

Chapter 8

State Space Models and Markov Switching Models



The state space methods or models provide a unified and flexible methodology and technology for handling a wide range of problems in time series analysis and are also applied in other fields including artificial intelligence. This chapter introduces the basic principle of state space methods and its application to SARIMAX modeling with Python, presents relationship between state space models and ARIMAX models using the local-level model, and lastly discusses the Markov switching model which is useful in econometrics and other disciplines.

8.1 State Space Models and Representations

The state space model or method is developed in Kalman (1960) and Kalman and Bucy (1961) for control engineering and actually is a very general model that subsumes a whole class of special cases of interest. State space modeling provides a unified approach to dealing with a wide range of problems in time series analysis. Since the 1970s, a large number of monographs and textbooks on state space modeling have been published. For example, Tsay and Chen (2019) gives a lot of examples to classical and novel applications of state space methods. Gómez (2016) uses the state space methods to analyze multivariate time series. Casals et al. (2016) discusses the theory and application of time series state space methods. Harvey (1989) has an online version published in 2014 and focuses on analysis and forecasting of structural time series models using state space approach. Durbin and Koopman (2012) is an excellent textbook on making time series analysis by state space methods. In addition, Many time series books include this topic. For example, Shumway and Stoffer (2017, Chapter 6), Brockwell and Davis (2016, Chapter 9), and Tsay (2010, Chapter 11) all have a chapter to address the state space methods and technology. In particular, for introduction to application of the

state space approach in the field of artificial intelligence, see, for instance, Russell and Norvig (2021, Chapter 14) or (Russell and Norvig, Chapter 15).

8.1.1 State Space Models

How is the state space model formulated? Suppose that there is a system, in which \mathbf{X}_t is the (unobserved) state vector of the system at time t and \mathbf{Y}_t is an observation or measurement vector at time t . A general state space model for the system consists of two equations: one describes how the system output (observation) is realized as a linear combination of different dynamic components and the other characterize the system dynamics, that is, the evolution of the system over time. Specifically, we give the following definition.

Definition 8.1 A linear state space model is of the form

$$\mathbf{X}_{t+1} = \mathbf{c}_t + \mathbf{F}_t \mathbf{X}_t + \mathbf{D}_t \mathbf{u}_t + \mathbf{R}_t \boldsymbol{\eta}_t, \quad \boldsymbol{\eta}_t \sim \mathbf{N}(\mathbf{0}, \mathbf{Q}_t), \quad (8.1)$$

$$\mathbf{Y}_t = \mathbf{d}_t + \mathbf{Z}_t \mathbf{X}_t + \mathbf{B}_t \mathbf{u}_t + \boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t \sim \mathbf{N}(\mathbf{0}, \mathbf{H}_t), \quad (8.2)$$

where \mathbf{X}_t is an m -dimensional (unobserved) state vector and called the state variables, \mathbf{Y}_t is a k -dimensional observation vector and known as the endogenous variables, \mathbf{u}_t is an n -dimensional input vector and referred to as the control or exogenous variables, $\boldsymbol{\eta}_t$ is an r -dimensional state error (disturbance or noise), and $\boldsymbol{\varepsilon}_t$ is a k -dimensional observation error (disturbance or noise). Equations (8.1) and (8.2) are, respectively, known as the state and observation equations.

Remarks about Definition 8.1:

- Equations (8.1) and (8.2) are sometimes also called the *transition equation* and *measurement equation*, respectively.
- The state intercept \mathbf{c}_t is an m -dimensional vector, and coefficients \mathbf{F}_t , \mathbf{D}_t , and \mathbf{R}_t are, respectively, $m \times m$, $m \times n$, and $m \times r$ matrices.
- The observation intercept \mathbf{d}_t is a k -dimensional vector, and coefficients \mathbf{Z}_t and \mathbf{B}_t are, respectively, $k \times m$ and $k \times n$ matrices.
- The state error covariance matrix \mathbf{Q}_t and observation error covariance matrix \mathbf{H}_t are, respectively, $r \times r$ and $k \times k$ positive semi-definite matrices.
- It is usually assumed that $\boldsymbol{\eta}_t$ and $\boldsymbol{\varepsilon}_t$ are uncorrelated, namely, $E(\boldsymbol{\eta}_s \boldsymbol{\varepsilon}_t') = \mathbf{0}$ for all s and t , but both of them may be correlated if needed. See, for example, Gómez (2016, Chapter 4).
- The initial state vector \mathbf{X}_1 is assumed to be uncorrelated with all of the error terms $\boldsymbol{\eta}_t$ and $\boldsymbol{\varepsilon}_t$ and often $\mathbf{X}_1 \sim \mathbf{N}(\mathbf{a}_1, \mathbf{P}_1)$. Moreover, the mean \mathbf{a}_1 and covariance matrix \mathbf{P}_1 of the distribution of \mathbf{X}_1 must be first specified, which is called *initialization*.
- In many important special cases, including ARIMA, VARMA, and so forth, the parameters \mathbf{c}_t , \mathbf{F}_t , \mathbf{D}_t , \mathbf{R}_t , \mathbf{Q}_t , \mathbf{d}_t , \mathbf{Z}_t , \mathbf{B}_t , and \mathbf{H}_t are all time-invariant, and

thus the time subscripts will be dropped. In this case, the state space model is also referred to as being time homogeneous.

Example 8.1 (CAPM with Time-Varying Coefficients) In finance, the capital asset pricing model (CAPM) provides a theoretical structure for the pricing of assets with risky returns and is used to determine an appropriate required rate of return of a financial asset. Now we consider a CAPM with time-varying coefficients as follows:

$$R_t = \alpha_t + \beta_t R_{M,t} + \varepsilon_t, \quad \varepsilon_t \sim \text{iidN}(0, \sigma_\varepsilon^2)$$

where R_t is the excess return of an asset and $R_{M,t}$ is the excess return of the market. What is more, it allows for the time-varying intercept α_t and slope β_t to evolve like a random walk. In other words, α_t and β_t , respectively, follow

$$\alpha_t = \alpha_{t-1} + \eta_t, \quad \eta_t \sim \text{iidN}(0, \sigma_\eta^2) \quad \text{and} \quad \beta_t = \beta_{t-1} + v_t, \quad v_t \sim \text{iidN}(0, \sigma_v^2)$$

where the innovations ε_t , η_t , and v_t are assumed to be mutually uncorrelated. If we let the state vector $\mathbf{X}_t = (\alpha_t, \beta_t)'$, then we can easily rewrite the CAPM as a state space model

$$\begin{aligned} \mathbf{X}_{t+1} &= \mathbf{X}_t + \boldsymbol{\eta}_t, \\ R_t &= [1, R_{M,t}] \mathbf{X}_t + \varepsilon_t \end{aligned}$$

where $\boldsymbol{\eta}_t = (\eta_{t+1}, v_{t+1})'$.

8.1.2 State Space Representations of Time Series

What is a state space form or representation for a time series? We give the following definition.

Definition 8.2 A vector time series \mathbf{Y}_t is said to have a state space form or representation if there is a state space model for \mathbf{Y}_t as specified by Eqs. (8.1) and (8.2).

There are a huge number of time series that can be represented in state space form. They include all the processes generated by SARIMA and VARMAX models in the previous chapters. Now let us have a look at a few examples.

Example 8.2 (State Space Form of the MA(1) Model) Given the following MA(1) model

$$Y_t = \mu + \varepsilon_t + \theta \varepsilon_{t-1},$$

and let $X_{t+1} = \theta \varepsilon_t$, then we obtain the representation of the MA(1) model as follows:

$$\begin{aligned} X_{t+1} &= 0 \cdot X_t + \theta \varepsilon_t, \\ Y_t &= \mu + X_t + \varepsilon_t. \end{aligned}$$

Example 8.3 (State Space Form of the ARMA(1,1) Model) Consider the following causal and invertible ARMA(1,1) model

$$Y_t = \varphi_0 + \varphi Y_{t-1} + \varepsilon_t + \theta \varepsilon_{t-1}. \quad (8.3)$$

Let the state variable $X_t = Y_t - \varepsilon_t$. Then we have

$$X_{t+1} = \varphi_0 + \varphi X_t + (\theta + \varphi) \varepsilon_t, \quad (8.4)$$

$$Y_t = X_t + \varepsilon_t. \quad (8.5)$$

These two equations form a state space representation for the ARMA(1,1) model (8.3) and are equivalent to it.

