Time Series Data Analysis and Forecasting

MSDS

Module 4

Topic Covered

- ➢Exponential Smoothing
- ➢Exponential smoothing methods
- ➢Autoregressive (AR) models
- ➢Moving average (MA) models
- ➢Integrated (I) models
- ➢Introduction to ARIMA models
- ➢Use cases

Exponential Smoothing

- Exponential smoothing was proposed in late 1950s (Brown, 1959; Holt, 1957; Winters, 1960)
- Forecasts produced using exponential smoothing methods are **weighted averages of past observations**, with weights decaying exponentially as observations get older.
- the more recent the observation, higher the associated weight.
- It generates reliable forecasts quickly and for a wide range of time series
- Provides major importance to applications in industry.
- selection of method is based on recognising key components of time series (**trend and seasonal)**
- Also, the way in which these enter smoothing method (e.g., in **an additive, damped or multiplicative manner**).

Simple Exponential Smoothing

- This method is suitable for forecasting data with **no clear trend or seasonal pattern**
- As the data in Figure**, do not display any clear trending** behavior or any seasonality.
- It shows is a rise in the last few years, which might suggest a trend.
- Using **naïve method**, all forecasts for future are **equal to last observed value** of series
- Or average method is used to forecast values

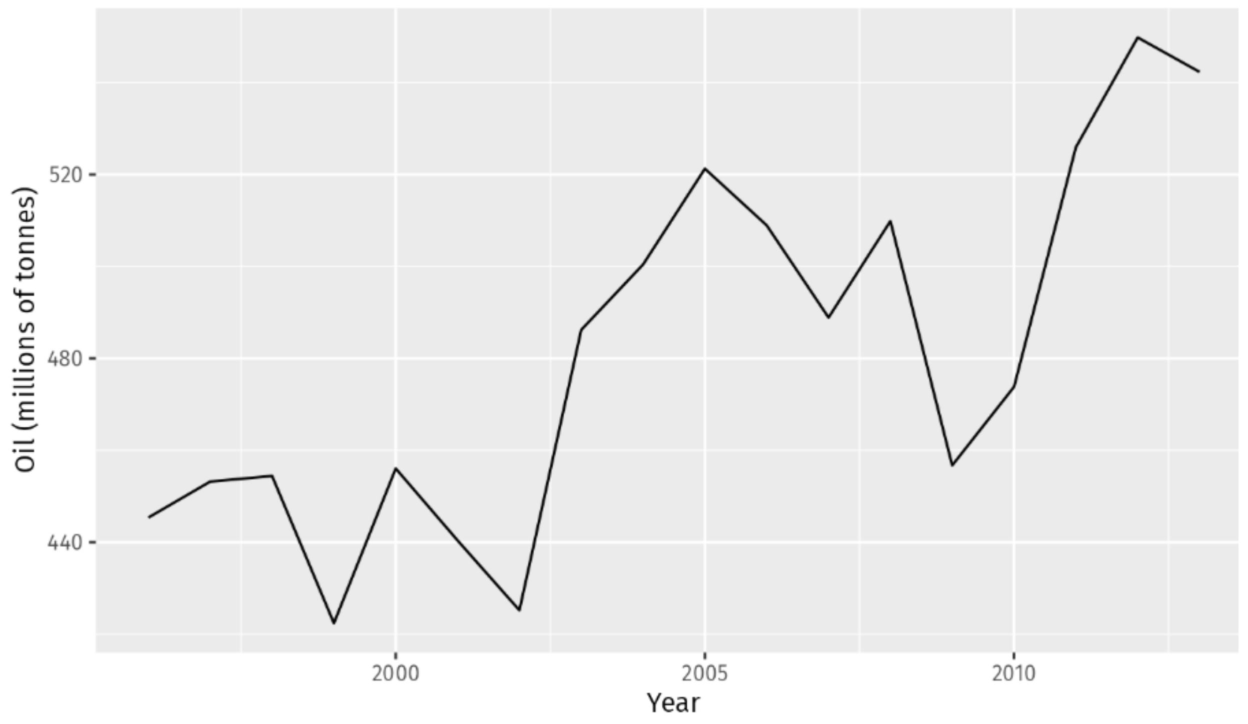


Figure 7.1: Oil production in Saudi Arabia from 1996 to 2013.

$$\hat{y}_{T+h|T} = y_T$$

$$\text{for } h = 1, 2, \ldots.$$

$$\hat{y}_{T+h|T} = \frac{1}{T}\sum_{t=1}^{T} y_t,$$

Simple Exponential Smoothing

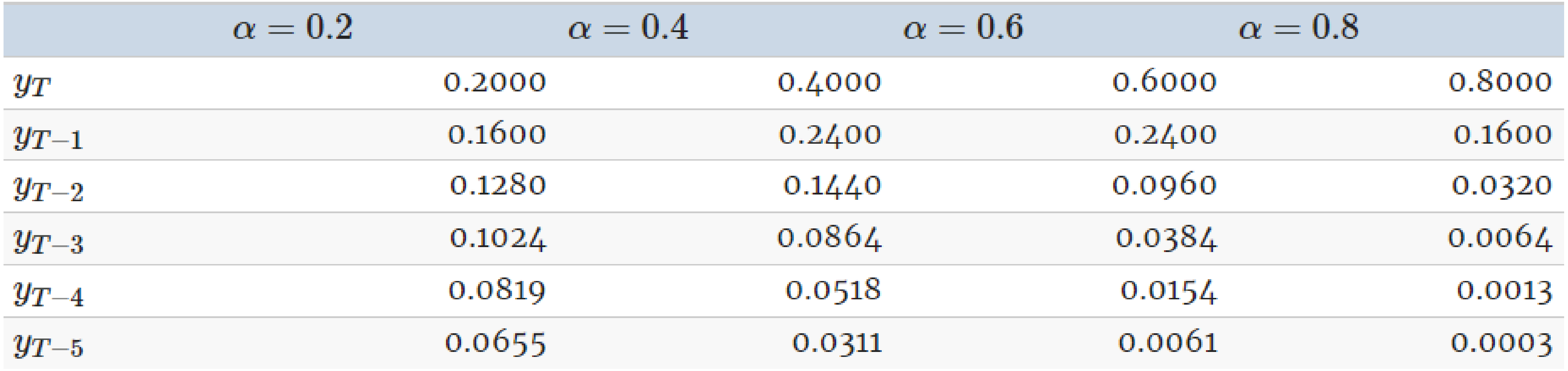
- As both naïve method or average method does not associate any weight to past values Weighted average method is used to make forecast better and more accurate.
- attach **larger weights to more recent observations** than to observations from the distant past
- In exponential method, Forecasts are calculated using **weighted averages**, where the weights decrease exponentially as observations come from further in the past — the smallest weights are associated with the oldest observations.

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha) y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots,$$

where $0 \le \alpha \le 1$ is the smoothing parameter. The one-step-ahead forecast for time $T+1$ is a weighted average of all of the observations in the series $y_1, \dots, y_T$. The rate at which the weights decrease is controlled by the parameter α.

