elliott & Timmermann: Economic forecasting

A little religion: Python vs. R

In most cases, you can use Python or R, depending on your preference:

- In Python, use `StatsModels` + `Pandas`:
  - Pandas: to manipulate data and dates
  - StatsModels: to estimate core time series models
- In R, Hyndman's `forecast` package is outstanding:
  - Use `lubridate` to manipulate dates
  - For serious forecasting, R is vastly superior
  - Only serious option for ETS
- Galvanize is a Python shop, so . . . we will use Python

Introduction

Time series data

Time series data is a sequence of observations of some quantity of interest, which are collected over time, such as:

- GDP
- The price of toilet paper or a stock
- Demand for a good
- Unemployment
- Web traffic (clicks, logins, posts, etc.)

Definition

We assume a time series, $\{y_t\}$, has the following properties:

- y_t is an observation of the level of y at time t
- $\{y_t\}$ is time series, i.e., the collection of observations:
  - May extend back to $t = 0$ or $t = -\infty$, depending on the problem.
  - E.g., $t \in \{0, ..., T\}$

Assumptions

We assume:

- Discrete time:
  - Sampling at regular intervals
  - . . . even if process is continuous
- Evenly spaced observations
- No missing observations

Caveat: only one observation?

Time series are hard to model because we only observe one realization of the path of the process:

- Often have limited data
- Must impose structure – such as assumptions of about correlation – in order to model
- Must project beyond support of the data.

Furthermore, in practice executives often ask for forecasts to CYA...

Components of a time series

Think of a time series as consisting of several different components:

- Trend
- Seasonal
- Periodic
- Irregular

Can be additive or multiplicative

Example decomposition from Hyndman et al.

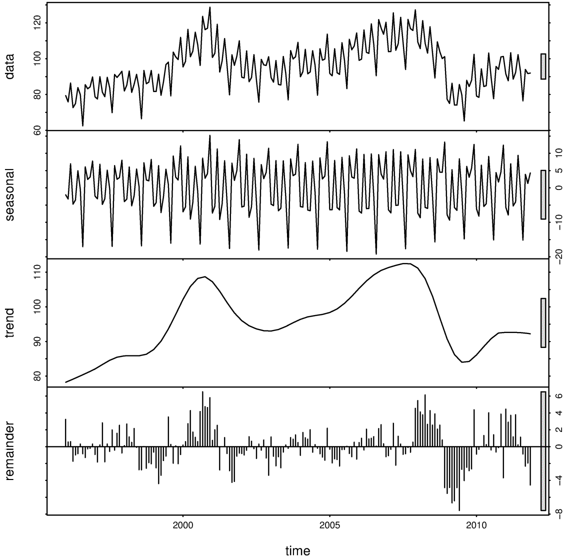

Figure 1:

Example time series from Hyndman et al.

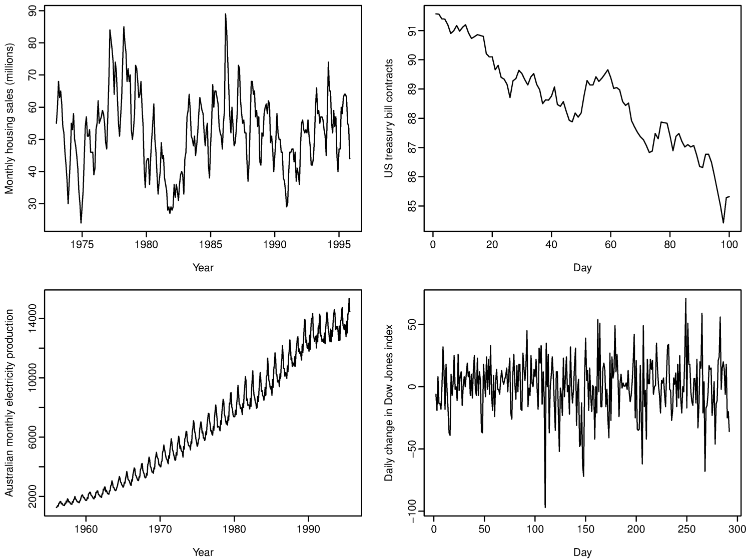

Figure 2:

Two popular models

Two common models:

- ARIMA(p,d,q):
  - A benchmark model
  - Captures key aspects of time series data
- Exponential smoothing (ETS):
  - Smooths out irregular shocks to model trend and seasonality
  - Updates forecast with linear combination of past forecast and current value
  - Also known as a "State space model"

Can also take a machine learning approach . . .