Note that there exist other state space representations for the ARMA(1,1) model. In fact, let $X_t = \varphi X_{t-1} + \varepsilon_t$, that is, an AR(1) process and the state vector $\mathbf{X}_t = (X_{t-1}, X_t)'$. Then we arrive at the equivalent model

$$\begin{bmatrix} X_t \\ X_{t+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \varphi \end{bmatrix} \begin{bmatrix} X_{t-1} \\ X_t \end{bmatrix} + \begin{bmatrix} 0 \\ \varepsilon_{t+1} \end{bmatrix}, \text{ namely, } \mathbf{X}_{t+1} = \begin{bmatrix} 0 & 1 \\ 0 & \varphi \end{bmatrix} \mathbf{X}_t + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \eta_t,$$

where $\eta_t = \varepsilon_{t+1}$. We say that this equation is simply the state equation for (8.3), and the observation equation is

$$Y_t = \varphi_0(1 - \varphi)^{-1} + \theta X_{t-1} + X_t = \varphi_0(1 - \varphi)^{-1} + [\theta \ 1] \mathbf{X}_t. \quad (8.6)$$

Actually, at this moment, due to $X_t = \varphi X_{t-1} + \varepsilon_t$, we have

$$\begin{aligned} Y_t - \varphi Y_{t-1} &= \varphi_0(1 - \varphi)^{-1} + \theta X_{t-1} + X_t - \varphi[\varphi_0(1 - \varphi)^{-1} + \theta X_{t-2} + X_{t-1}] \\ &= \varphi_0(1 - \varphi)^{-1}(1 - \varphi) + (\theta - \varphi)X_{t-1} + X_t - \varphi\theta X_{t-2} \\ &= \varphi_0 + (\theta - \varphi)X_{t-1} + X_t - \theta(X_{t-1} - \varepsilon_{t-1}) \\ &= \varphi_0 + X_t - \varphi X_{t-1} + \theta \varepsilon_{t-1} \\ &= \varphi_0 + \varepsilon_t + \theta \varepsilon_{t-1}. \end{aligned}$$

This means that Eq. (8.3) holds. Thus Eq. (8.6) is the observation equation for (8.3).

Example 8.3 illustrates that there may exist a few state space form for a single time series. In other words, state space representations are not unique. Furthermore,

it is worthy of noting that any linear state space model can be conversely translated into an equivalent VARMAX representation. See, for example, Casals et al. (2016, Chapter 9) and Gómez (2016).

8.2 Kalman Recursions

In this section, for simplicity and highlighting key ideas, we do not consider including the control variables in the state space model (8.1)–(8.2). That is, the state space model is now as follows:

$$\mathbf{X}_{t+1} = \mathbf{c}_t + \mathbf{F}_t \mathbf{X}_t + \mathbf{R}_t \boldsymbol{\eta}_t, \quad \boldsymbol{\eta}_t \sim \mathbf{N}(\mathbf{0}, \mathbf{Q}_t), \quad (8.7)$$

$$\mathbf{Y}_t = \mathbf{d}_t + \mathbf{Z}_t \mathbf{X}_t + \boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t \sim \mathbf{N}(\mathbf{0}, \mathbf{H}_t), \quad (8.8)$$

$$\mathbf{X}_1 \sim \mathbf{N}(\mathbf{a}_1, \mathbf{P}_1), \quad (8.9)$$

which is adequate for most purposes.

Once a state space model is specified for the observations $\mathbf{Y}_{1:t} = \{\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_t\}$, we can then consider a number of important algorithms and their applications. These algorithms are all recursion ones and often called the *Kalman recursions* due to the seminal papers by Kalman (1960) and Kalman and Bucy (1961). There are three fundamental problems associated with the state space model (8.7)–(8.9) as follows:

- *Kalman filtering*: Given $\mathbf{Y}_{1:t}$, to recover or estimate \mathbf{X}_t
- *Kalman forecasting*: Given $\mathbf{Y}_{1:t}$, to forecast \mathbf{X}_s where $s > t$
- *Kalman smoothing*: Given $\mathbf{Y}_{1:t}$, to estimate \mathbf{X}_s where $s < t$

For simplicity, we introduce the following notations. Let $\mathbf{X}_{h|j} = \mathbf{E}(\mathbf{X}_h | \mathbf{Y}_{1:j})$ and $\mathbf{P}_{h|j} = \text{Var}(\mathbf{X}_h | \mathbf{Y}_{1:j}) = \mathbf{E}[(\mathbf{X}_h - \mathbf{X}_{h|j})(\mathbf{X}_h - \mathbf{X}_{h|j})' | \mathbf{Y}_{1:j}]$ be, respectively, the conditional mean vector (viz., an estimate) and covariance matrix of \mathbf{X}_h given $\mathbf{Y}_{1:j}$, where we define the starting values $\mathbf{X}_{1|0} = \mathbf{a}_1$ and $\mathbf{P}_{1|0} = \mathbf{P}_1$ for $h = 1, j = 0$. Note that if $\mathbf{X}_{h|j}$ are viewed as the estimates of \mathbf{X}_h given $\mathbf{Y}_{1:j}$, then $\mathbf{P}_{h|j}$ is the covariance matrix of the estimation error. Furthermore, let $\mathbf{v}_h = \mathbf{Y}_h - \mathbf{Z}_h \mathbf{X}_{h|h-1} - \mathbf{d}_h$ be the estimate of $\boldsymbol{\varepsilon}_h$ and $\mathbf{V}_h = \mathbf{Z}_h \mathbf{P}_{h|h-1} \mathbf{Z}_h' + \mathbf{H}_h$ the conditional covariance matrix of \mathbf{v}_h given $\mathbf{Y}_{1:(h-1)}$. The following three theorems give a set of Kalman recursions to solve the three fundamental problems above.

Theorem 8.1 (Kalman Filtering) *Kalman filtering has the recursion algorithm*

$$\mathbf{X}_{t|t} = \mathbf{X}_{t|t-1} + \mathbf{P}_{t|t-1} \mathbf{Z}_t' \mathbf{V}_t^{-1} \mathbf{v}_t, \quad (8.10)$$

$$\mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \mathbf{P}_{t|t-1} \mathbf{Z}_t' \mathbf{V}_t^{-1} \mathbf{Z}_t \mathbf{P}_{t|t-1} \quad (8.11)$$

where $\mathbf{X}_{t|t}$ are the filtered estimates and $\mathbf{P}_{t|t}$ are the corresponding error covariance matrices. Besides, the conditional distribution of \mathbf{X}_t given $\mathbf{Y}_{1:t}$ is $N(\mathbf{X}_{t|t}, \mathbf{P}_{t|t})$.

Theorem 8.2 (Kalman Forecasting) The one-step-ahead predictors $\mathbf{X}_{t+1|t}$ of \mathbf{X}_{t+1} and error covariance matrices $\mathbf{P}_{t+1|t}$ are determined by the recursions

$$\mathbf{X}_{t+1|t} = \mathbf{c}_t + \mathbf{F}_t \mathbf{X}_{t|t}, \quad (8.12)$$

$$\mathbf{P}_{t+1|t} = \mathbf{F}_t \mathbf{P}_{t|t} \mathbf{F}_t' + \mathbf{R}_t \mathbf{Q}_t \mathbf{R}_t'. \quad (8.13)$$

In this case, the conditional distribution of \mathbf{X}_{t+1} given $\mathbf{Y}_{1:t}$ is $N(\mathbf{X}_{t+1|t}, \mathbf{P}_{t+1|t})$.

Theorem 8.3 (Kalman Smoothing) For $s = t, \dots, 2, 1$, the smoothed estimates $\mathbf{X}_{s|t}$ and error covariance matrices $\mathbf{P}_{s|t}$ are determined by the following backward recursions

$$\mathbf{r}_{s-1} = \mathbf{Z}_s' \mathbf{V}_s^{-1} \mathbf{v}_s + \mathbf{L}_s' \mathbf{r}_s \quad \text{with } \mathbf{r}_t = \mathbf{0},$$

$$\mathbf{X}_{s|t} = \mathbf{X}_{s|s-1} + \mathbf{P}_{s|s-1} \mathbf{r}_{s-1},$$

$$\mathbf{M}_{s-1} = \mathbf{Z}_s' \mathbf{V}_s^{-1} \mathbf{Z}_s + \mathbf{L}_s' \mathbf{M}_s \mathbf{L}_s \quad \text{with } \mathbf{M}_t = \mathbf{0},$$

$$\mathbf{P}_{s|t} = \mathbf{P}_{s|s-1} - \mathbf{P}_{s|s-1} \mathbf{M}_{s-1} \mathbf{P}_{s|s-1},$$

where $\mathbf{L}_s = \mathbf{F}_s - \mathbf{K}_s \mathbf{Z}_s$ and $\mathbf{K}_s = \mathbf{F}_s \mathbf{P}_{s|s-1} \mathbf{Z}_s' \mathbf{V}_s^{-1}$, which is called the Kalman gain (matrix) at time s .