Simple Exponential Smoothing

- table shows the weights attached to observations for four different values of α when forecasting using simple exponential smoothing.
- Note that sum of the weights even for a small value of α will be approximately one for any reasonable sample size.
- For any **α between 0 and 1**, weights attached to observations decrease exponentially as we go back in time, hence name **“exponential smoothing”**



	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$
y_T	0.2000	0.4000	0.6000	0.8000
y_{T-1}	0.1600	0.2400	0.2400	0.1600
y_{T-2}	0.1280	0.1440	0.0960	0.0320
y_{T-3}	0.1024	0.0864	0.0384	0.0064
y_{T-4}	0.0819	0.0518	0.0154	0.0013
y_{T-5}	0.0655	0.0311	0.0061	0.0003

Two Forms of Simple Exponential Smoothing

- Weighted average form
- Component form

Weighted average form for Simple Exponential Smoothing

The forecast at time $T+1$ is equal to a weighted average between the most recent observatio y_T and the previous forecast $\hat{y}_{T|T-1}$:

$$\hat{y}_{T+1|T} = \alpha y_T + (1-\alpha)\hat{y}_{T|T-1},$$

where $0 \leq \alpha \leq 1$ is the smoothing parameter. Similarly, we can write the fitted values as

$$\hat{y}_{t+1|t} = \alpha y_t + (1-\alpha)\hat{y}_{t|t-1},$$

for $t = 1, \dots, T$. (Recall that fitted values are simply one-step forecasts of the training data.)

$$\begin{aligned}
\hat{y}_{2|1} &= \alpha y_1 + (1-\alpha)\ell_0 \\
\hat{y}_{3|2} &= \alpha y_2 + (1-\alpha)\hat{y}_{2|1} \\
\hat{y}_{4|3} &= \alpha y_3 + (1-\alpha)\hat{y}_{3|2} \\
&\vdots \\
\hat{y}_{T|T-1} &= \alpha y_{T-1} + (1-\alpha)\hat{y}_{T-1|T-2} \\
\hat{y}_{T+1|T} &= \alpha y_T + (1-\alpha)\hat{y}_{T|T-1}.
\end{aligned}$$

$$\begin{aligned}
\hat{y}_{3|2} &= \alpha y_2 + (1-\alpha)\left[\alpha y_1 + (1-\alpha)\ell_0\right] \\
&= \alpha y_2 + \alpha(1-\alpha)y_1 + (1-\alpha)^2\ell_0 \\
\hat{y}_{4|3} &= \alpha y_3 + (1-\alpha)[\alpha y_2 + \alpha(1-\alpha)y_1 + (1-\alpha)^2\ell_0] \\
&= \alpha y_3 + \alpha(1-\alpha)y_2 + \alpha(1-\alpha)^2 y_1 + (1-\alpha)^3\ell_0 \\
&\vdots \\
\hat{y}_{T+1|T} &= \sum_{j=0}^{T-1}\alpha(1-\alpha)^j y_{T-j} + (1-\alpha)^T\ell_0.
\end{aligned}$$

Component form for Simple Exponential Smoothing

- This representations of exponential smoothing methods comprise a forecast equation and a smoothing equation for each of the components included in the method.
- The form for simple exponential smoothing is given by:

$$\begin{aligned} \text{Forecast equation} \qquad & \hat{y}_{t+h|t} = \ell_t \\ \text{Smoothing equation} \qquad & \ell_t = \alpha y_t + (1-\alpha)\ell_{t-1}, \end{aligned}$$

where ℓ_t is the level (or the smoothed value) of the series at time t. Setting $h = 1$ gives the fitted values, while setting $t = T$ gives the true forecasts beyond the training data.

Example: Oil Production

Simple exponential smoothing is applied to forecast oil production in Saudi Arabia.

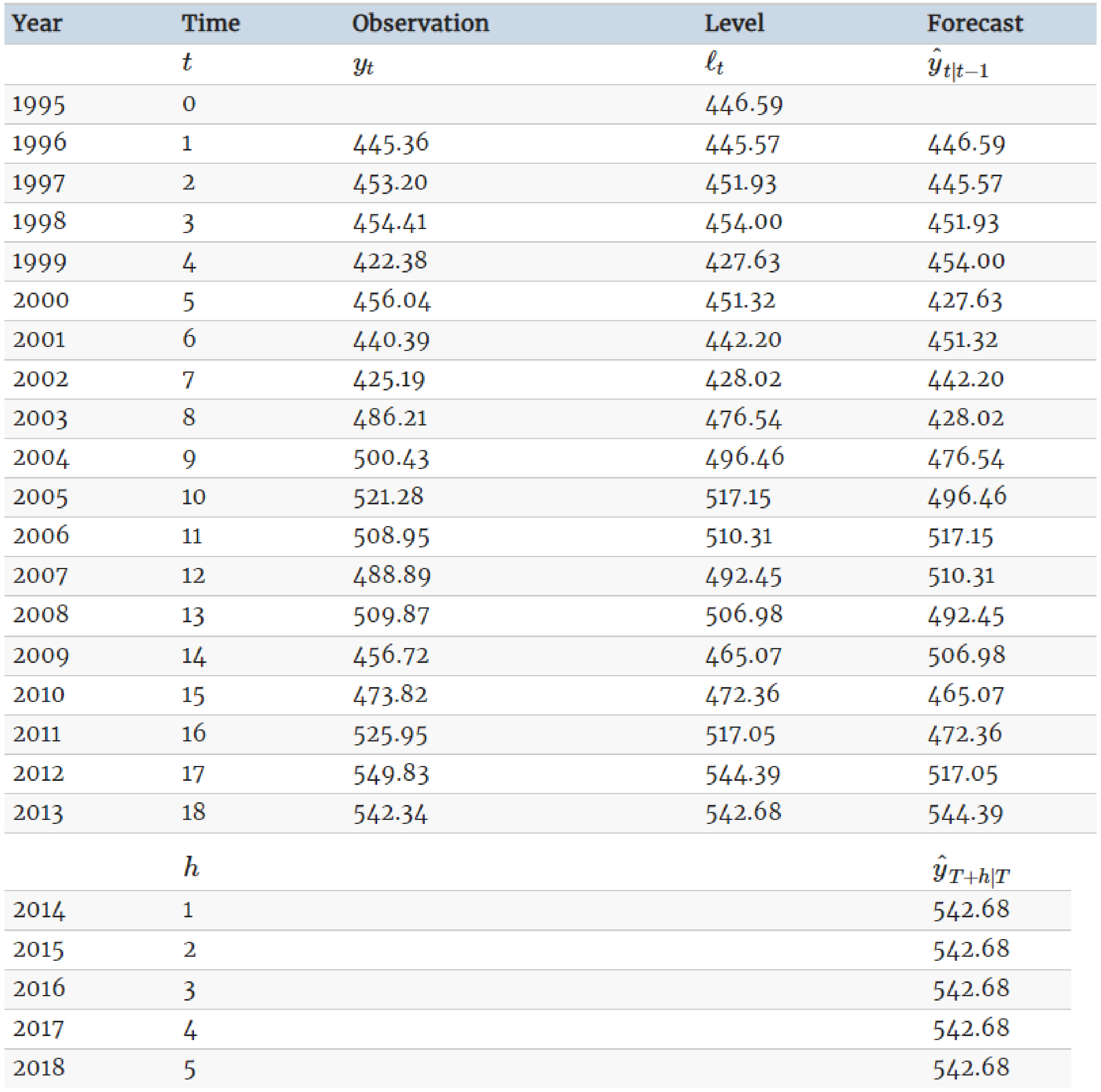
```
oildata <- window(oil, start=1996)
# Estimate parameters
fc <- ses(oildata, h=5)
# Accuracy of one-step-ahead training errors
round(accuracy(fc),2)
#>               ME  RMSE   MAE MPE MAPE MASE  ACF1
#> Training set 6.4 28.12 22.26 1.1 4.61 0.93 -0.03
```

This gives parameter estimates $\hat{\alpha} = 0.83$ and $\hat{\ell}_0 = 446.6$, obtained by minimising SSE over periods $t = 1, 2, \ldots, 18$, subject to the restriction that $0 \leq \alpha \leq 1$.

