METIS

Time Series Analysis

INTRODUCTION TO DATA SCIENCE – FALL 2018

EXTRA SESSION 8

AGENDA

Extra Session 8

1. Time series data
2. ARIMA
3. ARIMA variants
4. Facebook prophet
5. Google bsts

Introduction to Data Science

- Learning the steps in the Data Science Process
- Learning multiple model methodologies

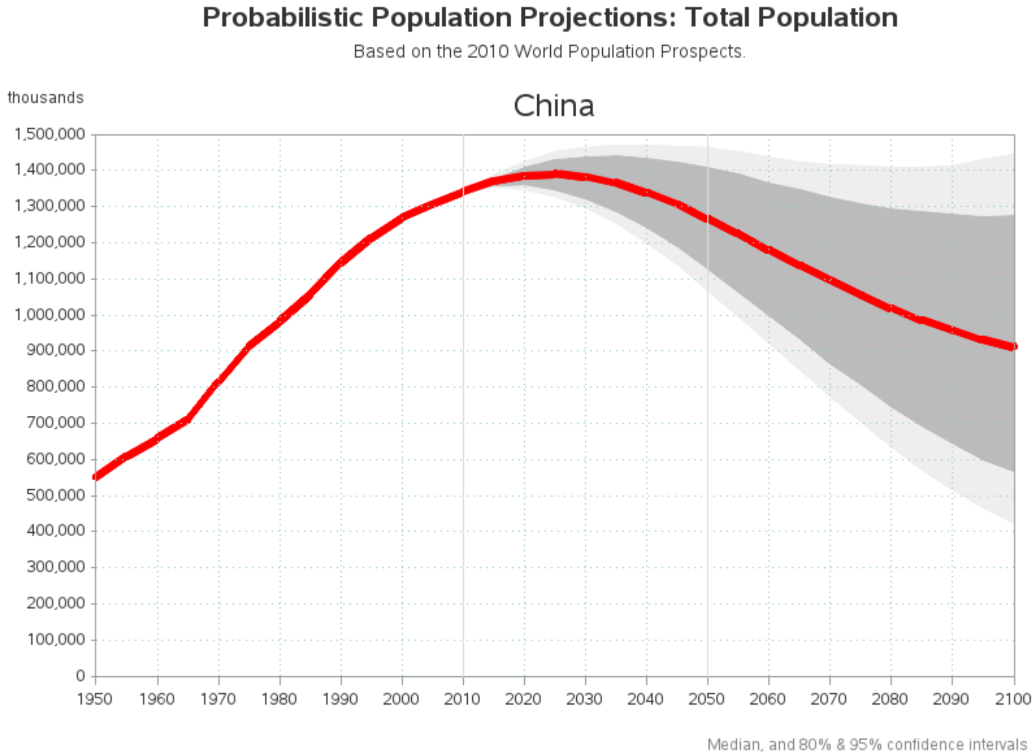
Data Evaluation	Cross Validation	Advanced Error Metrics	Visualization
Deep Learning	Neural Networks	Forecasting or NLP -RNN	Computer Vision -CNN
Machine Learning	Decision trees and Ensembles	Time Series	Clustering
Statistical Learning	Linear Regression	Logistic Regression	PCA–dimensionality reduction
Data Prep	Data Cleaning	Feature Engineering	Visualization
Foundation	Statistics	Python	Linear Algebra

Time series data

TIME DEPENDENT DATA

Forecasting

- Uses of forecasting:
  - Forecasting populations
  - Should we build another power plant in the next five years due to forecasts of energy demand?
  - Scheduling staff in a call center based on forecasts of call volume
  - Stocking an inventory based on forecasts of purchases
  - Forecasting stock prices?

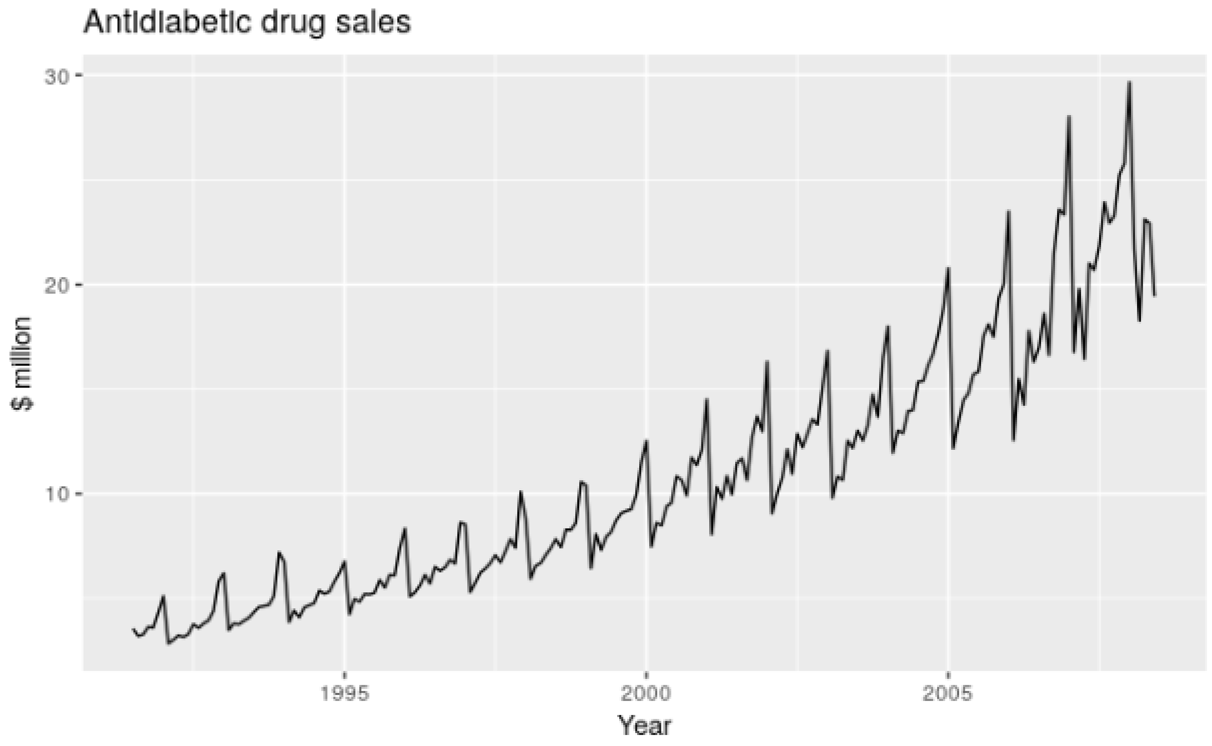


Basic steps in a forecasting task

- Problem definition
  - Who, what, how will be using the forcasts
- Gather information
- Exploratory analysis
- Choosing and fitting models
  - Depends on data, relationships, ways forecasts are to be used
- Using and evaluating a forecast model

Visualizing time series data

- Time series plot
  - X axis is typically time
- Time series data
  - Ordered by date