Notation

Some notation, following Hyndman:

- y_t: the level of some value of interest at time t
- ϵ_t: the value of a shock, ϵ, at time t
- $\hat{y}_{t+h|t}$ is the forecast for y_{t+h} based on the information available at time t

Lags

Often models use past values to predict future:

- AR(1): $y_t = \phi \cdot y_{t-1} + \epsilon_t$
- MA(1): $y_t = \mu + \epsilon_t + \psi \cdot \epsilon_{t-1}$
- Easier to write with lag operators:

$$\mathbb{L} : x_t \mapsto x_{t-1}$$

- With lag operators:
  - AR(1): $y_t = \phi \cdot \mathbb{L} y_t + \epsilon_t$
  - MA(1): $y_t = \mu + (1 + \psi \cdot \mathbb{L})\epsilon_t$

Concepts: basic statistics

First, we review some basic statistics:

- *expectation*:
  - $\mathbb{E}[g(x)] \equiv \int g(x) \cdot f(x) dx$
  - $g(x)$ is an arbitrary function
  - $f(x)$ is the probability density function
- *mean*:
  - A 'typical' value
  - $\mu(x) = \mathbb{E}[x]$
- *variance*:
  - A measure of volatility or the spread of a distribution
  - $\mathtt{Var}(x_t) = \mathbb{E}[(x_t - \mu(x_t)) * (x_t - \mu(x_t))^T]$
  - $\sigma^2(x_t) \equiv \mathtt{Var}(x_t)$
- *standard deviation*:
  - $\sigma(x_t) \equiv \sqrt{\mathtt{Var}(x_t)}$

Concepts: time series (1/2)

To understand persistence of a time series, examine:

- *autocovariance*:
  - How much a lag predicts a future value of a time series
  - $\texttt{acov}(x_t, x_{t-h}) \equiv \mathbb{E}[(x_t - \mu(x_t)) * (x_{t-h} - \mu(x_{t-h})))]$
  - Often written as $\gamma(s, t)$ or $\gamma(h)$ where $h = s - t$
- *autocorrelation*:
  - A dimensionless measure of the influence of one lag upon another
  - Helps determine which ARIMA model to use

$$\texttt{acorr}(x_t) = \frac{\texttt{acov}(x_t, x_{t+h})}{\sigma(x_t) \cdot \sigma(x_{t+h})}$$

  - Often written as $\rho(t) \equiv \gamma(t)/\gamma(0)$ for this case

Concepts: time series (2/2)

Special time series (easier to forecast):

- To forecast, need mean, variance, and correlation to be stable over time
- *strictly stationary*:
  - $\{x_t\}$ is strictly stationary if $f(x_1, ..., x_t) = f(x_{1+h}, ..., x_{t+h})$, $\forall h$
- *weakly stationary*
  - mean is constant for all periods: $\mu(x_t) = \mu(x_{t+h})$, $\forall h$
  - autocorrelation, $\rho(s, t)$, depends only on |s - t|
- *white noise*:
  - $\mathtt{acov}(x_t, x_{t+h}) = \mathtt{var}[x_t]$ iff $h = 0$ and 0 otherwise
  - is (weakly) stationary
  - is a key building block of time series models

Analog principles

Analog principle: replace expectations with sample averages when calculating statistics

- Intuition: the Weak Law of Large Numbers
- Examples:
  - Mean: $\mathbb{E}[x] \to \frac{1}{N}\sum_{i=1}^{N} x_i$
  - In general: $\mathbb{E}[g(x)] \to \frac{1}{N}\sum_{i=1}^{N} g(x_i)$
- Sometimes, we replace N with $N-1$ (e.g., for the variance):
  - So the statistic is *consistent*
  - E.g., $\mathbb{E}[\overline{x}] = \mathbb{E}[x_i] = \mu(x)$
  - I.e., the estimator is unbiased

The ARIMA model: concepts & terminology

ARIMA introduction

ARIMA is a benchmark model:

- ARIMA(p,d,q) consists of:
  - AR(p): persistence of history through AR terms
  - I(d): trend
  - MA(q): influence of past shocks through MA terms
- Can add higher order lags for seasonality
- If your fancy algorithm doesn't beat ARIMA, use ARIMA!

Terms: AR(p)

An AR(p) model captures the persistence of past *history*.