Note that plugging Eq. (8.10) into Eq. (8.12), we obtain the one-step-ahead predictors of \mathbf{X}_{t+1}

$$\mathbf{X}_{t+1|t} = \mathbf{c}_t + \mathbf{F}_t (\mathbf{X}_{t|t-1} + \mathbf{P}_{t|t-1} \mathbf{Z}_t' \mathbf{V}_t^{-1} \mathbf{v}_t) = \mathbf{c}_t + \mathbf{F}_t \mathbf{X}_{t|t-1} + \mathbf{K}_t \mathbf{v}_t.$$

From the equation above, we see that the Kalman gain as a weight gives a kind of improvement of forecasting. In addition, considering the one-step-ahead forecast of the observation variable \mathbf{Y}_{t+1} given $\mathbf{Y}_{1:t}$ and using Eq. (8.8), we arrive at

$$\mathbf{Y}_{t+1|t} = \mathbf{E}(\mathbf{Y}_{t+1} | \mathbf{Y}_{1:t}) = \mathbf{d}_{t+1} + \mathbf{Z}_{t+1} \mathbf{X}_{t+1|t},$$

and the covariance matrix of the forecast error is

$$\begin{aligned} \text{Var}(\mathbf{Y}_{t+1} | \mathbf{Y}_{1:t}) &= \mathbf{E}[(\mathbf{Y}_{t+1} - \mathbf{Y}_{t+1|t})(\mathbf{Y}_{t+1} - \mathbf{Y}_{t+1|t})' | \mathbf{Y}_{1:t}] \\ &= \mathbf{E}[(\mathbf{Z}_{t+1}(\mathbf{X}_{t+1} - \mathbf{X}_{t+1|t}) + \mathbf{e}_{t+1})(\mathbf{X}_{t+1} - \mathbf{X}_{t+1|t})' \mathbf{Z}_{t+1}' + \mathbf{e}_{t+1}' | \mathbf{Y}_{1:t}] \\ &= \mathbf{Z}_{t+1} \mathbf{P}_{t+1|t} \mathbf{Z}_{t+1}' + \mathbf{H}_{t+1} \\ &= \mathbf{V}_{t+1}. \end{aligned}$$

Hence, the one-step-ahead forecast problem is completely solved.

These Kalman recursions provide the necessary and efficient algorithms to compute prediction in an online manner. As for the initialization and parameter estimation problems as well as the proofs of these theorems, they can be found in Durbin and Koopman (2012), Tsay (2010, Chapter 11), Harvey (1989), and so on. In what follows, we shall discuss some applications of the state space model and Kalman recursions.

8.3 Local-Level Model and SARIMAX Models

8.3.1 Local-Level Model

In this subsection, we consider a famous state space model, namely, the *local-level model* or *random walk plus noise model*. It takes the form

$$Y_t = \mu_t + \varepsilon_t, \quad \varepsilon_t \sim \text{iidN}(0, \sigma_\varepsilon^2), \quad (8.14)$$

$$\mu_{t+1} = \mu_t + \eta_t, \quad \eta_t \sim \text{iidN}(0, \sigma_\eta^2) \quad (8.15)$$

where $\{\varepsilon_t\}$ and $\{\eta_t\}$ are mutually uncorrelated and are independent of $\mu_1 \sim N(a_1, p_1)$. In some literature, μ_t is known as the trend of the series Y_t . Clearly, Y_t is stationary and has no trend if $\sigma_\eta = 0$ and μ_t is actually a stochastic trend of the series Y_t (also a random walk) if $\sigma_\eta \neq 0$. Moreover, $Y_t = \mu_t$ if $\sigma_\varepsilon = 0$. When $\sigma_\varepsilon \neq 0$, we first-difference the series Y_t and arrive at

$$(1 - B)Y_t = \eta_{t-1} + \varepsilon_t - \varepsilon_{t-1}.$$

Let $\xi_t = \eta_{t-1} + \varepsilon_t - \varepsilon_{t-1}$. Obviously, ξ_t is stationary and $\xi_t \sim N(0, 2\sigma_\varepsilon^2 + \sigma_\eta^2)$. Furthermore, the autocorrelations of ξ_t

$$\rho_k = \begin{cases} -\sigma_\varepsilon^2 / (2\sigma_\varepsilon^2 + \sigma_\eta^2) \neq 0 & \text{if } k = 1, \\ 0 & \text{if } k > 1. \end{cases}$$

This illustrates that ξ_t follows an MA(1) model (see Table 3.1), and therefore the local-level model (8.14)–(8.15) has an ARIMA(0,1,1) representation

$$(1 - B)Y_t = (1 + \theta B)\omega_t$$

where ω_t is a white noise series. This representation can be viewed as a special case of the ARIMAX model. And this is also an example to translating a state space model into an equivalent ARIMAX model.

Let $\sigma_\varepsilon = 2$ and $\sigma_\eta = 1$. Then it turns out that $\rho_1 = -2^2 / (2 \times 2^2 + 1) = -4/9$. At this point, we can simulate a sample of size 300 from the local-level model with $\sigma_\varepsilon = 2$ and $\sigma_\eta = 1$. Its time series plot is shown in Fig. 8.1, which displays

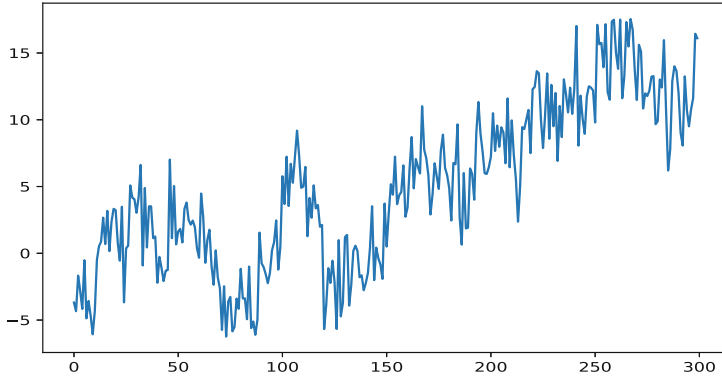


Fig. 8.1 Time plot of the sample simulated from the local-level model with $\sigma_\varepsilon = 2$ and $\sigma_\eta = 1$

 pyTSA_SimLLM

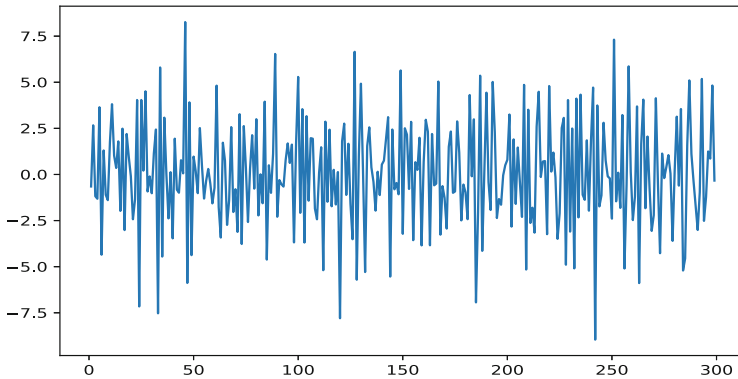


Fig. 8.2 Time series plot of the differences of the sample from the local-level model

 pyTSA_SimLLM

nonstationarity of the series. If we first-difference the sample and get the differenced series of the sample, then we can, respectively, draw the time plot and ACF plot of the differenced series shown in Figs. 8.2 and 8.3. We see that the time plot exhibits stationarity of the differenced series, the sample ACF value at lag 1 (viz., sample ρ_1) is around -0.4 (almost the same as the true value $-4/9$) and statistically significant, and at the same time, all the other ACF values are not significant.