Year	Observation
2012	123
2013	39
2014	78
2015	52
2016	110

ARIMA

THE CLASSIC TIME SERIES MODEL

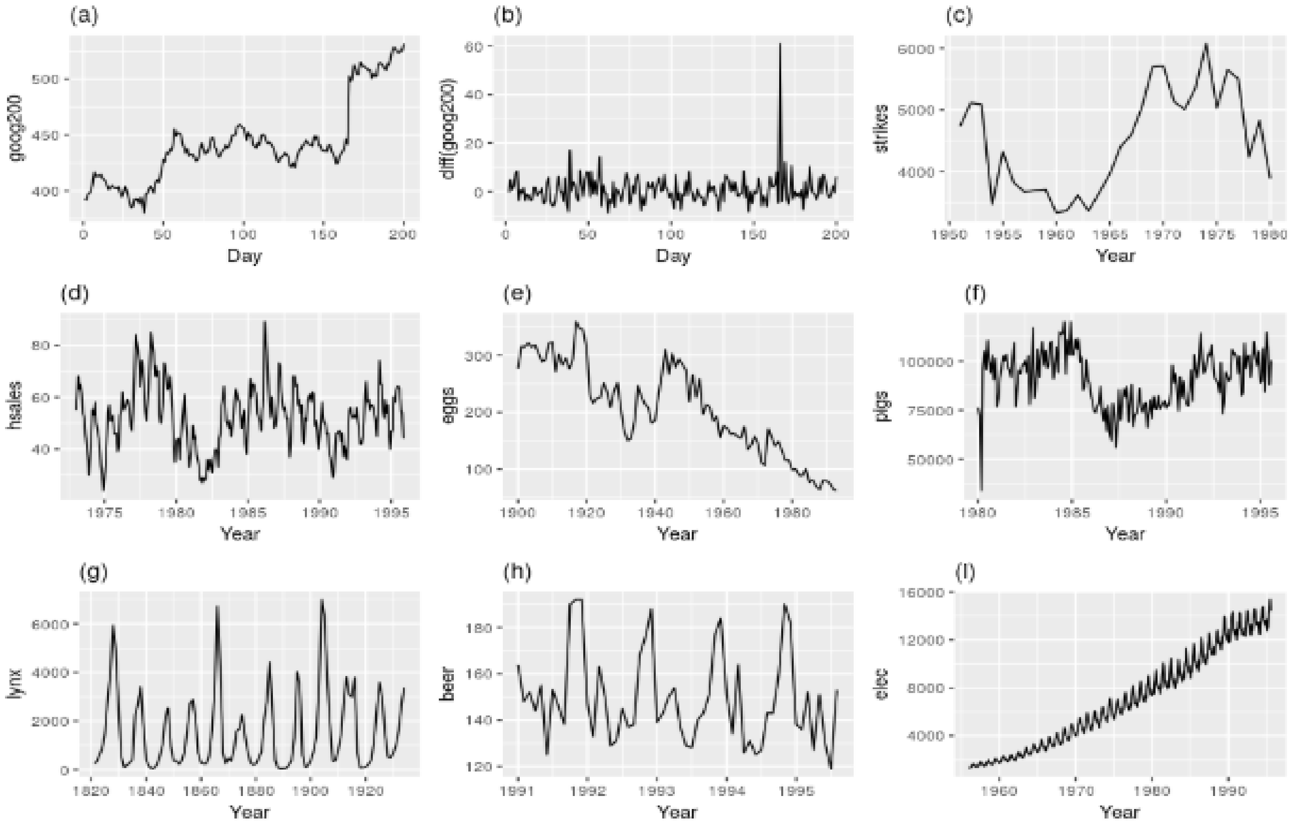
Stationarity and differencing

Stationary time series: properties do not depend on the time when the series is observed

- No seasonality
- No trend
- Constant variance

Which ones are stationary?

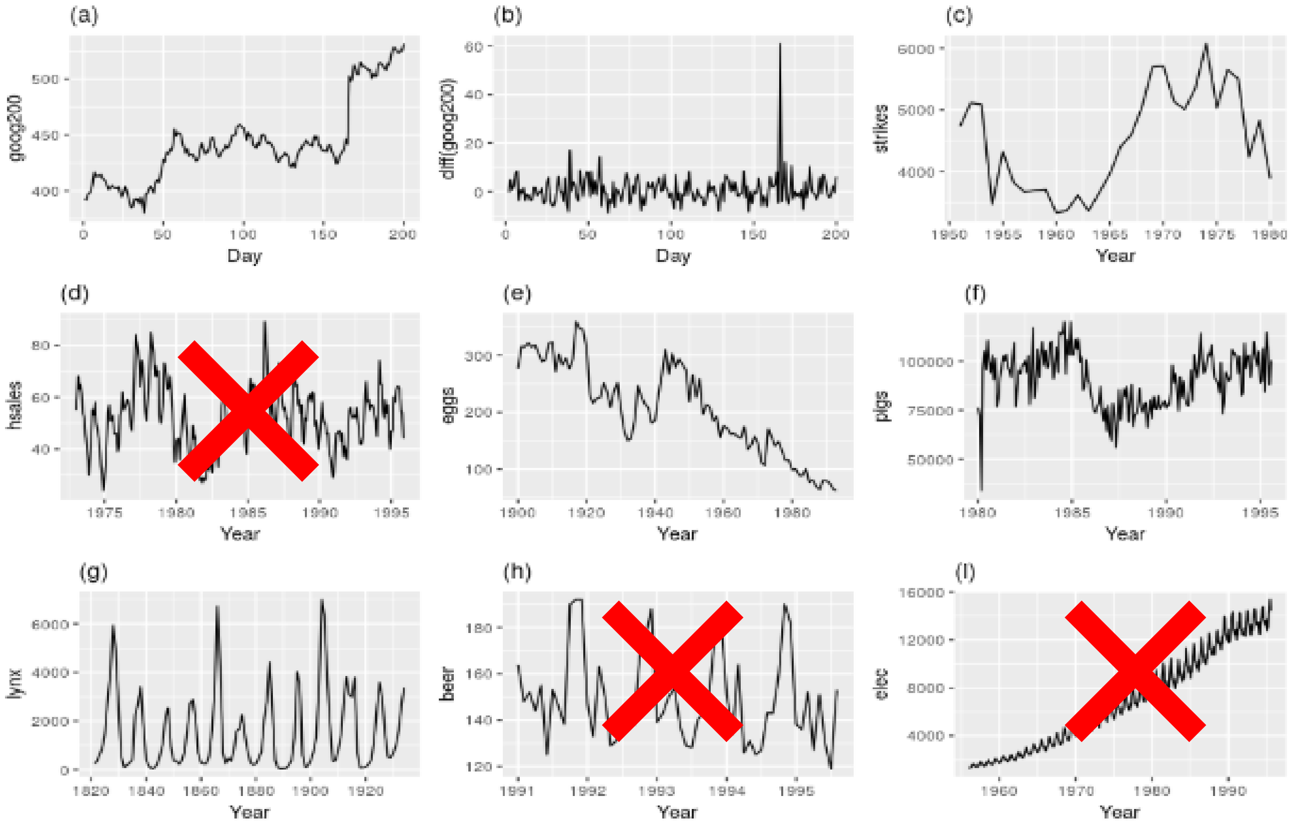
- Hint: only 2



No seasonality

Stationary time series: properties do not depend on the time when the series is observed