In Table 7.1 we demonstrate the calculation using these parameters. The second last column shows the estimated level for times $t = 0$ to $t = 18$; the last few rows of the last column show the forecasts for $h = 1, 2, 3, 4, 5$.

Example: Oil Production

- **second last column** shows the estimated level for times t=0 to t=18
- the **last few rows** of the last column show the forecasts for h=1,2,3,4,5.
- Forecasting the **total oil production** in millions of tonnes for Saudi Arabia



Year	Time	Observation	Level	Forecast
	t	y_t	ℓ_t	$\hat{y}_{t\|t-1}$
1995	0		446.59	
1996	1	445.36	445.57	446.59
1997	2	453.20	451.93	445.57
1998	3	454.41	454.00	451.93
1999	4	422.38	427.63	454.00
2000	5	456.04	451.32	427.63
2001	6	440.39	442.20	451.32
2002	7	425.19	428.02	442.20
2003	8	486.21	476.54	428.02
2004	9	500.43	496.46	476.54
2005	10	521.28	517.15	496.46
2006	11	508.95	510.31	517.15
2007	12	488.89	492.45	510.31
2008	13	509.87	506.98	492.45
2009	14	456.72	465.07	506.98
2010	15	473.82	472.36	465.07
2011	16	525.95	517.05	472.36
2012	17	549.83	544.39	517.05
2013	18	542.34	542.68	544.39
	h			$\hat{y}_{T+h\|T}$
2014	1			542.68
2015	2			542.68
2016	3			542.68
2017	4			542.68
2018	5			542.68

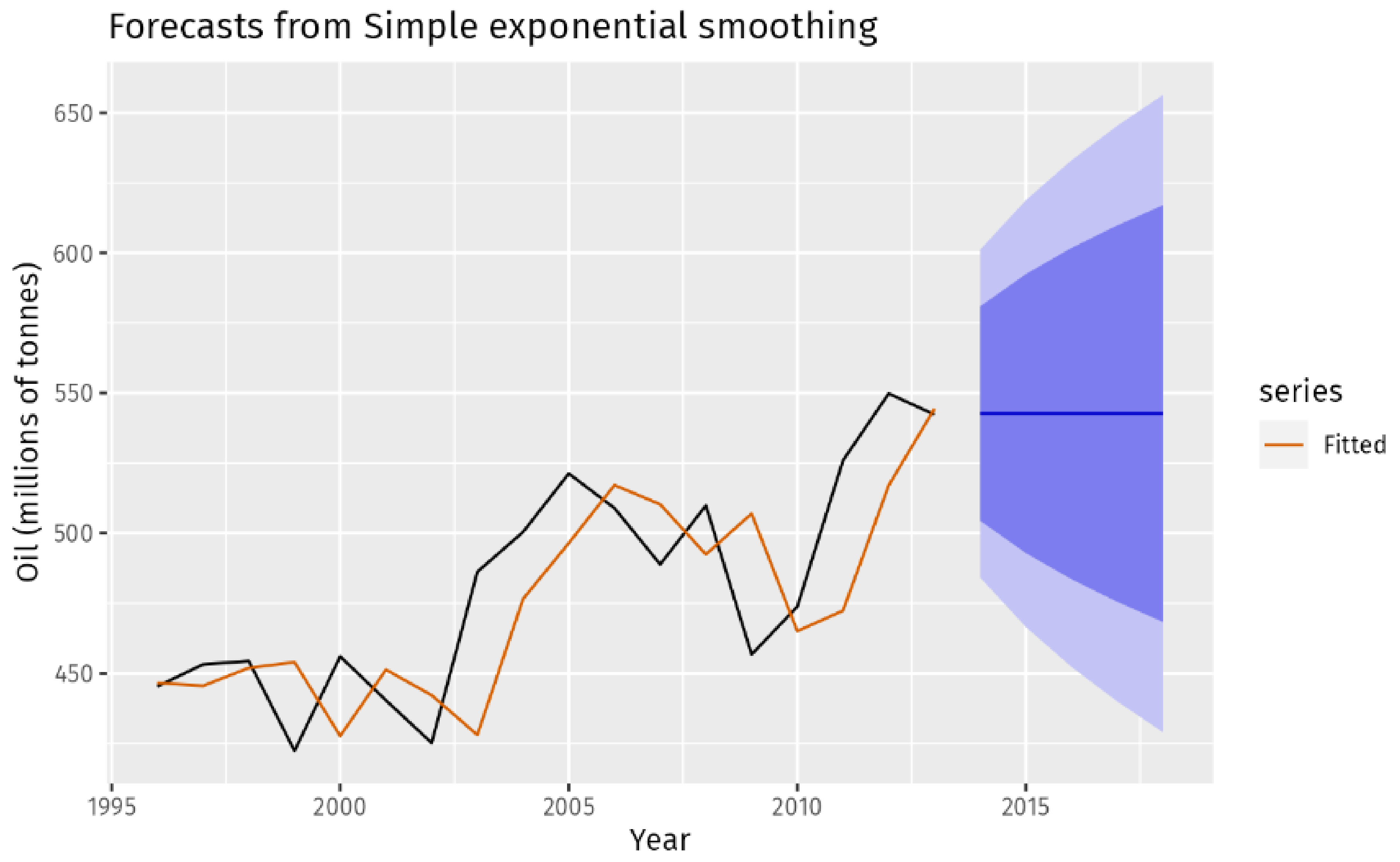


Figure 7.2: Simple exponential smoothing applied to oil production in Saudi Arabia (1996–2013).

Holt’s linear trend method

- Holt (1957) extended simple exponential smoothing to allow forecasting of data **with a trend**.
- This method involves a **forecast equation** and **two smoothing equations** (one for the level and one for the trend

$$\begin{aligned} \text{Forecast equation} \quad & \hat{y}_{t+h|t} = \ell_t + hb_t \\ \text{Level equation} \quad & \ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\ \text{Trend equation} \quad & b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \end{aligned}$$

where ℓ_t denotes an estimate of the level of the series at time t, b_t denotes an estimate of the trend (slope) of the series at time t, α is the smoothing parameter for the level, $0 \le \alpha \le 1$, and β^* is the smoothing parameter for the trend, $0 \le \beta^* \le 1$. (We denote this as β^* instead of β for

Autoregressive model

- In a multiple regression model, we forecast variable of interest using a linear combination of predictors.
- In an autoregression model, we forecast variable of interest using a **linear combination of *past values*** *of the variable*.
- The term ***autoregression*** indicates that it is a regression of variable against itself.
- an autoregressive model of order **p** can be written as

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

where ε_t is white noise. This is like a multiple regression but with *lagged values* of y_t as predictors. We refer to this as an **AR(p) model**, an autoregressive model of order p.