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> from PythonTsa.RandomWalk import RandomWalk_with_drift
>>> from PythonTsa.plot_acf_pacf import acf_pacf_fig
>>> np.random.seed(1379)
>>> rw0= RandomWalk_with_drift(drift=0.0, nsample=300, burnin=10)
>>> wn=np.random.normal(loc=0, scale=2.0, size=300)
>>> y=rw0+wn
>>> y.plot();plt.show()
```

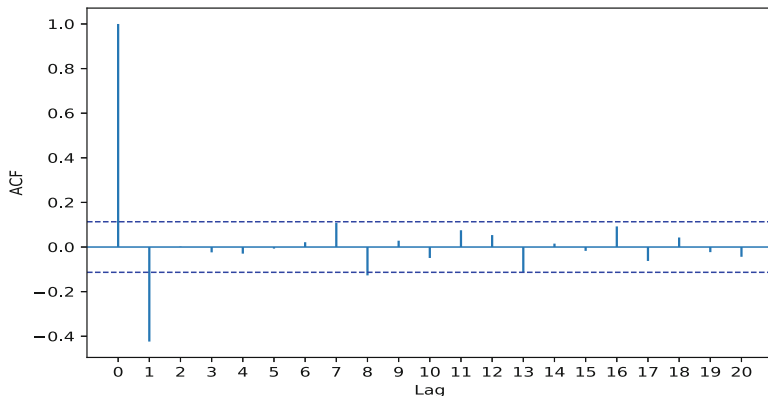



Fig. 8.3 ACF plot of the differenced series of the sample from the local-level model

 pyTSA_SimLLM

```
>>> dy=y.diff().dropna()
>>> dy.plot();plt.show()
>>> acf_pacf_fig(dy, both=False, lag=20)
>>> plt.show()
```

The local-level model is a simple case of the structural time series model (or unobserved component model). The interested reader can refer to Durbin and Koopman (2012) and Harvey (1989), among others.

8.3.2 SARIMAX Models

If we consider outer factors' impact on a time series, then we need to import exogenous regressors, namely, input variables. SARIMAX models are such a kind of models as to describe functionality of input variables. They have a few forms and we here introduce an often used form as follows:

$$\begin{cases} X_t = \beta_t' \mathbf{Y}_t + Z_t, & (8.16a) \end{cases}$$

$$\begin{cases} \varphi(B)\Phi(B^s)(1-B)^d(1-B^s)^D Z_t = \theta(B)\Theta(B^s)\varepsilon_t, \varepsilon_t \sim \text{WN}(0, \sigma_\varepsilon^2) & (8.16b) \end{cases}$$

where X_t is a univariate time series considered, $\mathbf{Y}_t = (Y_{t1}, Y_{t2}, \dots, Y_{tk})'$ is the input variables (exogenous regressors), $\beta_t = (\beta_{t1}, \beta_{t2}, \dots, \beta_{tk})$ is the coefficients corresponding to \mathbf{Y}_t , and Z_t is the regressing error. And Eq. (8.16b) is a SARIMA(p, d, q) (P, D, Q)_s model presented in Chap. 5. Thus Eqs. (8.16a)–(8.16b) are a regression with SARIMA errors. Note that in many cases, $\beta_t = \beta$ is time-invariant. Now let us look at an example to implementing SARIMAX modeling with Python. The Python function used here is `SARIMAX()` in the following module `statsmodels.tsa.statespace.sarimax`.

Example 8.4 (SARIMAX Model Building) The dataset “USEconomicChange.csv” in the folder Ptsadata is a time series that consists of changes (viz., growth rates) of the five US macroeconomic variables from the first quarter of 1970 to the third quarter of 2016. The five variables are consumption (cons), income (inc), production (prod), savings (sav), and unemployment (unem). First of all, we observe the correlation plots of the five variables shown in Fig. 8.4. It turns out that the five variables should be stationary as a five-dimensional vector series. In the light of economic common sense, we select the consumption variable cons as the endogenous (dependent) variable and all the others, namely, inc, prod, sav, and unem as the exogenous regressors. In the function SARIMAX(), let the parameters endog = cons and exog = {inc, prod, sav, unem}, and by trial and error, we choose order = (1,0,1) for the ARIMA(p, d, q) model. At the same time, all the other parameters are by default. Thus the ARMA model will be written as a state space Harvey representation. For more details on the Harvey representation, see Durbin and Koopman (2012) and Harvey (1989), among others. The estimated SARIMAX model is shown in the SARIMAX Results table in the following Python code. We see that the estimated coefficients of inc and prod are positive and the estimated coefficients of sav and unem negative. This is economically desirable. Furthermore, the ACF plot and p -value plot for Ljung-Box test of the residuals of the estimated SARIMAX model are, respectively, shown in Figs. 8.5 and 8.6. They clearly illustrate that the residual series can be viewed as a white noise and hence the estimated model should be adequate. However, we unexpectedly find that there exist autocorrelations (ARCH effect) in the squared residual series when we draw the ACF plot and p -value plot for Ljung-Box test of the squared residuals shown in Figs. 8.7 and 8.8. Therefore we need to continue to model the residuals of the estimated SARIMAX model. We fit a GARCH model to the residuals and arrive at the estimated GARCH(1,1) model shown in the Zero Mean - GARCH Model Results table in the following Python code. The p -value plots for Ljung-Box test of the residuals and squared residuals of the estimated GARCH(1,1) model are, respectively, shown in Figs. 8.9 and 8.10. They suggest that the residuals and squared residuals can be both seen as white noise series and there are no autocorrelations in them anymore. In conclusion, we build a complete and appropriate model for the given dataset as follows:

$$\begin{cases} cons_t = 0.253 + 0.731inc_t + 0.051prod_t - 0.046sav_t - 0.169unem_t + Z_t, \\ Z_t + 0.815Z_{t-1} = \eta_t + 0.729\eta_{t-1}, \\ \eta_t = \sigma_t\varepsilon_t, \varepsilon_t \sim \text{iidN}(0, 1), \\ \sigma_t^2 = 0.004 + 0.195\eta_{t-1}^2 + 0.776\sigma_{t-1}^2. \end{cases}$$

```
>>> import numpy as np
>>> import pandas as pd
>>> import matplotlib.pyplot as plt
>>> import statsmodels.api as sm
>>> from PythonTsa.plot_acf_pacf import acf_pacf_fig
```

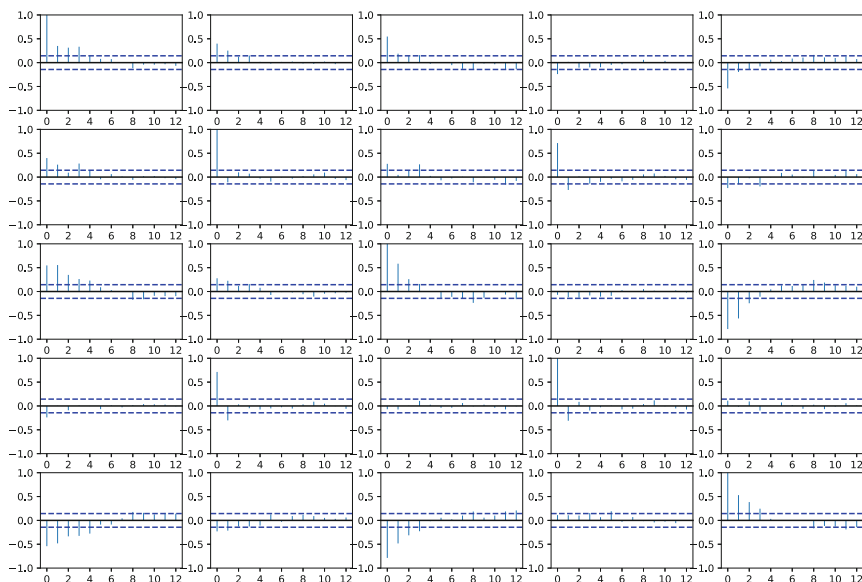


Fig. 8.4 Correlation plots of the US macroeconomic change time series in Example 8.4

 pyTSA_EconUS

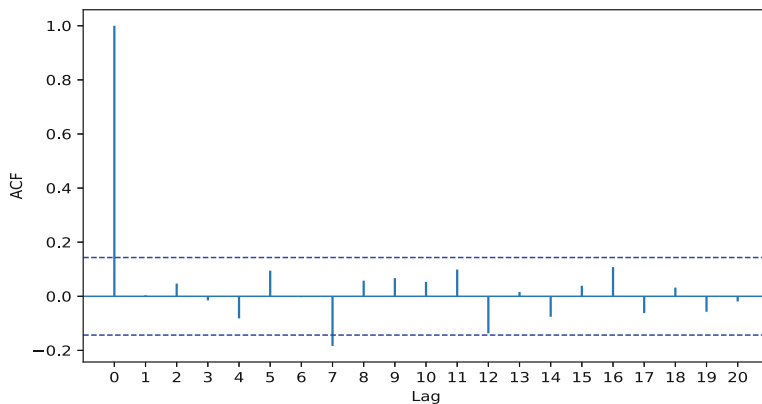


Fig. 8.5 ACF plot of the residual series of the SARIMAX model in Example 8.4

 pyTSA_EconUS