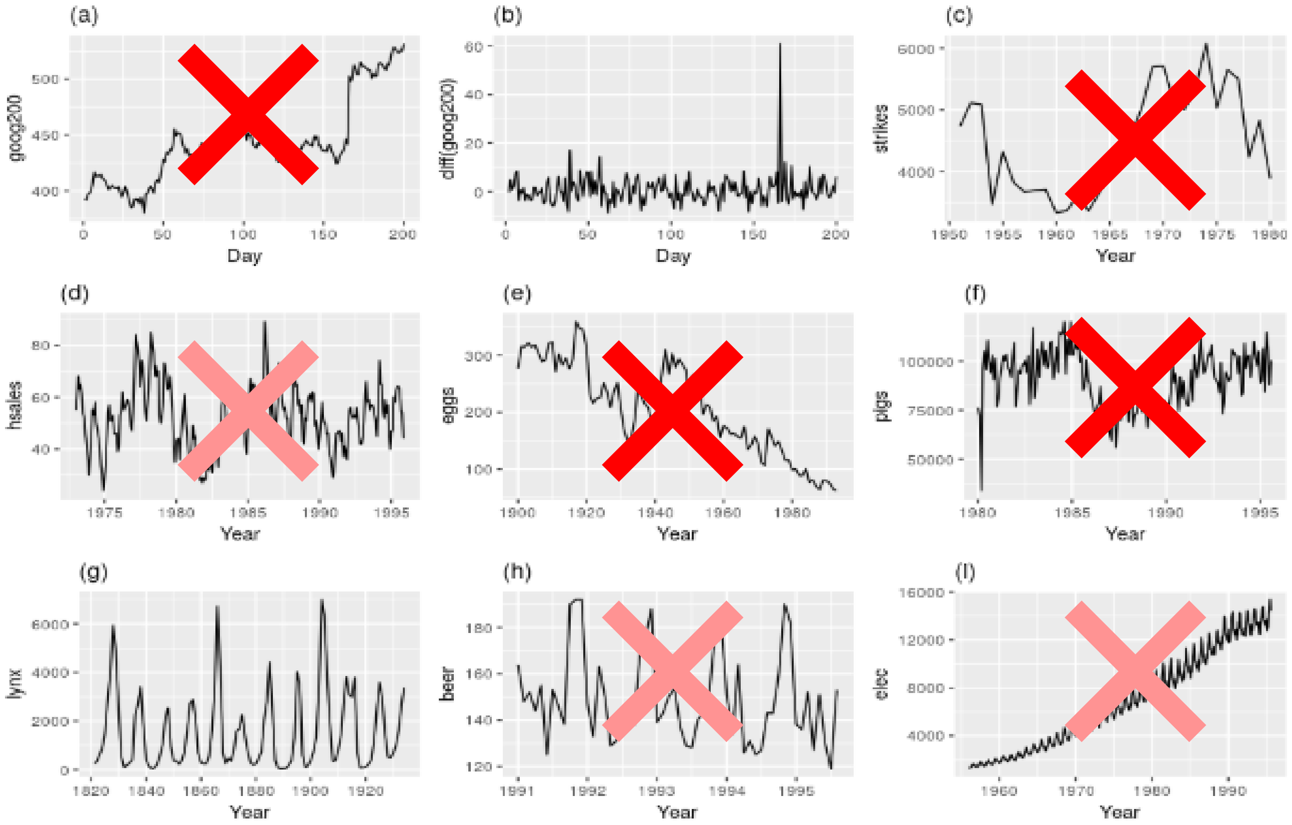
- No seasonality
- No trend or level changes
- Constant variance



No trends or changing levels

Stationary time series: properties do not depend on the time when the series is observed

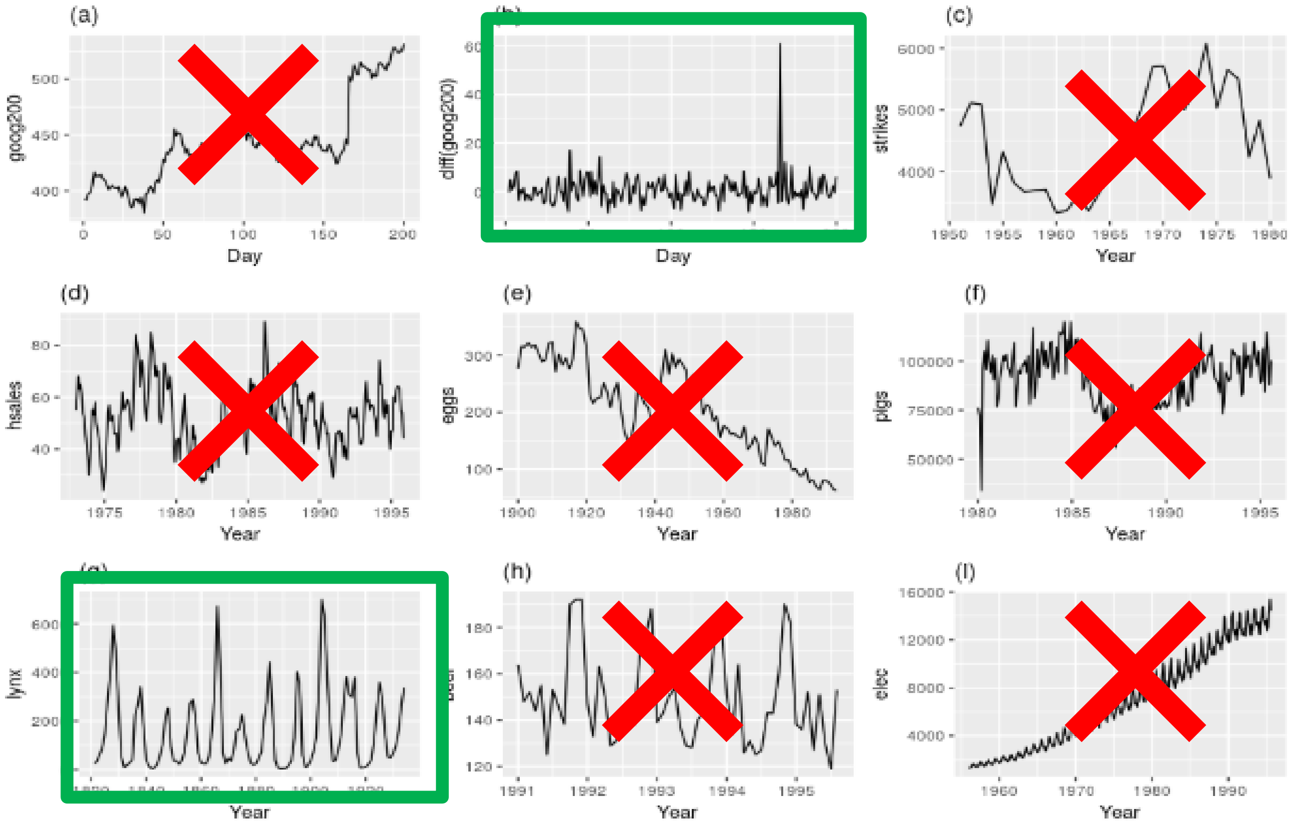
- No trend
- No seasonality
- Constant variance



Stationary series

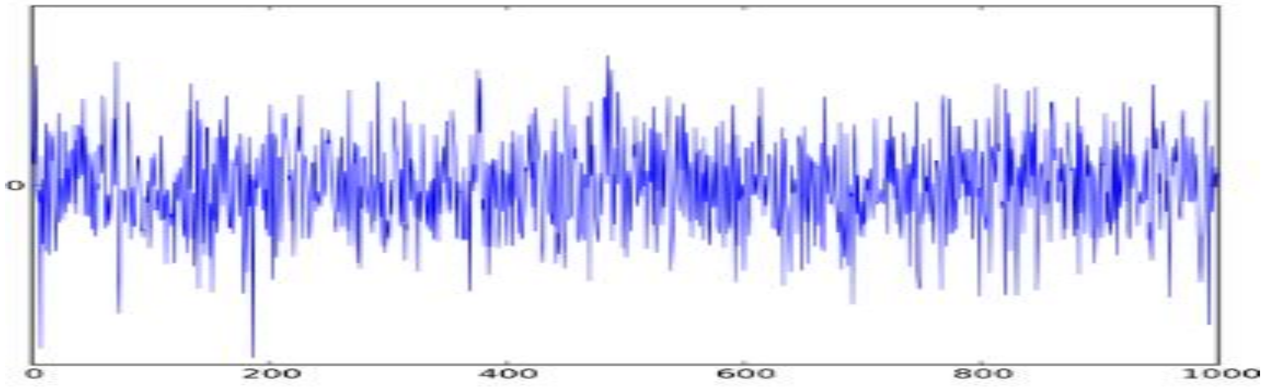
Stationary time series: properties do not depend on the time when the series is observed