Autoregressive model

- Autoregressive models are flexible at handling a wide range of time series patterns
- Figure show series from an AR(1) model and an AR(2) model

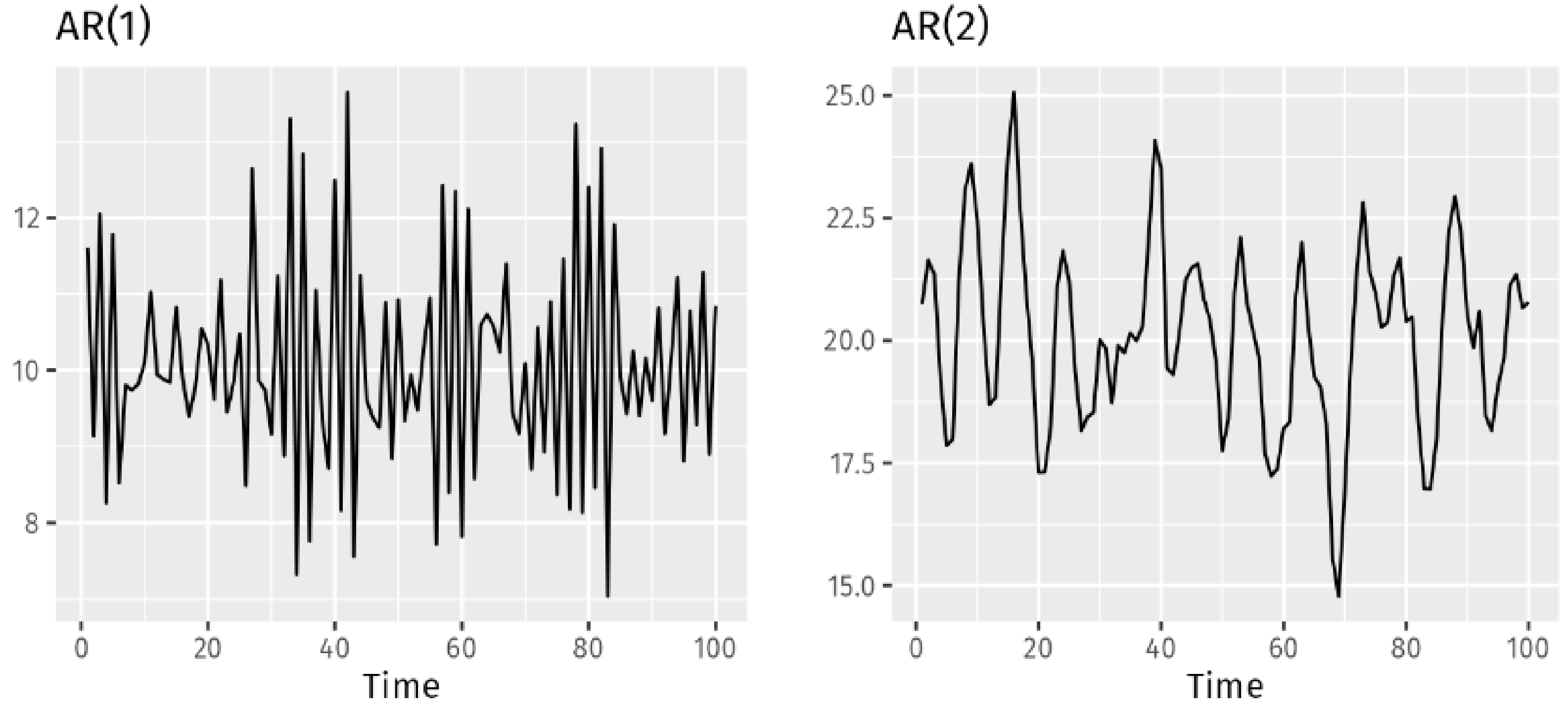


Figure 8.5: Two examples of data from autoregressive models with different parameters. Left: AR(1) with $y_t = 18 - 0.8y_{t-1} + \varepsilon_t$. Right: AR(2) with $y_t = 8 + 1.3y_{t-1} - 0.7y_{t-2} + \varepsilon_t$. In both cases, ε_t is normally distributed white noise with mean zero and variance one.

Autoregressive model

For an AR(1) model:

- when $\phi_1 = 0$, y_t is equivalent to white noise;
- when $\phi_1 = 1$ and $c = 0$, y_t is equivalent to a random walk;
- when $\phi_1 = 1$ and $c \neq 0$, y_t is equivalent to a random walk with drift;
- when $\phi_1 < 0$, y_t tends to oscillate around the mean.

We normally restrict autoregressive models to stationary data, in which case some constraints on the values of the parameters are required.

- For an AR(1) model: $-1 < \phi_1 < 1$.
- For an AR(2) model: $-1 < \phi_2 < 1$, $\phi_1 + \phi_2 < 1$, $\phi_2 - \phi_1 < 1$.

When $p \geq 3$, the restrictions are much more complicated. R takes care of these restrictions when estimating a model.

Moving average model(MA)

- Rather than using past values of the forecast variable in a regression, a moving average model uses **past forecast errors** in a regression-like model.

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

where ε_t is white noise. We refer to this as an **MA(q) model**, a moving average model of order q. Of course, we do not *observe* the values of ε_t, so it is not really a regression in the usual sense.

- each value of y_t is thought of as a weighted moving average of past few forecast errors.
- However, **moving average *models*** should not be confused with **moving average *smoothing***
- A moving average model is used for **forecasting future values**, while moving average smoothing is used for **estimating the trend-cycle** of past values.

Moving average model(MA)