- AR(p) means *auto-regressive of order p*:

$$y_t = \mu + \phi_1 \cdot y_{t-1} + ... + \phi_p \cdot y_{t-p} + \epsilon_t$$

- Often, written with lag operators and polynomials:

$$\Phi(\mathbb{L}) \cdot y_t = \mu + \epsilon_t$$

- $\Phi(\cdot)$ is polynomial of order p:

$$\Phi(x) = 1 - \phi_1 \cdot x^1 - ... - \phi_p \cdot x^p$$

Terms: MA(q)

An MA(q) model captures the persistence of past *shocks*.

- MA(q) means *moving average of order q*:

$$y_t = \epsilon_t + \psi_1 \cdot \epsilon_{t-1} + ... + \psi_q \cdot \epsilon_{t-q}$$

- Often, written with lag operators and polynomials:

$$y_t = \Psi(\mathbb{L}) \cdot \epsilon_t$$

- $\Psi(\cdot)$ is polynomial of order q:

$$\Psi(x) = 1 + \psi_1 \cdot x^1 + ... + \psi_q \cdot x^q$$

$\Rightarrow$ Do not confuse with computing the moving average of $\{y_t\}$, which is often used to aggregate data.

Terms: I(d)

An I(d) model captures the non-stationary trend.

- I(d) means *integrated of order d*:
  $y_t = y_{t-1} + \mu + \epsilon_t$
- d is how many times you must difference the series so that it is stationary
- Usually, $d \in \{0, 1, 2\}$
- Differencing should remove the trend component
- Example: random walk (with drift)
- Compute differences with `np.diff(n=d)` or `pd.Series.diff(periods=d)` to turn ARIMA into ARMA.

ARIMA models

An ARIMA(p,d,q) is a general model which includes AR, I, and MA:

- AR(p): AR of order p
- I(d): I of order d
- MA(q): MA of order q

Remarks:

- AR, I, and/or MA may be missing from a general ARIMA model
- May also include seasonal components ... Specify as ARIMA(p,d,q)(P,D,Q)
- If $d = 0 \Rightarrow$ ARIMA becomes ARMA

Estimating ARIMA models using Box-Jenkins

Box-Jenkins methodology

Use Box-Jenkins's approach to fit an ARIMA model:

1. Exploratory data analysis (EDA):
    - plot time series, ACF, PACF
    - identify hypotheses, models, and data issues
    - aggregate to an appropriate grain
2. Fit model(s)
    - Difference until stationary
    - Test for a unit root (Augmented Dicky-Fuller (ADF))
    - Transform until variance is stable
3. Examine residuals: are they white noise?
4. Test and evaluate on out of sample data
5. Worry about:
    - structural breaks
    - forecasting for large h with limited data ⇒ need a "panel of experts"
    - seasonality, periodicity

Modeling flow chart from Hyndman et al.

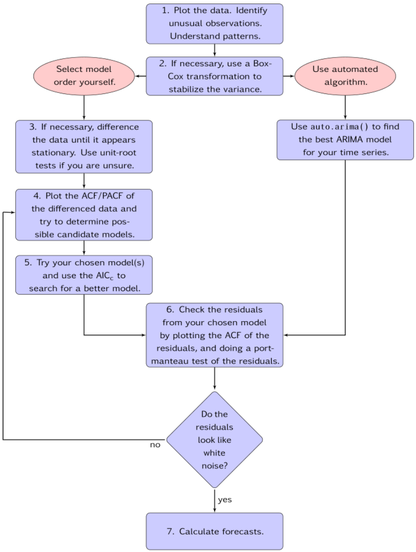

Figure 3:

Graphical tools

Plot data to develop understanding of data and possible models:

- Key diagnostic plots:
  - Plot time series, y_t, vs t
  - Plot autocorrelation function (ACF), i.e., $\rho(h)$ vs. h
  - Plot partial autocorrelation function (PACF)
- Repeat for first and second differences, if necessary:
  - Compute differences with `np.diff(n=d)` or `pd.Series.diff(periods=d)`
  - Transform series, if necessary, e.g, $y_t \to \log(y_t)$
  - Check stationarity: i.e., no trend and constant variance

Autocorrelation function (ACF)

Shows likely order of the *MA(q)* part of the ARIMA(p,d,q) model:

- Plots $\rho(h)$ vs. lags h
- Find largest significant spike
- Consider order q, where $q = \texttt{largest lag}$

```
import statsmodels.api as sm
data = sm.tsa.arma_generate_sample(ar=[ 0.7, 0.0, 0.3],
    ma=[0.2, -0.1], nsample=100)
sm.graphics.tsa.plot_acf(data, lags=28, alpha=0.05)
plt.show()
```