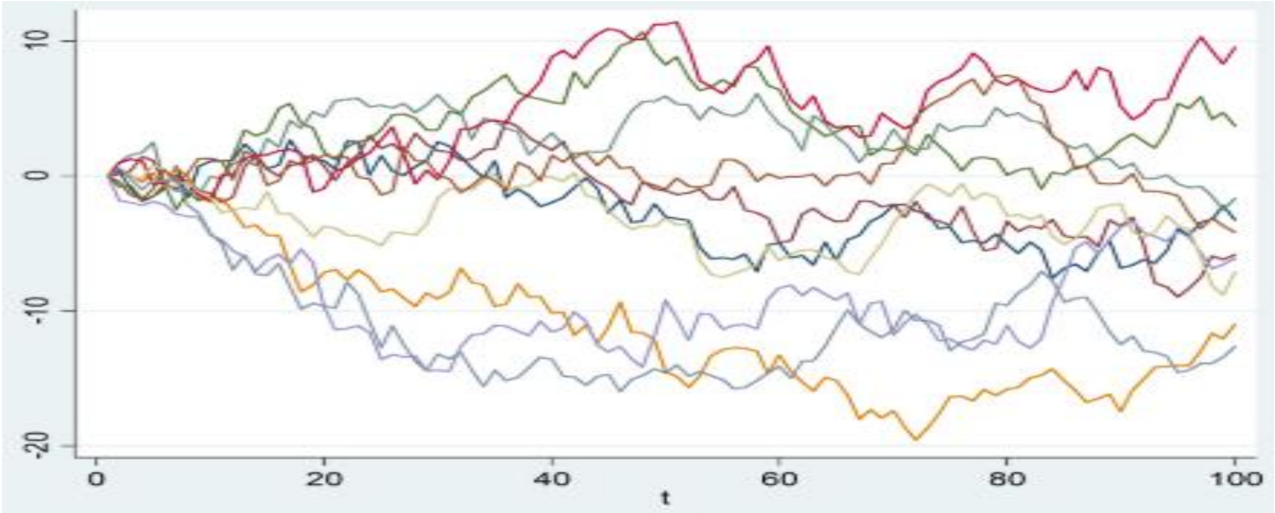
- No trend
- No seasonality
- Constant variance

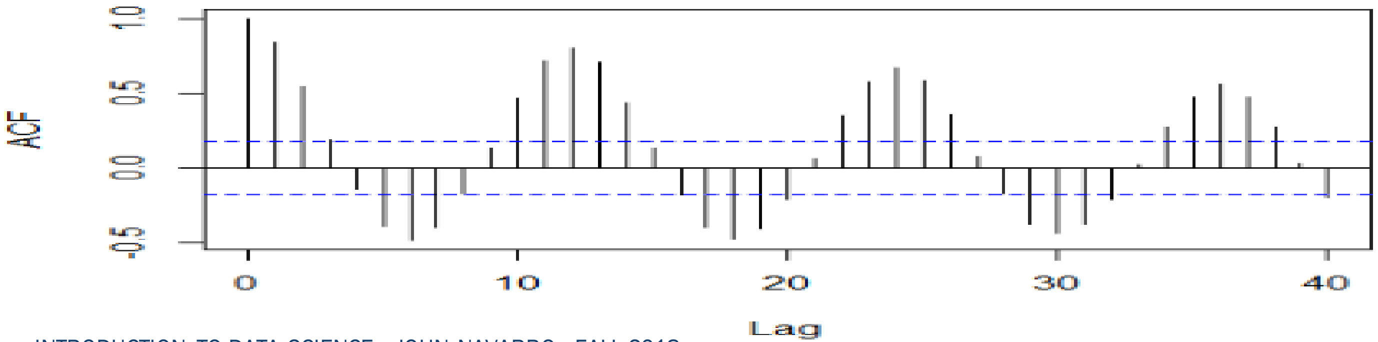


Terminology

- White noise
  - Variables are independent and identically distributed with a mean of 0
  - All variables have the same variance and zero correlation with other values in the series
- Random walk
  - Each step is random, but is dependent on the location of the previous step
- Autocorrelation
  - Plot of correlation between observations







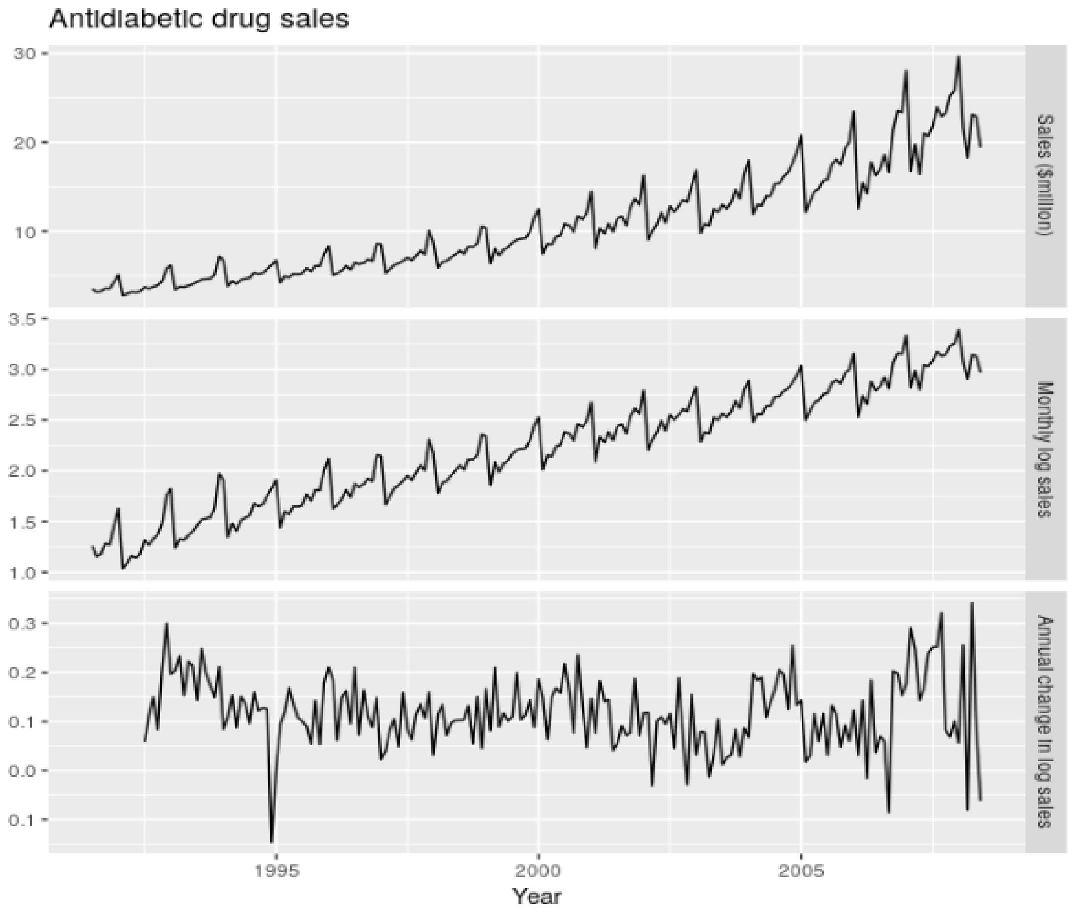
Stationarity and differencing

Log transformation helps stabilize the variance of a time series

Differencing is computing the differences between consecutive observations

Helps to stabilize the mean of the time series by removing changes in the level

Log difference is a common transformation



AR – autoregressive models

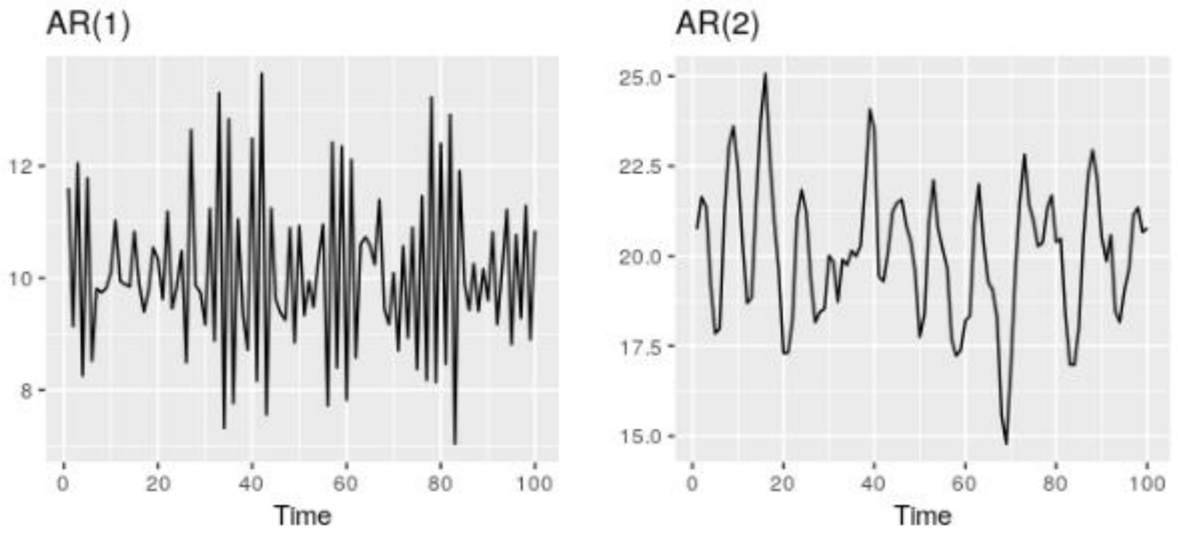
Forecast the variable of interest using a linear combination of **past values of the variable.**