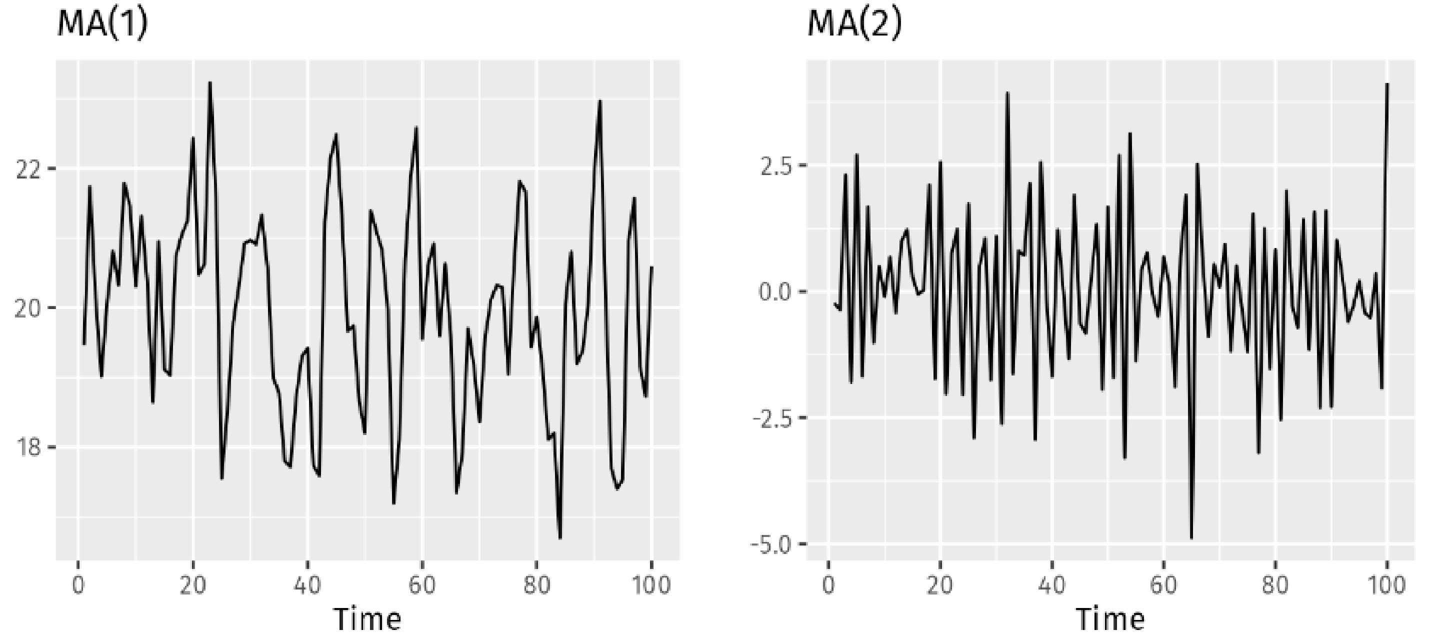


Figure 8.6: Two examples of data from moving average models with different parameters. Left: MA(1) with $y_t = 20 + \varepsilon_t + 0.8\varepsilon_{t-1}$. Right: MA(2) with $y_t = \varepsilon_t - \varepsilon_{t-1} + 0.8\varepsilon_{t-2}$. In both cases, ε_t is normally distributed white noise with mean zero and variance one.

ARIMA Model

- **ARIMA**, short for **AutoRegressive Integrated Moving Average**, is a forecasting algorithm based on the idea that the information in the past values of the time series can alone be used to predict the future values.
- Specified by three order parameters: (p, d, q),where,
    - p is the order of the AR term
    - q is the order of the MA term
    - d is the number of differencing required to make the time series stationary

Types of ARIMA Model

- **ARIMA** : Non-seasonal Autoregressive Integrated Moving Averages
- **SARIMA** : Seasonal ARIMA
- **SARIMAX** : Seasonal ARIMA with exogenous variables

If a time series, has seasonal patterns, then we need to add seasonal terms and it becomes SARIMA, short for **Seasonal ARIMA**.

Non Seasonal ARIMA Model

- If we combine differencing with **autoregression** and a **moving average model**, we obtain a non-seasonal ARIMA model.
- ARIMA is an acronym for Auto Regressive Integrated Moving Average ("integration" is reverse of differencing).
- The full model can be written as

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

- The "predictors" on the right hand side include both lagged values of yt and lagged errors called as **ARIMA(p, d, q)** Model where,

p=the order of the regressive part
d=degree of the first differencing involved
q=order of the moving average part

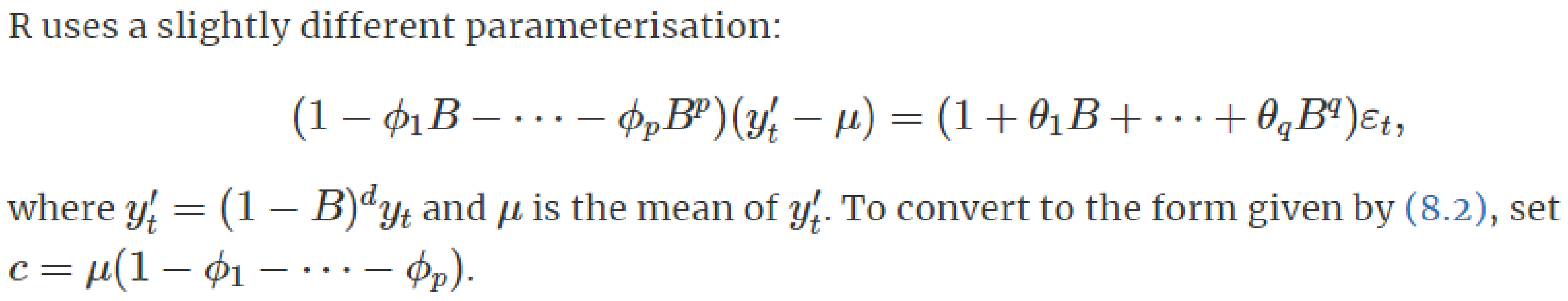
Special cases ARIMA Model

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)

Once we start combining components in this way to form more complicated models, it is much easier to work with the backshift notation.
Equation in backshift notation is now:-

$$\underset{\underset{\text{AR}(p)}{\uparrow}}{(1-\phi_1 B - \cdots - \phi_p B^p)} \quad \underset{\underset{d\text{ differences}}{\uparrow}}{(1-B)^d y_t} \quad = \quad c + \underset{\underset{\text{MA}(q)}{\uparrow}}{(1+\theta_1 B + \cdots + \theta_q B^q)\varepsilon_t}$$

ARIMA Model in R



R uses a slightly different parameterisation:

$$(1 - \phi_1 B - \cdots - \phi_p B^p)(y'_t - \mu) = (1 + \theta_1 B + \cdots + \theta_q B^q)\varepsilon_t,$$

where $y'_t = (1 - B)^d y_t$ and μ is the mean of y'_t. To convert to the form given by (8.2), set $c = \mu(1 - \phi_1 - \cdots - \phi_p)$.

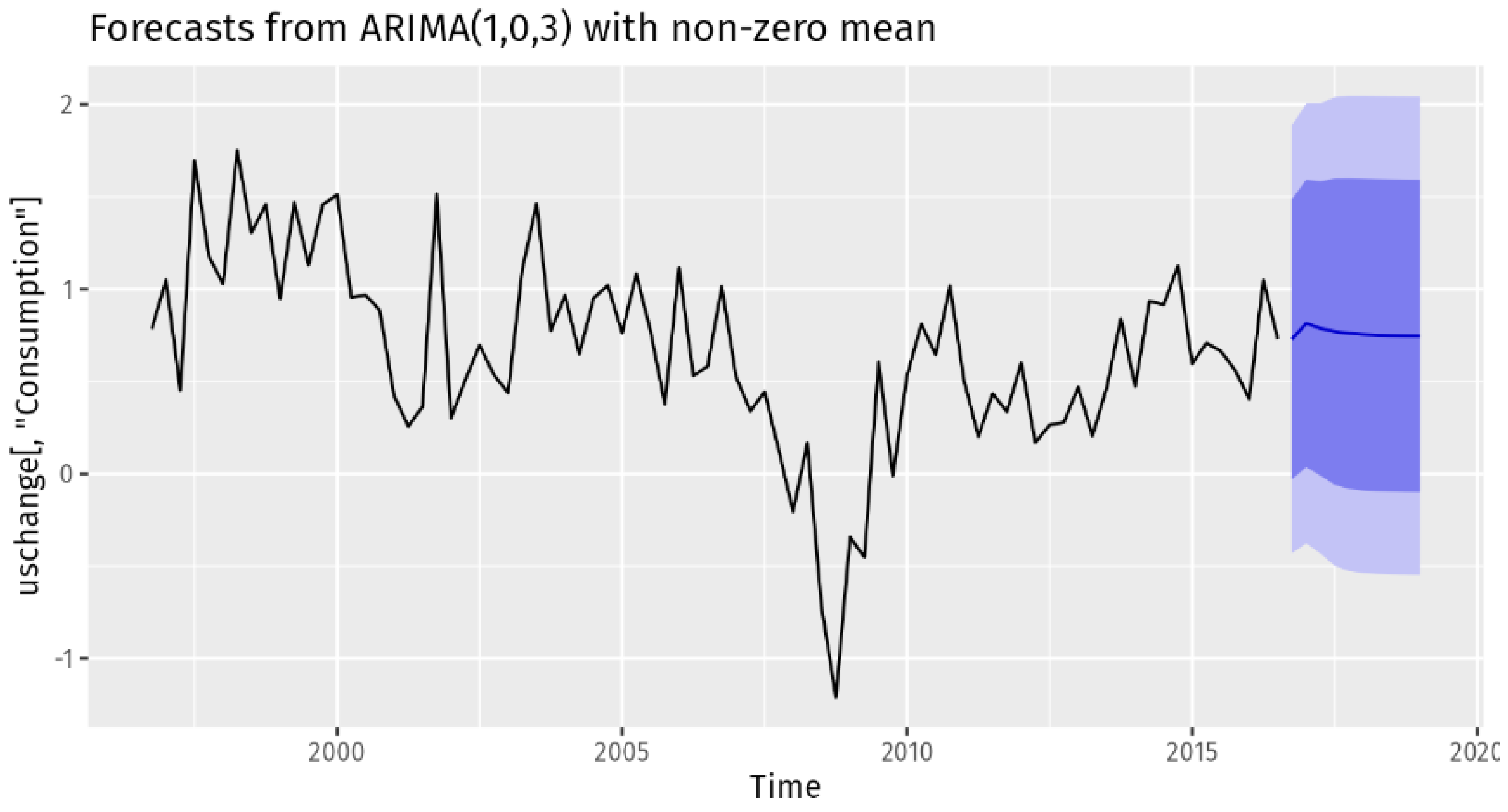
- Selecting values for p,d and q can be difficult.
- But, **auto.arima()** function in R does it for you automatically.