```
>>> from PythonTsa.LjungBoxtest import plot_LB_pvalue
>>> from PythonTsa.plot_multi_ACF import multi_ACFfig
>>> from statsmodels.tsa.statespace.sarimax import SARIMAX
>>> from arch import arch_model
>>> from PythonTsa.datadir import getdtapath
>>> dtapath=getdtapath()
>>> uscc=pd.read_csv(dtapath + 'USEconomicChange.csv')
```

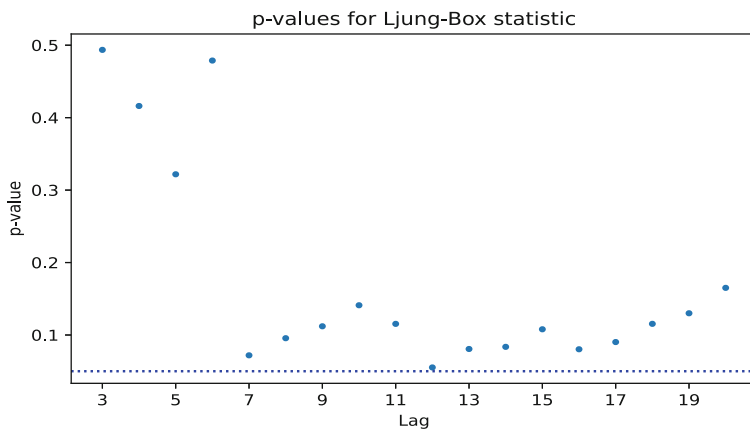


Fig. 8.6 *p*-Values for the Ljung-Box test of the residuals of the SARIMAX model in Example 8.4

 pyTSA_EconUS

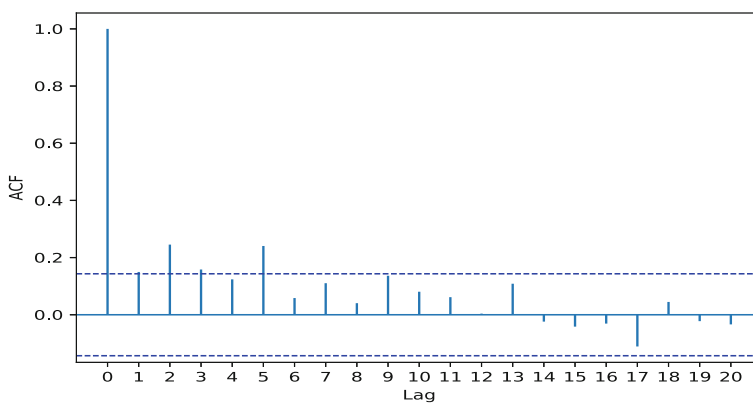


Fig. 8.7 ACF plot of the squared series of the residuals of the SARIMAX model in Example 8.4

 pyTSA_EconUS

```
>>> dates=pd.date_range('1970/3/31', periods=len(uscc), freq='Q')
>>> uscc.index=dates
>>> uscc=uscc.drop(columns=['Time'])
>>> uscc.rename(columns={'Consumption':'cons', 'Income':'inc',
'Production':'prod', 'Savings':'sav', 'Unemployment':'unem'},
inplace=True)
>>> multi_ACFfig(uscc, nlags=12)
>>> plt.show()
>>> X=uscc[['inc', 'prod', 'sav', 'unem']]
>>> Y=uscc['cons']
>>> X=sm.add_constant(X, prepend=False)
>>> sarimaxmod=SARIMAX(endog=Y, exog=X, order=(1,0,1))
```

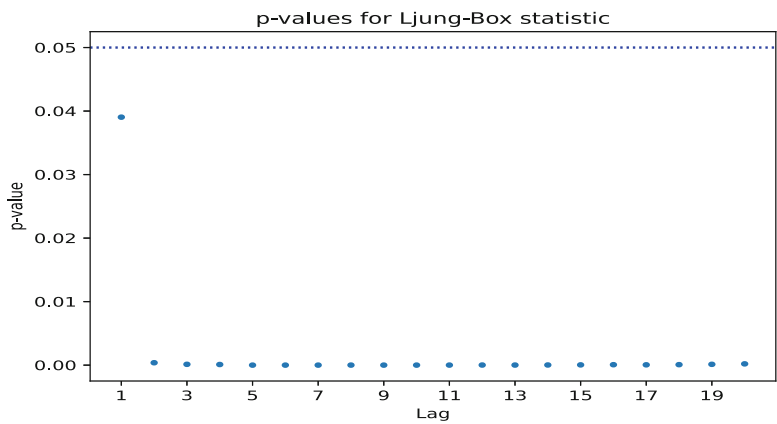


Fig. 8.8 *p*-Values for the Ljung-Box test of the squared residuals of the SARIMAX model in Example 8.4

 pyTSA_EconUS

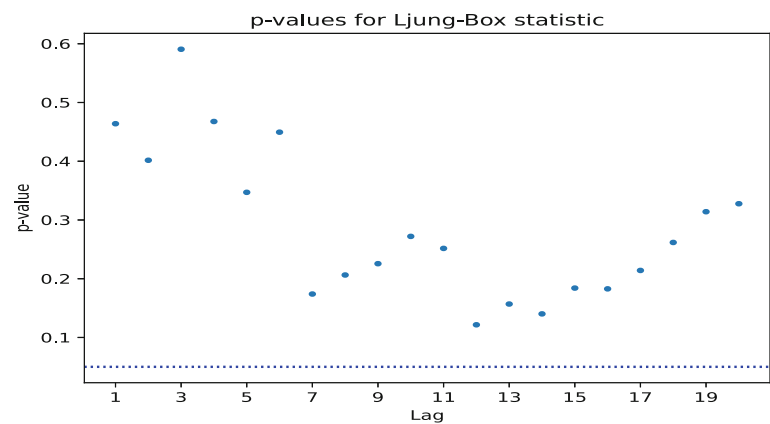


Fig. 8.9 *p*-Values for the Ljung-Box test of the residuals of the GARCH model in Example 8.4

 pyTSA_EconUS

```
>>> sarimaxfit=sarimaxmod.fit(dis=False)
>>> print(sarimaxfit.summary())
SARIMAX Results
=====
Dep. Variable:      cons      No. Observations:      187
Model:              SARIMAX(1, 0, 1)  Log Likelihood      -52.873
Date:              Sat, 04 Dec 2021  AIC              121.746
Time:              20:15:53          BIC              147.595
Sample:            03-31-1970       HQIC             132.220
                  - 09-30-2016
Covariance Type:    opg
```

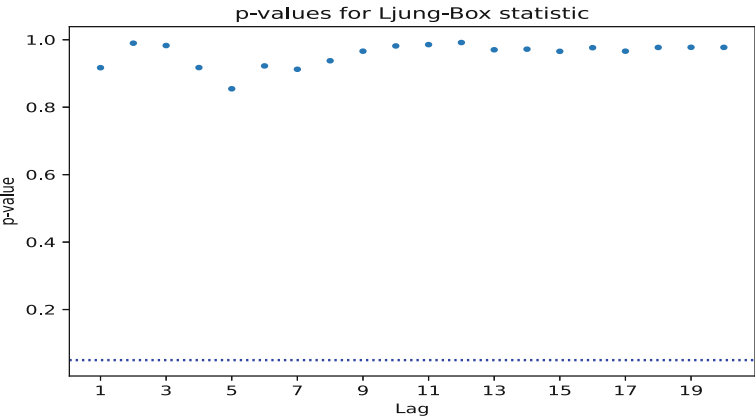


Fig. 8.10 *p*-Values for the Ljung-Box test of the squared residuals of the GARCH model in Example 8.4



	coef	std err	z	P> z	[0.025	0.975]
inc	0.7313	0.033	22.148	0.000	0.667	0.796
prod	0.0506	0.027	1.898	0.058	-0.002	0.103
sav	-0.0460	0.002	-23.691	0.000	-0.050	-0.042
unem	-0.1690	0.128	-1.322	0.186	-0.420	0.082
const	0.2534	0.038	6.589	0.000	0.178	0.329
ar.L1	-0.8148	0.218	-3.743	0.000	-1.242	-0.388
ma.L1	0.7290	0.257	2.836	0.005	0.225	1.233
sigma2	0.1030	0.009	11.474	0.000	0.085	0.121

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	21.79
Prob(Q):	0.96	Prob(JB):	0.00
Heteroskedasticity (H):	0.51	Skew:	0.52
Prob(H) (two-sided):	0.01	Kurtosis:	4.32

```
Warnings:
[1] Covariance matrix calculated using the outer product
    of gradients (complex-step).
>>> sarimaxresid=sarimaxfit.resid
>>> acf_pacf_fig(sarimaxresid, both=False, lag=20)
>>> plt.show()
>>> plot_LB_pvalue(sarimaxresid, noestimatedcoef=2, nolags=20)
>>> plt.show()
# the residuals are of Model (8.16b) and so noestimatedcoef=2.
>>> acf_pacf_fig(sarimaxresid**2, both=False, lag=20)
>>> plt.show()
>>> plot_LB_pvalue(sarimaxresid**2, noestimatedcoef=0, nolags=20)
>>> plt.show()
>>> archmod = arch_model(sarimaxresid, mean='Zero').fit(dispatch='off')
```