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t,$$

Order of p

Et is white noise

AR(p) model: an autoregressive model of order p



MA – moving average models

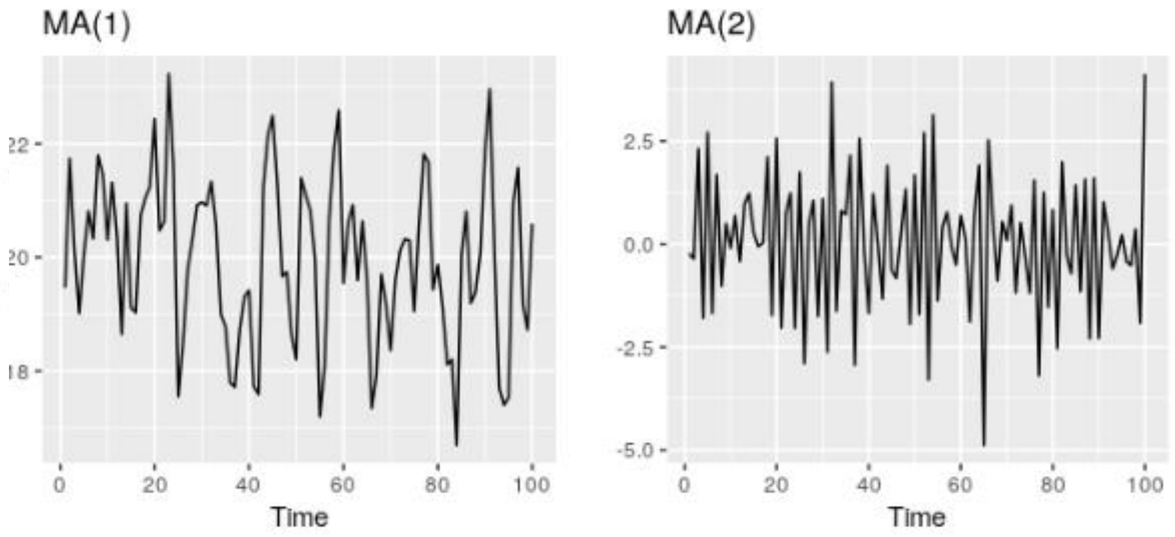
Forecast the variable of interest using the **past forecast errors in a regression like model**.

$$y_t = c + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \cdots + \theta_q\varepsilon_{t-q},$$

Order of q

Et is white noise

MA(q) model: a moving average model of order q



ARIMA – putting it together

Combine differencing with auto regression and a moving average model

ARIMA Auto Regressive Integrated Moving Average model

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t,$$

$p =$	order of the autoregressive part;
$d =$	degree of first differencing involved;
$q =$	order of the moving average part.

ARIMA(p,d,q) model

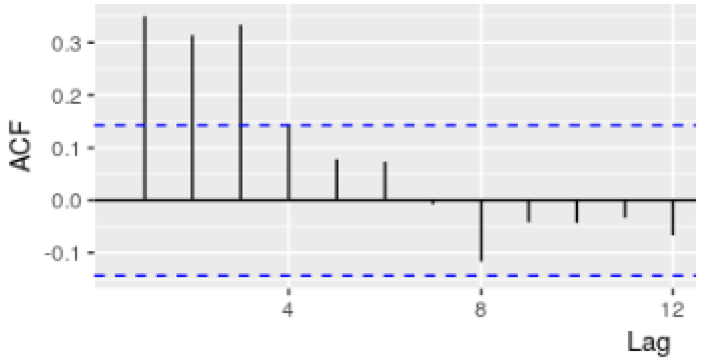
White noise	ARIMA(0,0,0)
Random walk	ARIMA(0,1,0) with no constant
Random walk with drift	ARIMA(0,1,0) with a constant
Autoregression	ARIMA(p,0,0)
Moving average	ARIMA(0,0,q)

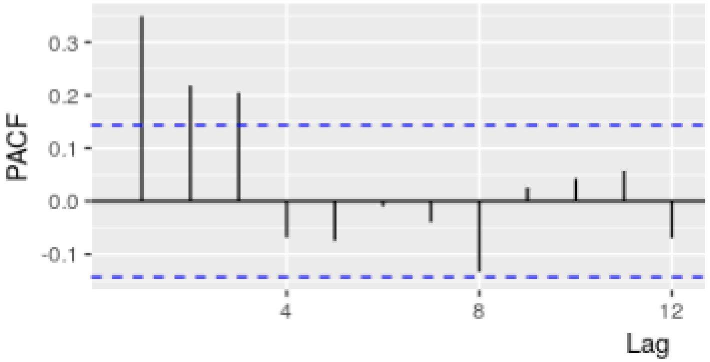
Special cases of ARIMA

ACF and PACF plots

ACF plot shows the autocorrelations between any y_t and y_{t-k}

PACF measures the relationship of any y_t and y_{t-k} after removing the effects of lags 1,2,3,4..k-1