Understanding ARIMA Model(1)

- **auto.arima()** function is useful, but anything automated can be a little dangerous, and it is worth understanding something of the behaviour of the models
- The constant c has an important effect on the long-term forecasts obtained from these models.
- If **c=0 and d=0**, the long-term forecasts will go to zero.
- If **c=0 and d=1**, the long-term forecasts will go to a non-zero constant.
- If **c=0 and d=2**, the long-term forecasts will follow a straight line.
- If **c≠0 and d=0**, the long-term forecasts will go to the mean of the data.
- If **c≠0 and d=1**, the long-term forecasts will follow a straight line.
- If **c≠0 and d=2**, the long-term forecasts will follow a quadratic trend.

Understanding ARIMA Model(2)

- Value of d has an effect on prediction interval
- Higher the value of d , more rapidly prediction interval increase in size
- For d=0, long-term forecast standard deviation will go to the standard deviation of the historical data, so the prediction intervals will all be essentially the same.

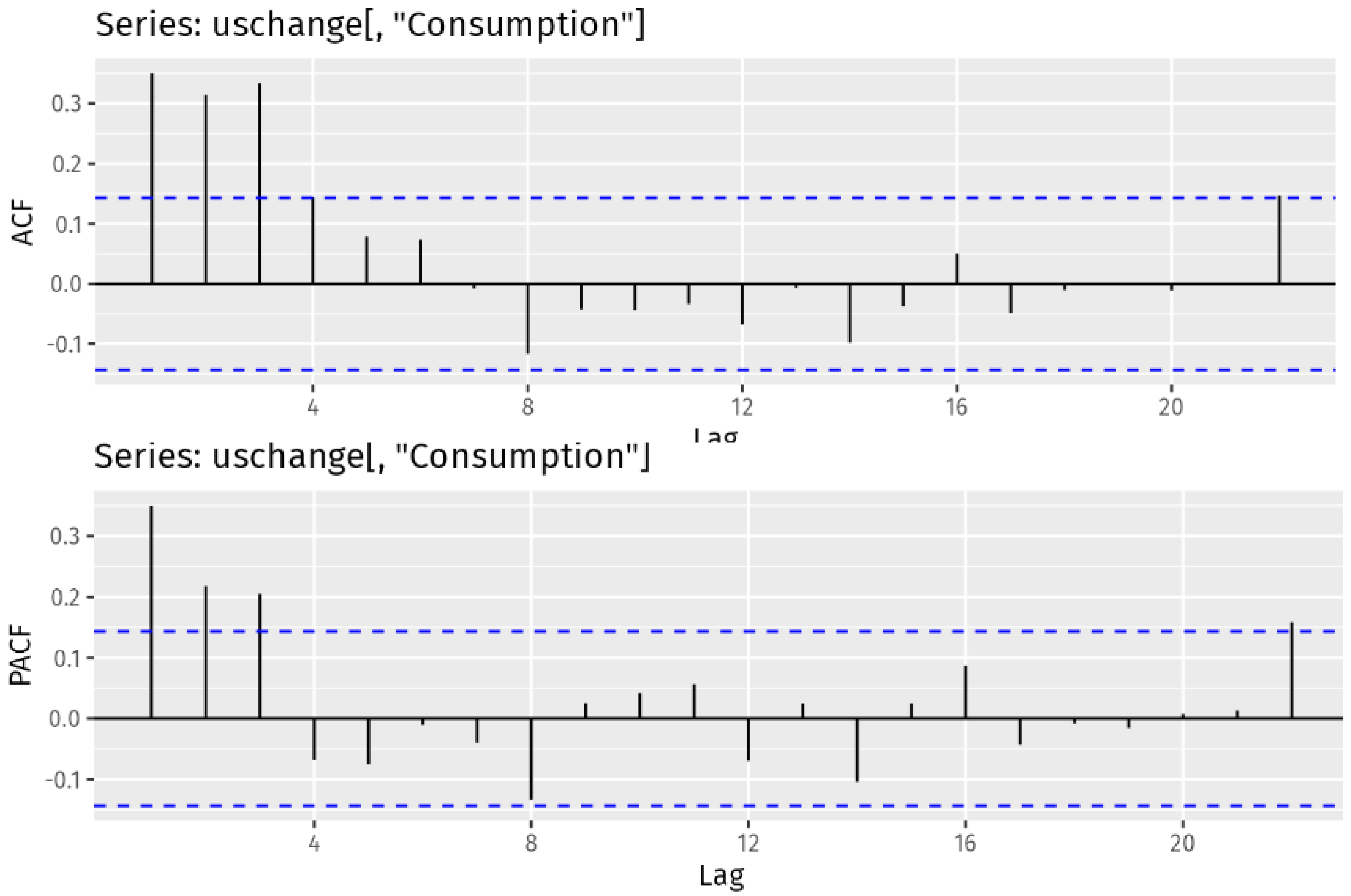


Understanding ARIMA Model(3)

- The value of **p** is important if the data show cycles.
- To obtain cyclic forecasts, it is necessary to have **$p \geq 2$**, along with some additional conditions on parameters
- It is usually not possible to tell, simply from a time plot, **what values of p and q** are appropriate for data.
- It is possible to use the ACF plot, and the closely related PACF plot, to determine appropriate values for p and q.

Showing ACF and PACF pot for US change data

Figures 8.9 and 8.10 shows the ACF and PACF plots for the US consumption data shown in Figure 8.7. The partial autocorrelations have the same critical values of $\pm 1.96/\sqrt{T}$ as for ordinary autocorrelations, and these are typically shown on the plot as in Figure 8.9.