```
# mean='Zero' means no mean equation in GARCH models.
>>> print(archmod.summary())
```

Zero Mean - GARCH Model Results

```
=====
Dep. Variable:          None    R-squared:                0.000
Mean Model:             Zero Mean  Adj. R-squared:          0.005
Vol Model:              GARCH    Log-Likelihood:         -35.8494
Distribution:           Normal   AIC:                   77.6988
Method:                Maximum Likelihood  BIC:                   87.3921
                                     No. Observations:        187
Date:                  Sat, Dec 04 2021    Df Residuals:          184
Time:                  21:42:20           Df Model:              3
                                     Volatility Model
=====
               coef    std err          t      P>|t|      95.0% Conf. Int.
-----
omega  4.0515e-03  3.711e-03   1.092    0.275  [-3.222e-03, 1.133e-02]
alpha[1]  0.1950  7.952e-02   2.453  1.418e-02  [3.917e-02,  0.351]
beta[1]   0.7756  7.608e-02  10.195  2.078e-24  [ 0.627,  0.925]
=====
Covariance estimator: robust
>>> archresid = archmod.std_resid
>>> plot_LB_pvalue(archresid, noestimatedcoef=0, nolags=20)
>>> plt.show()
>>> plot_LB_pvalue(archresid**2, noestimatedcoef=0, nolags=20)
>>> plt.show()
```

8.4 Markov Switching Models

The Markov switching model is a flexible class of nonlinear time series models. It has been popular, especially in economic and business cycle analysis since the publication of Hamilton (1989). This section will present the model in brief.

8.4.1 Definitions

The Markov switching model is widely used in econometrics and other disciplines. It involves multiple structures that characterize the time series variable behaviors in different regimes and permits switching between these structures. The Markov switching model provides such a kind of switching mechanism so that it is controlled by an unobservable state variable which follows a first-order Markov chain. It is sometimes also referred to as the (*Markov*) *regime switching model*. Frühwirth-Schnatter (2006) gives a detailed Bayesian analysis of this model. Kim and Nelson (1999) elaborate on a class of state space models with Markov switching. For a time

series X_t , a general Markov switching model is of the form

$$X_t = \mu_{S_t} + \mathbf{Y}_t \boldsymbol{\beta}_{S_t} + \varphi_{1,S_t}(X_{t-1} - \mu_{S_{t-1}} - \mathbf{Y}_{t-1} \boldsymbol{\beta}_{S_{t-1}}) + \cdots \\ + \varphi_{p,S_t}(X_{t-p} - \mu_{S_{t-p}} - \mathbf{Y}_{t-p} \boldsymbol{\beta}_{S_{t-p}}) + \varepsilon_t, \quad \varepsilon_t \sim \text{WN}(0, \sigma_{S_t}^2) \quad (8.17)$$

where $\mathbf{Y}_t = (Y_{t1}, Y_{t2}, \dots, Y_{tk})$ are the input variables (exogenous regressors), $\boldsymbol{\beta}_{S_t}$ are the corresponding regression coefficient vectors, and S_t is a Markov chain for regimes. Here a regime is viewed as a state that the Markov chain may take. Assume that all the possible values of state variable S_t are $S = \{1 : K\}$. Then S_t satisfies the Markov property

$$P(S_t = j | S_{t-1} = i, S_{t-2} = s_{t-2}, \dots, S_1 = s_1) = P(S_t = j | S_{t-1} = i) = p_{ij}$$

where $j, i, s_{t-2}, \dots, s_1 \in S$ and p_{ij} is called the *one-step transition probability* from state i to state j . And the state transition is governed by the (state) transition (probability) matrix $\mathbf{P} = [p_{ij}]$. Besides, the error ε_t is also written as $\varepsilon_t = \sigma_{S_t} \eta_t$ where $\eta_t \sim \text{WN}(0, 1)$. And it is often assumed that the error ε_t is normally distributed, namely, $\varepsilon_t \sim \text{iidN}(0, \sigma_{S_t}^2)$.

If there are no exogenous regressors in Eq. (8.17), it is reduced to

$$X_t = \mu_{S_t} + \varphi_{1,S_t}(X_{t-1} - \mu_{S_{t-1}}) + \cdots + \varphi_{p,S_t}(X_{t-p} - \mu_{S_{t-p}}) + \varepsilon_t, \quad \varepsilon_t \sim \text{WN}(0, \sigma_{S_t}^2)$$

and called the *Markov switching autoregressive model*. Hamilton (1989) uses a special case of the Markov switching autoregressive model to study the business cycle dynamic of US GDP growth rates. And if there are no autoregressive terms in Eq. (8.17), it is reduced to

$$X_t = \mu_{S_t} + \mathbf{Y}_t \boldsymbol{\beta}_{S_t} + \varepsilon_t, \quad \varepsilon_t \sim \text{WN}(0, \sigma_{S_t}^2) \quad (8.18)$$

and known as the *Markov switching (dynamic) regression model*. Moreover, in some applications, the autoregressive coefficients have no switching, and then the Markov switching model is reduced to

$$X_t = \mu_{S_t} + \mathbf{Y}_t \boldsymbol{\beta}_{S_t} + \varphi_1(X_{t-1} - \mu_{S_{t-1}} - \mathbf{Y}_{t-1} \boldsymbol{\beta}_{S_{t-1}}) + \cdots \\ + \varphi_p(X_{t-p} - \mu_{S_{t-p}} - \mathbf{Y}_{t-p} \boldsymbol{\beta}_{S_{t-p}}) + \varepsilon_t, \quad \varepsilon_t \sim \text{WN}(0, \sigma_{S_t}^2). \quad (8.19)$$

Naturally, the variance of the error ε_t may also have no switching: $\sigma_{S_t}^2 = \sigma^2$. Nevertheless, for the Markov switching model in Eq. (8.17) to be meaningful, there must be a switching mechanism in at least one of the const term, the regression coefficients, the autoregressive coefficients, and the error variance. Furthermore, note that an important alternative to Eq. (8.17) is

$$X_t = \varphi_{0,S_t} + \varphi_{1,S_t} X_{t-1} + \cdots + \varphi_{p,S_t} X_{t-p} + \mathbf{Y}_t \boldsymbol{\beta}_{S_t} + \varepsilon_t, \quad \varepsilon_t \sim \text{WN}(0, \sigma_{S_t}^2). \quad (8.20)$$

This is another form of the Markov switching model. For more information on it, see Tsay and Chen (2019), McCulloch and Tsay (1994), and so forth.

Example 8.5 (A Two-State Markov Switching Model) Consider the following two-state Markov switching model with the autoregressive order 1 and one exogenous regressor:

$$X_t = \mu_{S_t} + Y_t \beta_{S_t} + \varphi_{S_t} (X_{t-1} - \mu_{S_{t-1}} - Y_{t-1} \beta_{S_{t-1}}), \quad \varepsilon_t \sim \text{WN}(0, \sigma_{S_t}^2).$$

If the two states are $S = \{0, 1\}$, then the state transition of the model is driven by the transition probabilities

$$P(S_t = 1 | S_{t-1} = 0) = a, \quad P(S_t = 0 | S_{t-1} = 1) = b$$

where $0 < a, b < 1$. And the transition matrix is

$$\mathbf{P} = \begin{bmatrix} 1-a & a \\ b & 1-b \end{bmatrix}.$$

In this case, we need to estimate the parameters $a, b, \mu_0, \mu_1, \beta_0, \beta_1, \varphi_0, \varphi_1, \sigma_0^2$, and σ_1^2 for building the two-state Markov switching model.

8.4.2 Examples

On theoretical discussion of the estimation issue of the Markov switching model, the reader can refer to Tsay and Chen (2019), Frühwirth-Schnatter (2006), and Kim and Nelson (1999), among others. As for carrying out estimation of the Markov switching model, there are two Python functions (classes): `MarkovRegression` and `MarkovAutoregression` available in the package `statsmodels`. The function `MarkovAutoregression` is used to estimate the model in Eq. (8.17) and `MarkovRegression` to the Markov switching regression model in Eq. (8.18). Note that in order to estimate the model in Eq. (8.20), we can view the lagged dependent variables X_{t-1}, \dots, X_{t-p} as a part of the exogenous regressors and then use the function `MarkovRegression`. What follows, we give a few examples to applications of the Markov switching model.

Example 8.6 (Log Returns of Germany DAX Daily Index) The dataset “DAXlogret” in the folder `Ptsadata` is the log return series of the Germany DAX daily index from October 24, 2014 to July 19, 2019 and denoted as `logret`. Its time series plot is shown in Fig. 8.11 and clearly displays heteroscedasticity. Thus we try to fit a Markov switching autoregressive model to the dataset. We find that we can select the order $p = 1$. The estimated model is shown in the table `Markov Switching Model Results` of the following Python code. Both

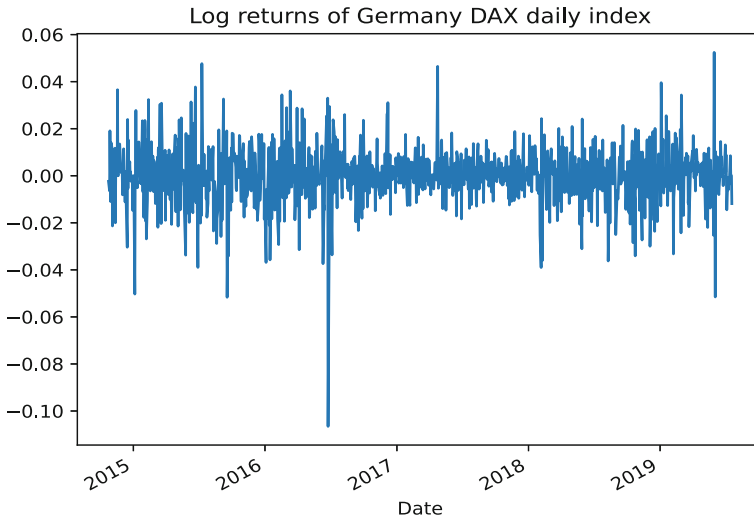


Fig. 8.11 Time series plot of the log returns of Germany DAX daily index in Example 8.6

 pyTSA_MarkovReturnsDAX

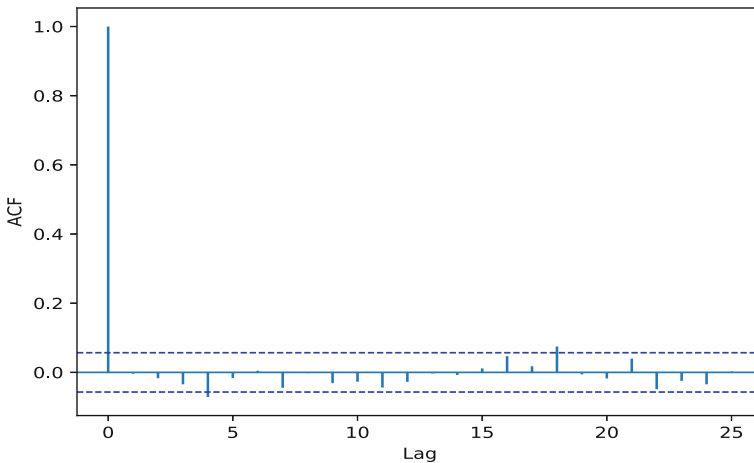


Fig. 8.12 ACF plot of the residuals of the estimated Markov switching model in Example 8.6

 pyTSA_MarkovReturnsDAX

ACF plot shown in Fig. 8.12 and p -value plot for the Ljung-Box test shown in Fig. 8.13 of the model residuals illustrate that the estimated model is appropriate.