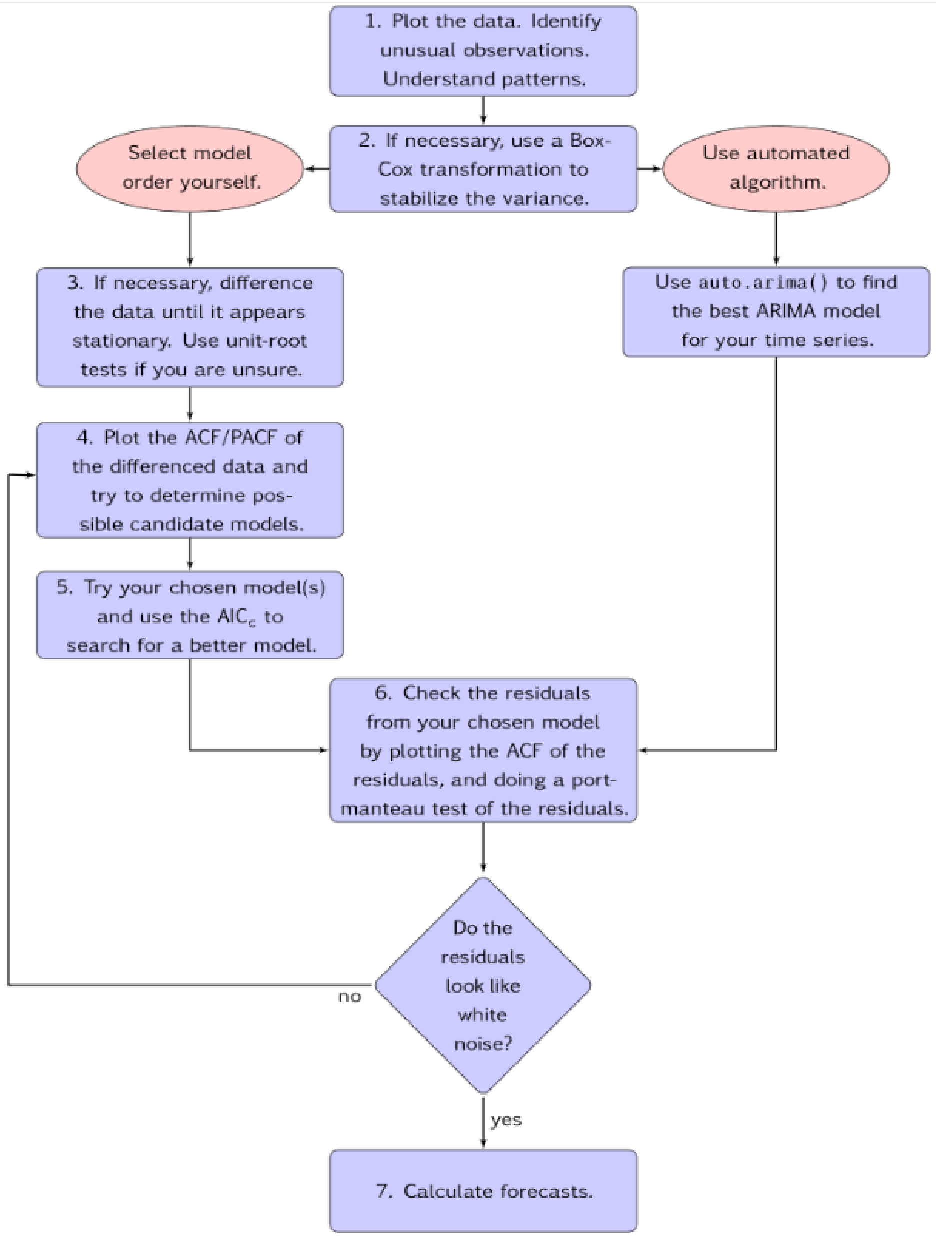


To determine p in an ARIMA(p,d,0) model: want to see that ACF is exponentially decaying or sinusoidal and look for a lag in PDF at p

To determine q in an ARIMA(0,d,q) model: want to see that PACF is exponentially decaying or sinusoidal and look for a lag in ACF at q

Modeling procedure

1. Plot the data and identify any unusual observations.
2. If necessary, transform the data (using a Box-Cox transformation) to stabilize the variance.
3. If the data are non-stationary, take first differences of the data until the data are stationary.
4. Examine the ACF/PACF: Is an ARIMA(p,d,0p,d,0) or ARIMA(0,d,q0,d,q) model appropriate?
5. Try your chosen model(s), and use the AICc to search for a better model.
6. Check the residuals from your chosen model by plotting the ACF of the residuals, and doing a portmanteau test of the residuals. If they do not look like white noise, try a modified model.
7. Once the residuals look like white noise, calculate forecasts



ARIMA variants

DIFFERENT APPLICATIONS FOR ARIMA MODELS

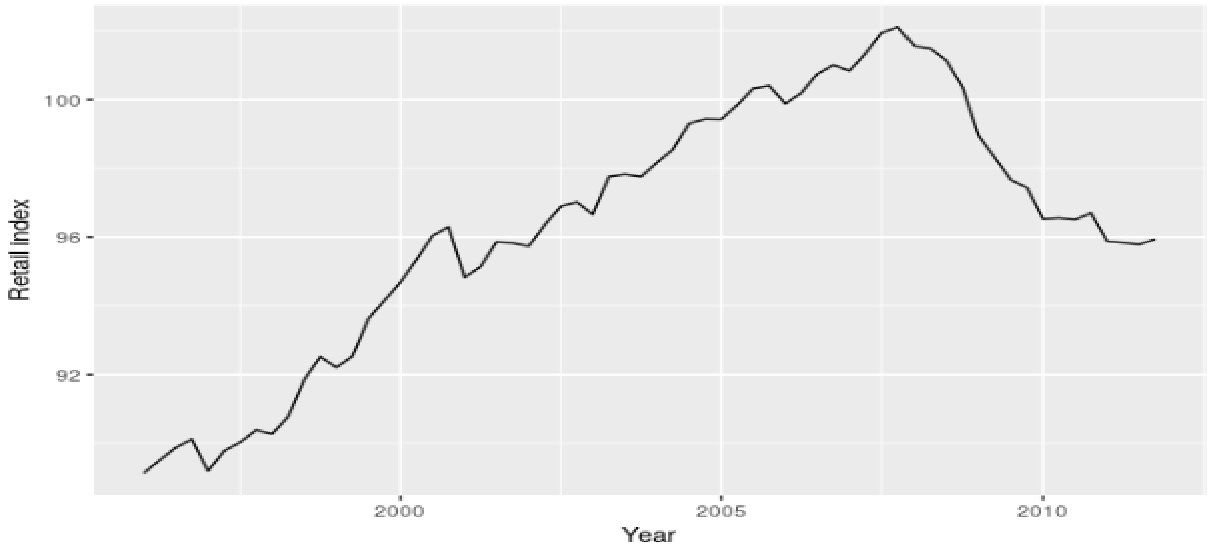
Seasonal ARIMA models

Contains additional seasonal terms in the ARIMA models we have seen so far

Where m = number of observations per year

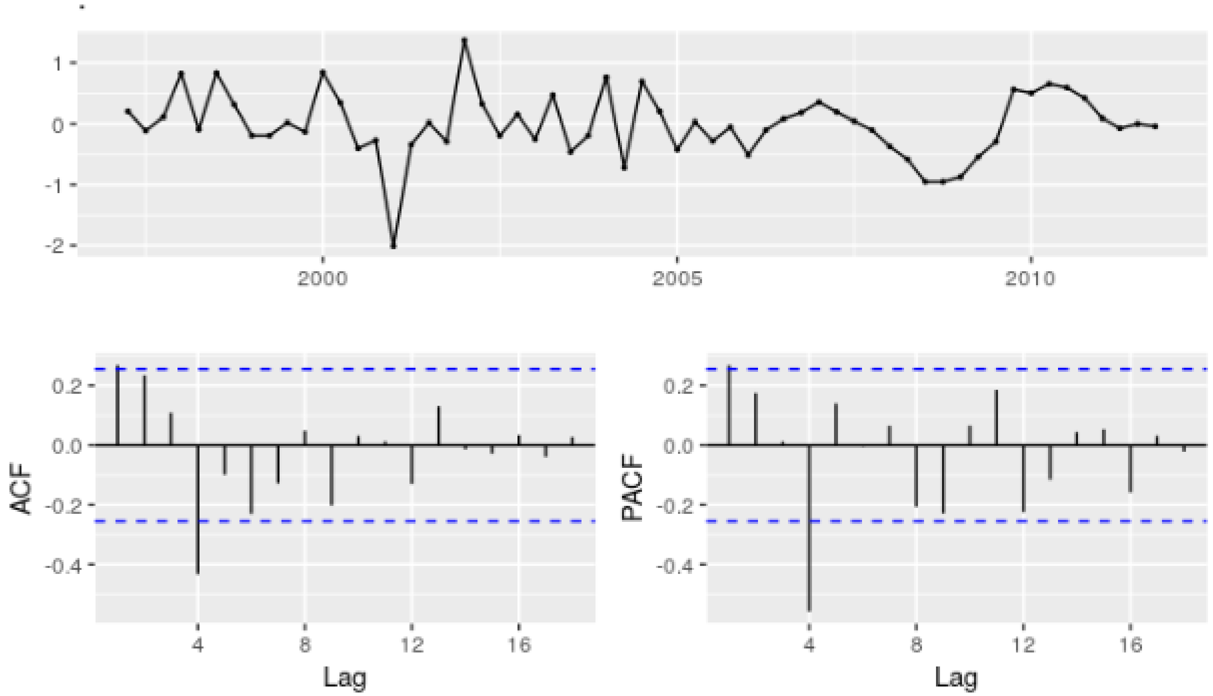
Uppercase notation for the seasonal parts and lowercase for the non seasonal parts.

$$\text{ARIMA} \quad \underbrace{(p,d,q)}_{\substack{\uparrow \\ \text{Non-seasonal part} \\ \text{of the model}}} \quad \underbrace{(P,D,Q)_m}_{\substack{\uparrow \\ \text{Seasonal part of} \\ \text{of the model}}}$$