- There are three spikes in the ACF, followed by an almost significant spike at lag 4.
- In PACF, there are three significant spikes, and then no significant spikes thereafter (apart from one just outside the bounds at lag 22).
- As per ACF and PACF, an ARIMA(3,0,0) model might be appropriate.

Choosing between ARIMA(p, d,0) and ARIMA(0,d,q)

- If the data are from an ARIMA(p,d,0) or ARIMA(0,d,q) model, then ACF and PACF plots can be helful in detrmining p and q.
- If p and q are both positive then plots do not help in finding suitable values of p and q.

Data may follow **ARIMA(p, d, 0)** if ACF and PACF plots of diffrenced data show patterns like:-

- ACF is exponentially decaying or sinusoidal
- There is significant lag at lag p in PACF but none beyond lag p.

Data may follow **ARIMA(0, d, q)** if ACF and PACF plots of diffrenced data show patterns like:-

- PACF is exponentially decaying or sinusoidal
- There is significant lag at lag q in ACF but none beyond lag q

Estimation and order selection

Maximum likelihood estimation-

- Once the model order has been identified (values of p, d and q), we need to estimate the parameters c, ϕ1,…,ϕp,…, θ1,…,θq.
- When R estimates the ARIMA model, it uses *maximum likelihood estimation* (MLE).
- This technique finds values of the parameters which maximise the probability of obtaining the data that we have observed.
- For ARIMA models, MLE is similar to the *least squares* estimates that would be obtained by minimizing,

$$\sum_{t=1}^{T} \varepsilon_t^2.$$

Information Criteria

- Akaike’s Information Criterion (AIC), which was useful in selecting predictors for regression, is also useful for determining the order of an ARIMA model.
- It can be written as

$$\text{AIC} = -2\log(L) + 2(p+q+k+1),$$

- where L is the likelihood of the data, k=1 if c≠0 and k=0 if c=0.
- Note that the last term in parentheses is the number of parameters in the model (including σ2, variance of the residuals).

- For ARIMA models, the corrected AIC can be written as

$$\text{AICc} = \text{AIC} + \frac{2(p+q+k+1)(p+q+k+2)}{T-p-q-k-2},$$

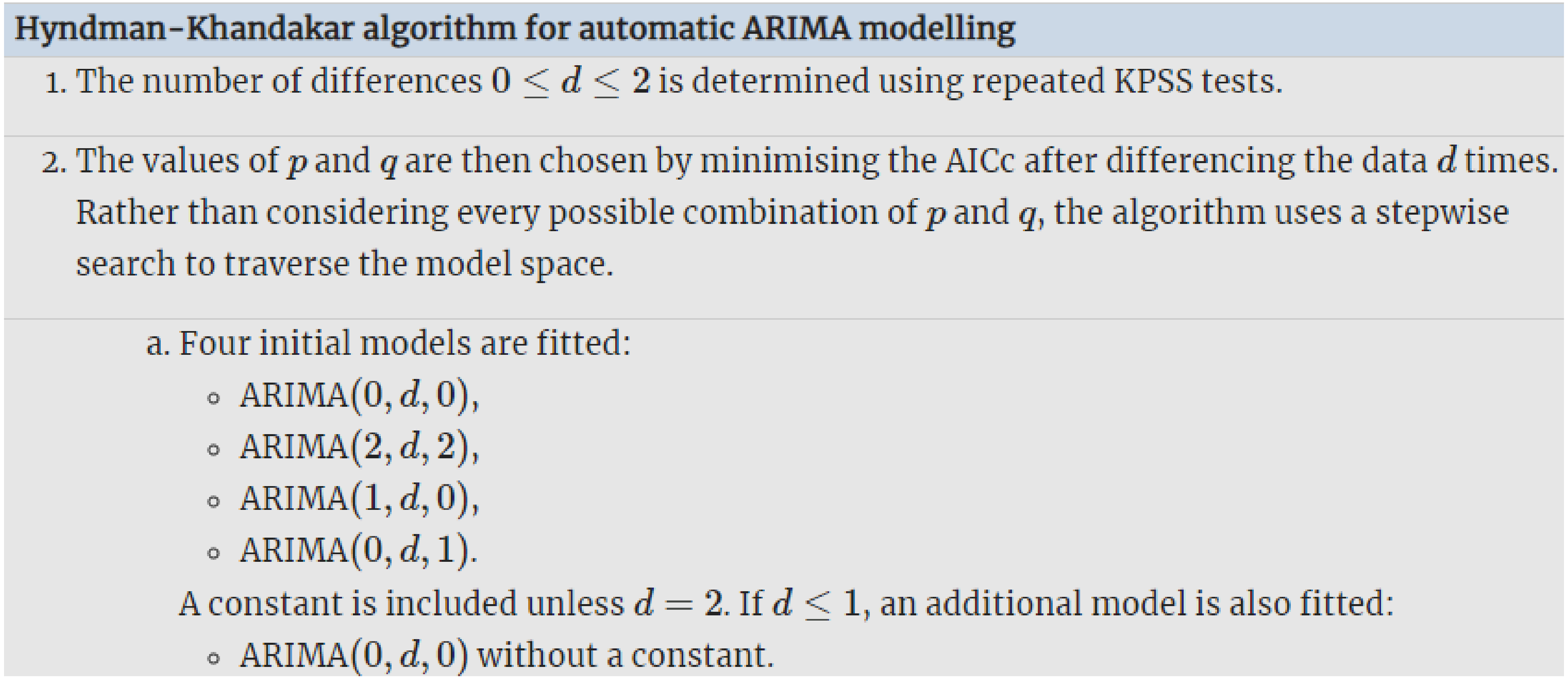
Bayesian Information Criterion

- **Bayesian information criterion** (**BIC**) or **Schwarz information criterion**
- It is a criterion for model selection among a finite set of models
- models with lower BIC are generally preferred

$$\mathrm{BIC} = \mathrm{AIC} + [\log(T) - 2](p + q + k + 1).$$

How does auto.arima() work?(1)

- auto.arima() function in R used a variation of the Hyndman-Khandakar algorithm
- Algorithm combines unit root tests, minimization of the AICc anf MLE to obtain an ARIMA model
- Argument to auto.arima() provide many variations on algorithms