```
>>> import numpy as np
>>> import pandas as pd
>>> import statsmodels.api as sm
>>> import matplotlib.pyplot as plt
```

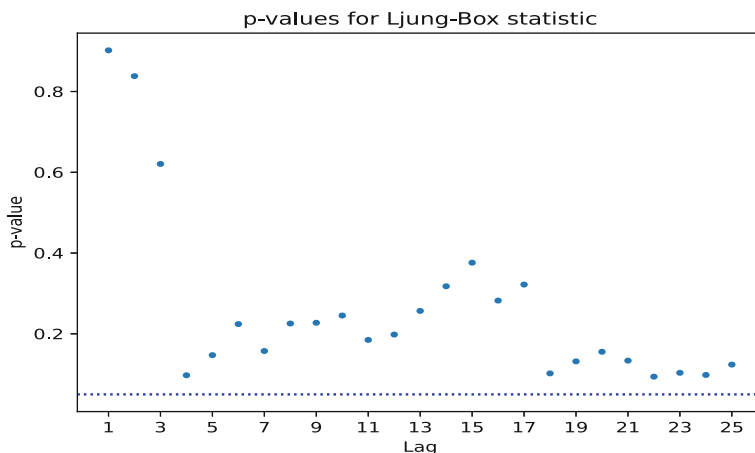


Fig. 8.13 *p*-Value plot for the Ljung-Box test of the model residual series in Example 8.6

 [pyTSA_MarkovReturnsDAX](#)

```
>>> from PythonTsa.plot_acf_pacf import acf_pacf_fig
>>> from PythonTsa.LjungBoxtest import plot_LB_pvalue
>>> from PythonTsa.datadir import getdtapath
>>> dtapath=getdtapath()
>>> daxlogret=pd.read_csv(dtapath + 'DAXlogret.csv',header=0)
>>> daxlogret.index=pd.DatetimeIndex(daxlogret.Date)
>>> logret=daxlogret.Logret
>>> logret.plot(title='Log returns of Germany DAX daily index')
>>> plt.show()
>>> mod= sm.tsa.MarkovAutoregression(logret, k_regimes=2,
>>>                                order=1, switching_variance=True)
>>> modfit=mod.fit()
>>> print(modfit.summary())
```

Markov Switching Model Results

```
=====
Dep. Variable:          Logret      No. Observations:      1190
Model:                MarkovAutoregression      Log Likelihood      3692.208
Date:                  Mon, 20 Dec 2021      AIC      -7368.417
Time:                  16:23:08      BIC      -7327.763
Sample:                0      HQIC      -7353.096
                        - 1190
```

Covariance Type: approx

Regime 0 parameters

```
=====
              coef      std err          z      P>|z|      [0.025      0.975]
-----
const          0.0008          0.000      2.643      0.008          0.000          0.001
sigma2  5.438e-05   5.55e-06      9.791      0.000      4.35e-05   6.53e-05
ar.L1       -0.0253          0.040     -0.630      0.529         -0.104          0.053
```

Regime 1 parameters

```
=====
              coef      std err          z      P>|z|      [0.025      0.975]
-----
const      -0.0013         0.001     -1.483     0.138     -0.003         0.000
sigma2       0.0003      3.44e-05      8.724     0.000         0.000         0.000
ar.L1       -0.1172         0.055     -2.125     0.034     -0.225     -0.009
              Regime transition parameters
=====
              coef      std err          z      P>|z|      [0.025      0.975]
-----
p[0->0]      0.9287         0.022     42.967     0.000         0.886         0.971
p[1->0]      0.1354         0.047      2.869     0.004         0.043         0.228
=====
Warnings:
[1] Covariance matrix calculated using numerical
    (complex-step) differentiation.
>>> modresid=modfit.resid
>>> acf_pacf_fig(modresid, both=False, lag=25); plt.show()
>>> plot_LB_pvalue(modresid, noestimatedcoef=0, nlags=25)
>>> plt.show()
```

Example 8.7 (Federal Funds Rate, Output Gap, and Inflation Rate) In this example, we are to analyze an American economic quarterly dataset that is built in the Python package `statsmodels`. It consists of three time series variables: `fedfunds`, `ogap`, and `inf`. Here `fedfunds` is the time series of federal funds rate, `ogap` the series of output gap (viz., GDP gap), and `inf` the series of inflation rate. The sample period for `fedfunds` and `ogap` is from July 1, 1954 to October 1, 2010, but `inf` is from July 1, 1955 to October 1, 2010. The time series plots of the three variables are shown in Fig. 8.14. Note here that `ogap` and `inf` are viewed as two exogenous variables which might have an influence on the endogenous variable `fedfunds`. By trial and error, we finally select the Markov switching model in Eq. (8.17) with order $p = 4$, the number of regimes 2, and the variance of the error ε_t as well as the regression coefficients having no switching to fit to the dataset. The estimated model is shown in the table Markov Switching Model Results in the following Python code. We see that the estimates of most parameters are statistically significant, especially the estimates of the autoregressive coefficients at lag 4 whether the regime is 0 or 1. What is more, observing the ACF plot of the model residual series shown in Fig. 8.15 and the p -value plot for the Ljung-Box test of the residuals shown in Fig. 8.16, we conclude that the residual series can be seen as a white noise. (By the way, we have tried to estimate several other Markov switching models, but their residuals all appear serially correlated.) Therefore, we have built an adequate Markov switching model for the three variables. Lastly we can plot the smoothed probabilities of the high and low federal funds rate regimes shown in Fig. 8.17 and calculate the expected durations of the high and low regimes. It turns out that the high regime is expected to persist for about 3.6 quarters, whereas the low regime is expected to persist for about 38.4 quarters.