Seasonal ARIMA – ACF/PACF

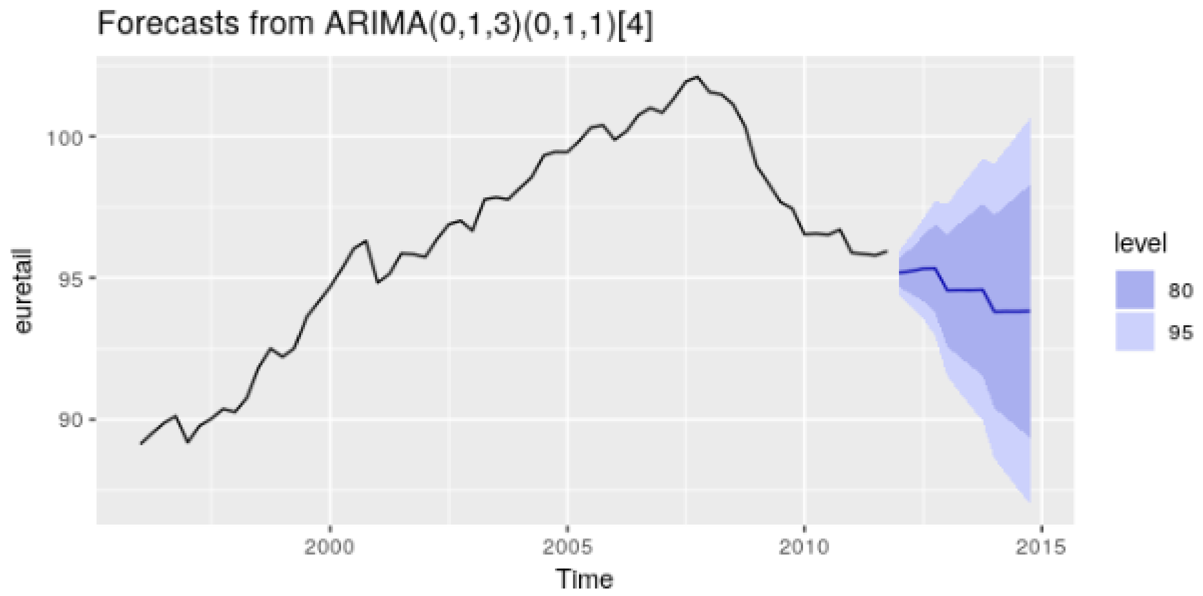
- The seasonal part of the model will be seen in the seasonal lags of the ACF or PACF plots
  - For example a spike at lag 12 in the ACF but no other significant spikes
  - Exponential decay in the seasonal lags of the PACF at 12, 24, 26 etc
- Modeling procedure is the same as non-seasonal data except that we need to select seasonal AR and MA terms as well as non seasonal components



Data is twice differenced. Seasonally (lag 4) and first order (lag 1)
ACF: Spike at lag 1 is NS MA(1), spike at lag 4 is Sea MA(4)
Suggested model to begin with ARIMA(0,1,1)(0,1,1)4

Seasonal ARIMA – fitting models

- We fit different models and compare using error metrics like AICc
- Predictions are made by the model, then seasonal components added back in to final predictions



VAR – Vector autoregressions

Consider a data set where variables all influence each other.

Typical example is economic model
- Personal consumption
- Personal Income

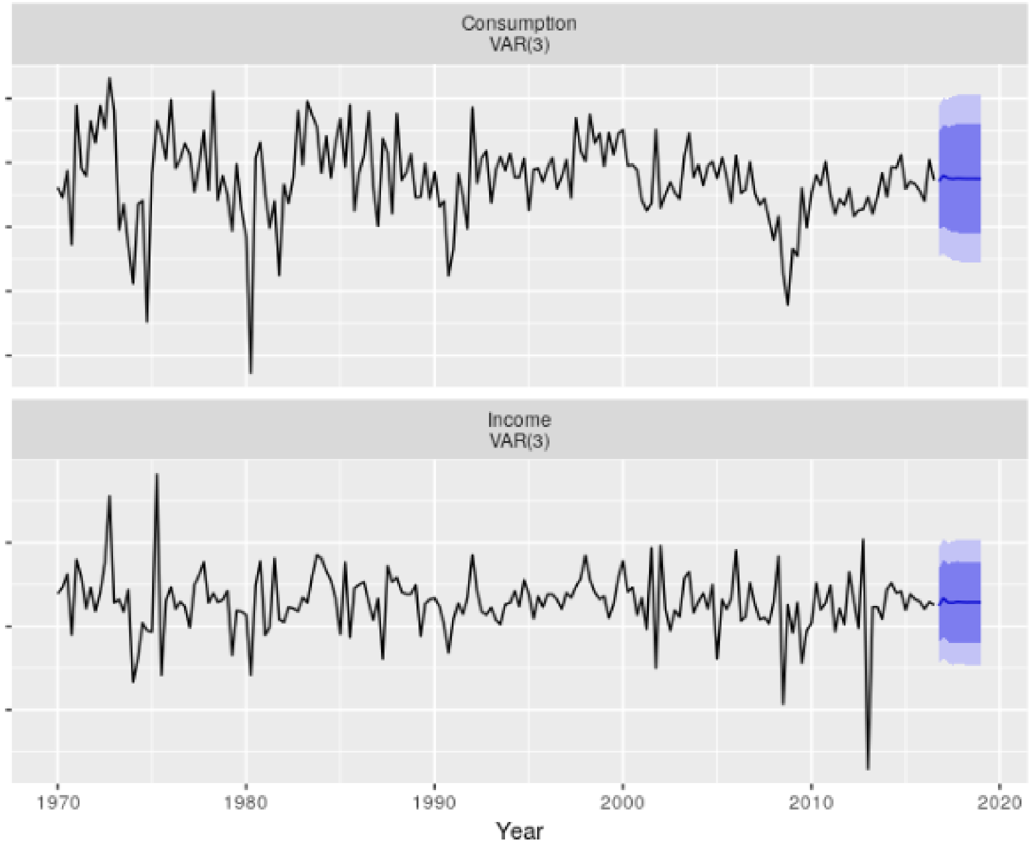


Figure 11.10: Forecasts for US consumption and income generated from a VAR(3).

Facebook prophet

FORECASTING AT SCALE

What is prophet?

- Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly and daily seasonality, plus holiday effects
- Prophet is open source software released by Facebook’s Core Data Science team.
- Prophet is robust to outliers, missing data and dramatic changes in your time series
- Use human interpretable parameters to improve your forecast by adding domain knowledge
- Available in R and Python

PROPHET

https://facebook.github.io/prophet/

Using prophet in python

- Create an instance of the Prophet class then call fit and predict methods
- Input is a data frame with two columns: **ds** (datestamp: YYYY-MM-DD) and **y** (numeric)