Partial autocorrelation function (PACF)

Shows likely order of the *AR(p)* part of the ARIMA(p,d,q) model:

- Plots partial autocorrelation vs. lags h
- Partial autocorrelation uses a regression method to compute effect of just a single lag but not intermediate lags like ACF
- Consider order p, where $p = \texttt{largest lag}$

```
import statsmodels.api as sm
data = sm.tsa.arma_generate_sample(ar=[ 0.7, 0.0, 0.3],
    ma=[0.2, -0.1], nsample=100)
sm.graphics.tsa.plot_pacf(data, lags=28, alpha=0.05)
plt.show()
```

Example: plotting series, ACF, and PACF (1/3)

You will do this all the time, so create a helper function:

```
def ts_diag_plot(data, lags=28):
    fig = plt.figure(figsize=(15,10))
    ax1 = fig.add_subplot(311)
    ax1.plot(data)
    ax1.set_title('y_t vs. t')
    ax2 = fig.add_subplot(312)
    sm.graphics.tsa.plot_acf(data, lags=lags, ax=ax2)
    ax3 = fig.add_subplot(313)
    sm.graphics.tsa.plot_pacf(data, lags=lags, ax=ax3)
    fig.show()
    return fig
```

Example: diagnostic plots (2/3)

```
from tsplot import ts_diag_plot

fake = sm.tsa.arma_generate_sample(ar=[ 0.7, 0.0, 0.3],
    ma=[0.2, -0.1], nsample=100)
fig = ts_diag_plot(fake)
```

Example: diagnostic plots (3/3)

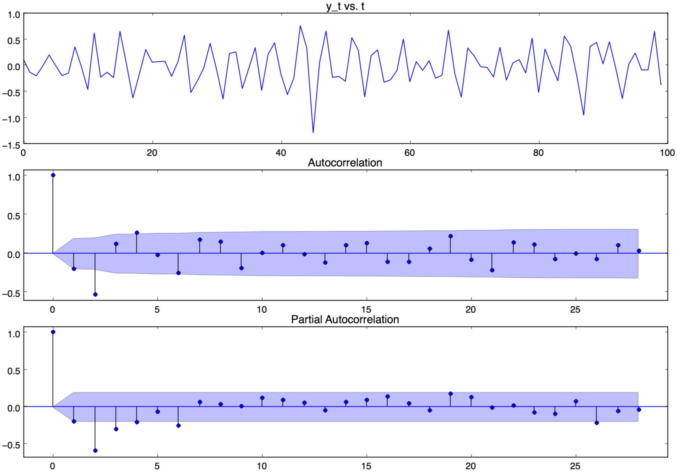

Figure 4:Three diagnostic plots

Practical advice

Questions to ask

Look at the time series plots and ask:

- Is it stationary?
- Is there a trend?
- Is the variance stable?
- Are there seasonal or periodic components?
- What AR and MA terms are likely present?
- Are there structural breaks in the data?
- Do I have enough data to forecast at horizon h?

Stabilizing the time series

You need to stabilize the time series before estimating a model:

- Transform data to stabilize variance:
  - $y_t \leftarrow \log(y_t)$
  - Verify via Box-Cox test
  - Verify by plotting
- Transform data so series is stationary:
  - Compute first or second difference
  - $y_t \leftarrow \Delta y_t$ or $y_t \leftarrow \Delta^2 y_t$
  - Verify by portmanteau test

Fit an ARIMA model

To fit a model:

- Split data into train set (earlier observations) and test set (later observations)
- To forecast at horizon h, should have at least $3 \times h$ observations to train plus h observations to test:
  - I.e., you cannot forecast demand in two years if you only have three months of data
  - If these conditions are violated, you need a ‘panel of experts’
  - More data is better, especially if seasonality is present
- To identify optimal order of model:
  - Examine ACF and PACF
  - Difference until stationary
  - Number of differences is order d for I(d)
  - Use `sm.tsa.arma_order_select_ic` to generate and compare several models
  - Use cross validation

Example: (1/2)

```
import statsmodels.api as sm
data = sm.datasets.macrodata.load_pandas()
df = data.data
df.index = pd.Index(
    sm.tsa.datetools.dates_from_range('1959Q1', '2009Q3'))
y = df.m1
X = df[['realgdp', 'cpi']]
model = sm.tsa.ARIMA(endog=y, order=[1,1,1])
# model2 = sm.tsa.ARIMA(endog=y, order=[1,1,1], exog=X)
results = model.fit()
results.summary()
```