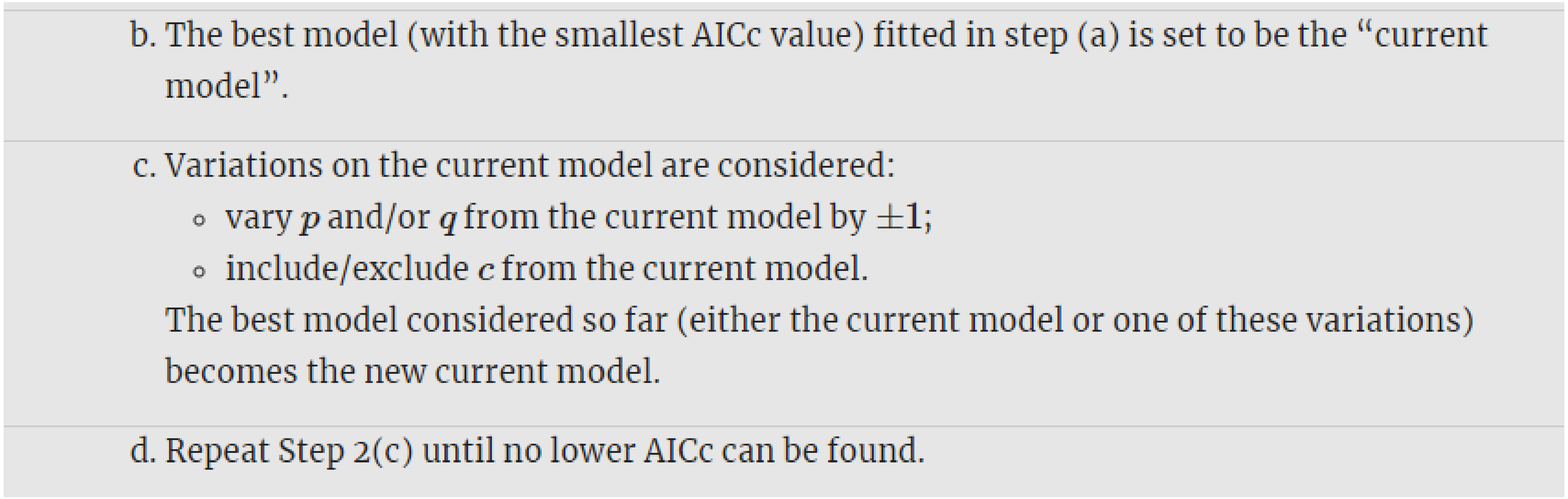
Hyndman-Khandakar algorithm for automatic ARIMA modelling

1. The number of differences $0 \leq d \leq 2$ is determined using repeated KPSS tests.
2. The values of p and q are then chosen by minimising the AICc after differencing the data d times. Rather than considering every possible combination of p and q, the algorithm uses a stepwise search to traverse the model space.

   a. Four initial models are fitted:
      - $\text{ARIMA}(0, d, 0)$,
      - $\text{ARIMA}(2, d, 2)$,
      - $\text{ARIMA}(1, d, 0)$,
      - $\text{ARIMA}(0, d, 1)$.

   A constant is included unless $d = 2$. If $d \leq 1$, an additional model is also fitted:
      - $\text{ARIMA}(0, d, 0)$ without a constant.

How does auto.arima() work?(2)



b. The best model (with the smallest AICc value) fitted in step (a) is set to be the "current model".

c. Variations on the current model are considered:

- vary p and/or q from the current model by ± 1;
- include/exclude c from the current model.

The best model considered so far (either the current model or one of these variations) becomes the new current model.

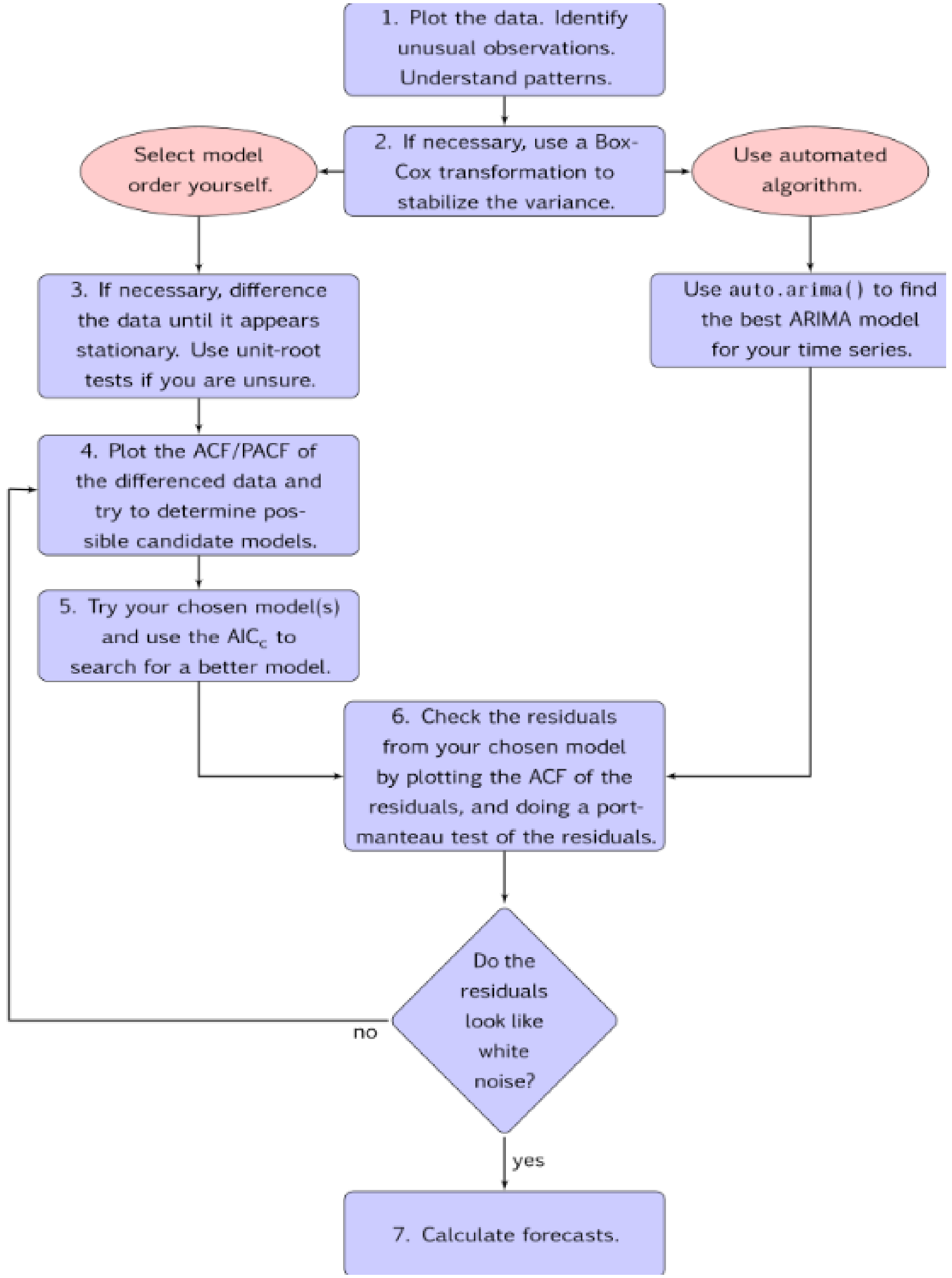
d. Repeat Step 2(c) until no lower AICc can be found.

Modeling Procedure

When fitting an ARIMA model to a set of (non-seasonal) time series data, the following procedure provides a useful general approach.

1. Plot the data and identify any unusual observations.
2. If necessary, transform the data (using a Box-Cox transformation) to stabilise 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, 0) or ARIMA(0,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.

Modeling Process


1. Plot the data. Identify unusual observations. Understand patterns.
2. If necessary, use a Box-Cox transformation to stabilize the variance.
Select model order yourself.
Use automated algorithm.
3. If necessary, difference the data until it appears stationary. Use unit-root tests if you are unsure.
Use auto.arima() to find the best ARIMA model for your time series.
4. Plot the ACF/PACF of the differenced data and try to determine possible candidate models.
5. Try your chosen model(s) and use the AIC_c 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.
Do the residuals look like white noise?
no
yes
7. Calculate forecasts.