	DS	Y
0	2007-12-10	9.590761
1	2007-12-11	8.519590
2	2007-12-12	8.183677
3	2007-12-13	8.072467
4	2007-12-14	7.893572

```
# Python
m = Prophet()
m.fit(df)
```

```
# Python
future = m.make_future_dataframe(periods=365)
future.tail()
```

```
# Python
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

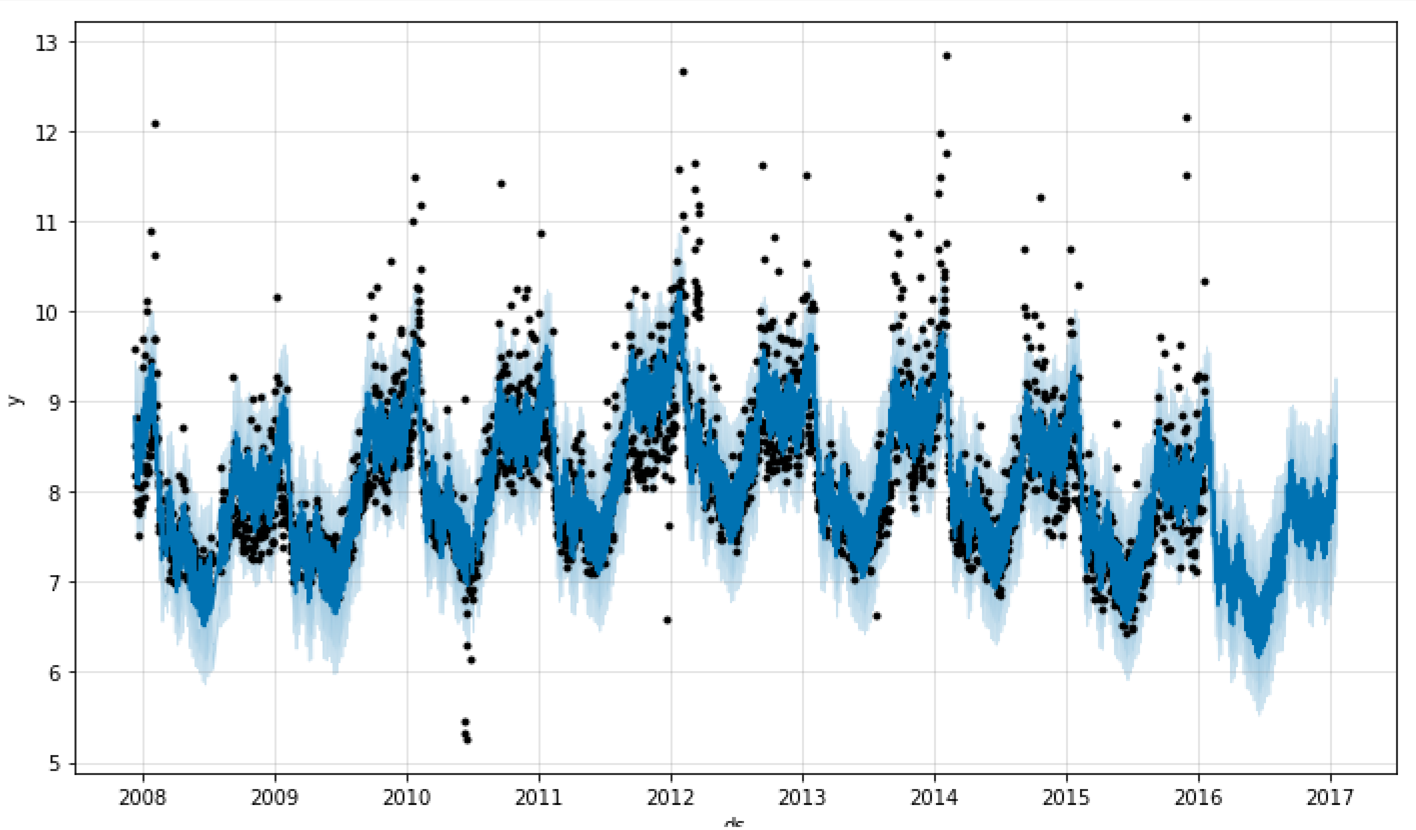
	DS	YHAT	YHAT_LOWER	YHAT_UPPER
3265	2017-01-15	8.199274	7.489884	8.969065
3266	2017-01-16	8.524244	7.790682	9.266504
3267	2017-01-17	8.311615	7.553025	9.049803
3268	2017-01-18	8.144232	7.428174	8.864747
3269	2017-01-19	8.156091	7.395160	8.883232

https://facebook.github.io/prophet/

Plotting forecasts

- Call the prophet.plot method and pass in the forecast data frame

```
# Python
fig1 = m.plot(forecast)
```

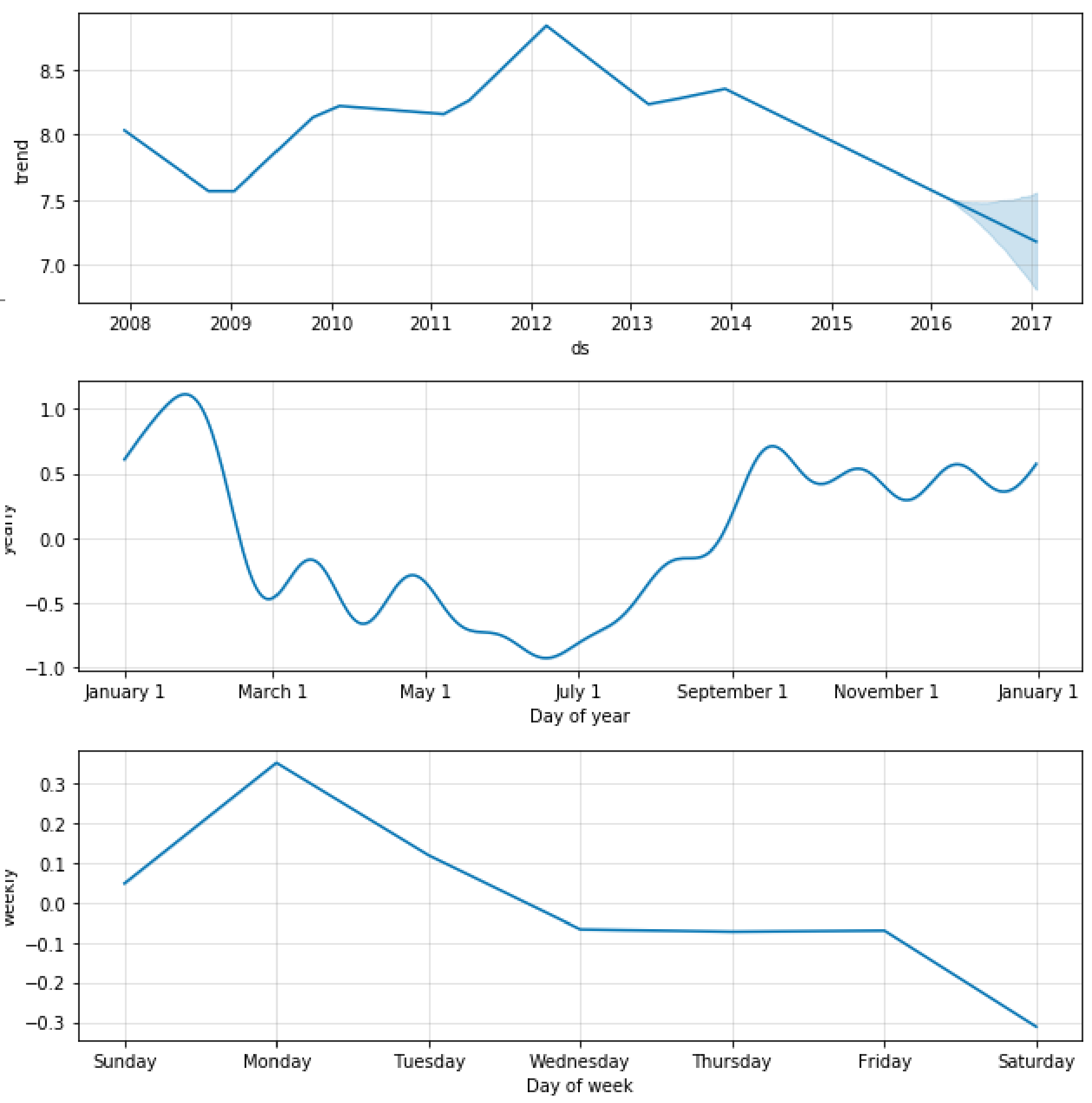


https://facebook.github.io/prophet/

Forecast components

- Call the prophet.plot_components method to see the trend yearly and weekly seasonalities

```
# Python
fig2 = m.plot_components(forecast)
```



Appendix

Google bsts

STRUCTURAL TIME SERIES MODELING

Acknowledgments

Sources for this lecture include but not limited to:

Hyndman, Forecasting principles and practice

https://facebook.github.io/prophet/

http://www.unofficialgoogledatascience.com/2017/07/fitting-bayesian-structural-time-series.html