Example: (2/2)

```
In [54]: results.summary()
Out[54]:
<class 'statsmodels.iolib.summary.Summary'>
"""
                             ARIMA Model Results
==============================================================================
Dep. Variable:                   D.m1   No. Observations:                  202
Model:                 ARIMA(1, 1, 1)   Log Likelihood                -759.253
Method:                       css-mle   S.D. of innovations             10.364
Date:                Wed, 01 Jul 2015   AIC                           1526.507
Time:                        11:17:55   BIC                           1539.740
Sample:                    06-30-1959   HQIC                          1531.861
                         - 09-30-2009
==============================================================================
                 coef    std err          z      P>|z|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
const          7.9682      2.595      3.071      0.002         2.882    13.054
ar.L1.D.m1     0.8290      0.061     13.521      0.000         0.709     0.949
ma.L1.D.m1    -0.3806      0.093     -4.090      0.000        -0.563    -0.198
                                    Roots
=============================================================================
                 Real           Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1             1.2063           +0.0000j           1.2063            0.0000
MA.1             2.6272           +0.0000j           2.6272            0.0000
-----------------------------------------------------------------------------
"""
```

Figure 5:Example: summary output from ARIMA model

Prediction intervals

A forecast of $\{y_t\}$ at time $t + h$ computes:

- $\hat{y}_{t+h|t}$, the expected mean of y_t at time $t + h$ conditional on the information available at t
- The *prediction interval*
  - Contains future realization of the mean y_{t+h} with probability $1 - \alpha$
  - Increases the further you forecast into the future
- **A prediction interval is not a confidence interval**:
  - A prediction interval contains the future realization of a random variable with $\Pr = 1 - \alpha$
  - A confidence interval contains the true value of a parameter with $\Pr = 1 - \alpha$
- See Hyndman's blog post for further discussion

Forecasting

Can use `results.forecast` to compute out of sample predictions:

- Use `alpha` to choose appropriate prediction interval, e.g., 80%, 90%, 95%, etc.
- Do not use the prediction interval to forecast quantiles of $\hat{y}_{t+h|t}$
- Note: documentation incorrectly refers to the *prediction interval* as the *confidence interval*
- Can supply (forecasted) value of exogenous predictors

```
>> y_hat, stderr, pred_int = results.forecast(steps=h,
    alpha=0.05)
```

Forecast: prediction intervals

Prediction plot includes a *prediction interval*:

- Contains future realization of y_{t+h} with probability $1 - \alpha$
- A prediction interval is not a confidence interval

```
results.plot_predict('2009Q3', '2014Q4', dynamic=True,
    plot_insample=True)
plt.show()
```

Example: prediction intervals

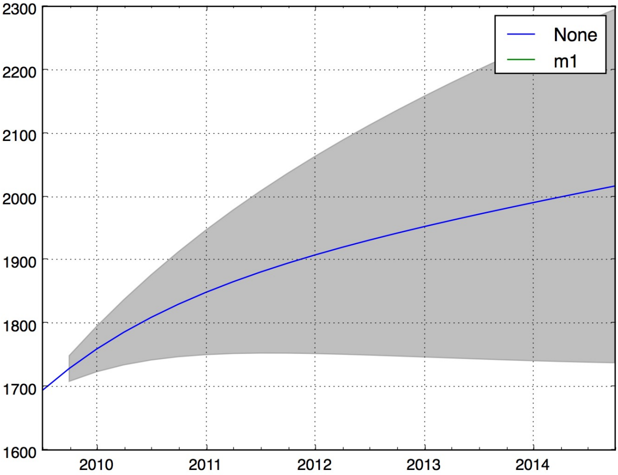

Figure 6:Prediction plot

Evaluate

"Trust, but verify":

- Check residuals are white noise:
  - Examine ACF & PACF
  - Compute portmanteau (Box-Pierce, Box-Ljung) test to see if residuals are correlated
- Check solver converged!
- Remember: simple models often outperform fancy models on new data
- Compare any forecast against the benchmark forecast
  - Choose a benchmark such as mean or random walk with drift
  - Fit model on training set and evaluate on test set
  - To compare multiple forecasts, use a sliding window

Common metrics

It is common to use several metrics for evaluation:

- *Root mean squared error*:

$$RMSE \equiv \sqrt{\frac{1}{H} \sum \left(y_{t+h} - \hat{y}_{t+h|t}\right)^2}$$

- *Mean absolute error*:

$$MAE \equiv \frac{1}{H} \sum \left|y_{t+h} - \hat{y}_{t+h|t}\right|$$

- *Mean absolute percentage error*:

$$MAPE \equiv \frac{1}{H} \sum \left|\frac{y_{t+h} - \hat{y}_{t+h|t}}{y_{t+h}}\right|$$

Model selection

Use information criterion to evaluate models:

- Several information criteria exist: AIC, **AICc**, BIC
  - Essentially, log-likelihood plus penalty for adding parameters
  - Measures fit vs. parsimony of model
  - Different criteria have different finite sample properties
- Choose model with lowest information criterion
- Especially helpful if you have limited data
- Popular, pre-ML method, but consider cross-validation if you have enough data

Tips & Tricks

Some hard won wisdom:

- Work at the appropriate level of aggregation (grain):
  - Don't use 5 minute resolution data to forecast at $h =$ one month
- Don't forecast beyond what the data will support
  - You should have $4 \times h$ amount of data to forecast at horizon h
- Err on the side of simplicity
- Or, take a machine learning approach:
  - Try a set of lags and differences plus other predictors
  - Use regularization and/or variable selection
  - See Taieb & Hyndman for an approach which uses boosting.

Advanced ARIMA techniques

For more complicated situations:

- Add Fourier terms to capture periodic behavior
- Add other covariates which can improve prediction
- Use a vector autoregressive integrated moving average model (VARIMA) to capture dynamics of a system of equations

Exponential smoothing (ETS) models

ETS introduction

Exponential smoothing models are a benchmark model:

- Robust performance
- Easy to explain to non-technical stakeholders
- Easy to estimate with limited computational resources
- Forecast well because of parsimony

The model

The model consists of smoothing equations for

- Forecast
- Level
- Trend (optional)
- Seasonality (optional)

Remarks:

- Can use either an additive or multiplicative specification
- Can use a state space formulation

Example: simple exponential smoothing – ETS(ANN)

Simple exponential smoothing updates forecast based on latest realization of y_t:

- Forecast equation: $\hat{y}_{t+1|t} = \ell_t$
- Level equation: $\ell_t = \alpha \cdot y_t + (1 - \alpha) \cdot \ell_{t-1}$

If $y_t = \hat{y}_{t|t-1} + \epsilon_t$, can use *error correction* formulation:

- $y_t = \ell_{t-1} + \epsilon_t$
- $\ell_t = \ell_{t-1} + \alpha \cdot \epsilon_t$

Example: Holt's linear model – ETS(AAN)

ETS(AAN) adds slope to the model to better handle a trend:

- Forecast equation: $\hat{y}_{t+h|t} = \ell_t + h \cdot b_t$
- Level equation: $\ell_t = \alpha \cdot y_t + (1 - \alpha) \cdot (\ell_{t-1} + b_{t-1})$
- Trend equation: $b_t = \beta^* \cdot (\ell_t - \ell_{t-1}) + (1 - \beta^*) \cdot b_{t-1}$

Hyndman's taxonomy

Hyndman categorizes exponential smoothing models as ETS:

- E for type of error
- T for type of trend
- S for type of seasonality

Typical values are:

- A for additive
- M for multiplicative
- N for none
- A_d for additive damped
- M_d for multiplicative damped

Example:

ETS makes it easy to describe the type of model you want to use:

- ETS(AAN):
  - Has additive error and trend but no seasonality
  - Simple exponential smoothing
  - I.e., Holt's linear method, 'double exponential smoothing'
- ETS(AAA):
  - Holt-Winters' method
  - Adds seasonality

The ETS model

Python provides partial support for ETS:

- See Panda's `pandas.stats.moments.ewma`
- User unfriendly
- Best to use R's `ets` function in the `forecast` package

ETS vs. ARIMA

ARIMA features & benefits:

- Benchmark model for almost a century
- Much easier to estimate with modern computational resources
- Easy to diagnose models graphically
- Easy to fit using Box-Jenkins methodology

ETS features & benefits:

- Can handle non-linear and non-stationary processes
- Can be computed with limited computational resources
- Not always a subset of ARIMA
- Easier to explain to non-technical stakeholders

Summary

Summary

You should now be able to answer the following questions:

- What are the steps in the Box-Jenkins's approach?
- How much data do I need to forecast at horizon h?
- How should I evaluate a forecast?
- What are the benefits of ARIMA vs. ETS?