```
>>> import numpy as np
>>> import pandas as pd
```

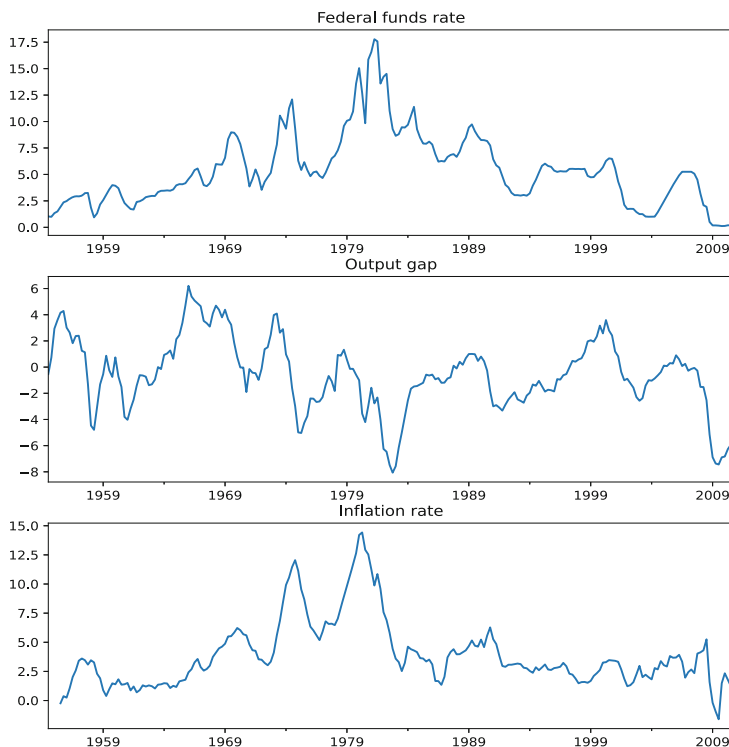


Fig. 8.14 Time series plots of the three US economic variables in Example 8.7

 pyTSA_MacroUS2

```
>>> import statsmodels.api as sm
>>> import matplotlib.pyplot as plt
>>> from PythonTsa.plot_acf_pacf import acf_pacf_fig
>>> from PythonTsa.LjungBoxtest import plot_LB_pvalue
>>> from statsmodels.tsa.regime_switching.tests.test
    _markov_regression import fedfunds, ogap, inf
>>> index=pd.date_range('1954-07-01', '2010-10-01', freq='QS')
>>> dta_fedfunds = pd.Series(fedfunds, index=index)
>>> dta_ogap = pd.Series(ogap, index=index)
>>> dta_inf = pd.Series(inf, index=index)
>>> dta_inf.head()
1954-07-01      NaN
1954-10-01      NaN
1955-01-01      NaN
1955-04-01      NaN
1955-07-01    -0.234724
>>> fig = plt.figure()
>>> dta_fedfunds.plot(ax=fig.add_subplot(311))
>>> plt.title("Federal funds rate")
>>> dta_ogap.plot(ax=fig.add_subplot(312))
```

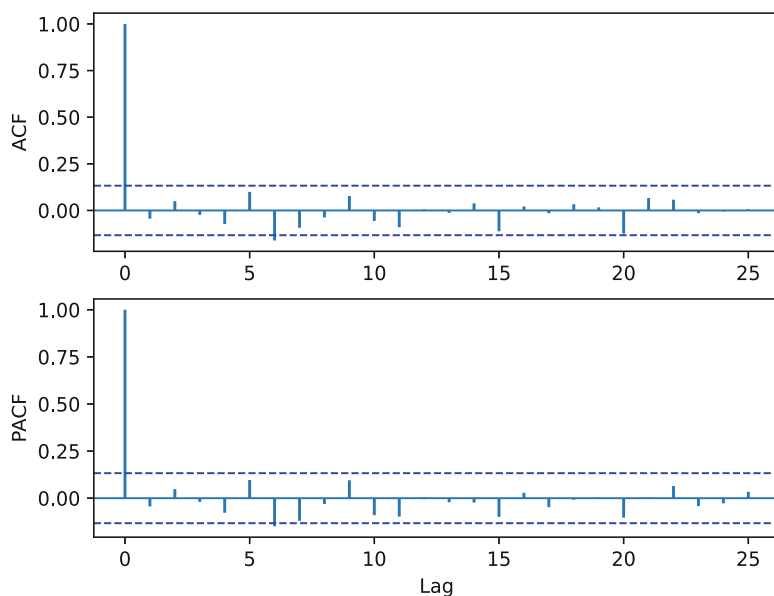



Fig. 8.15 ACF and PACF plots of the residuals of the Markov switching model in Example 8.7

 pyTSA_MacroUS2

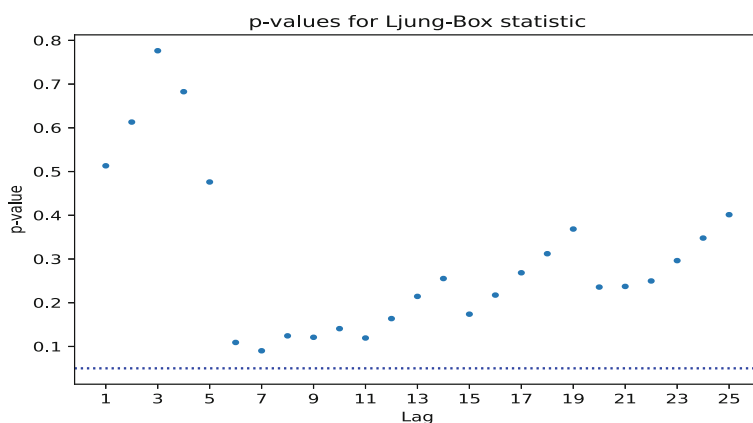


Fig. 8.16 p-Value plot for the Ljung-Box test of the model residual series in Example 8.7

 pyTSA_MacroUS2

```
>>> plt.title("Output gap")
>>> dta_inf.plot(ax=fig.add_subplot(313))
>>> plt.title("Inflation rate")
>>> plt.show()
>>> exog1 = pd.concat((dta_ogap, dta_inf), axis=1).iloc[4:]
>>> mymod_fedfunds = sm.tsa.MarkovAutoregression(
```

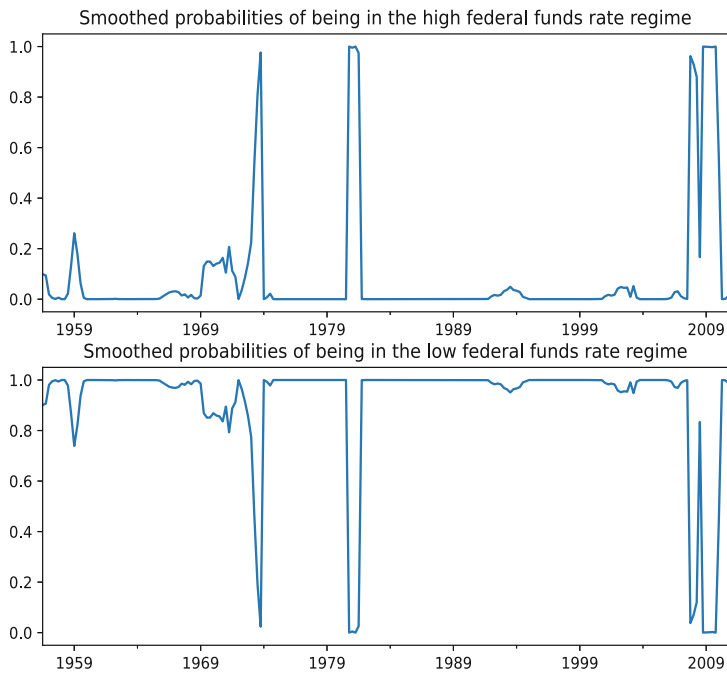


Fig. 8.17 Smoothed probabilities of being in the high and low rate regimes in Example 8.7

 [pyTSA_MacroUS2](#)

```
dta_fedfunds.iloc[4:], k_regimes=2, order=4, exog=exog1)
>>> myres_fedfunds = mymod_fedfunds.fit()
>>> print(myres_fedfunds.summary())
```

Markov Switching Model Results

```
=====
Dep. Variable:          y      No. Observations:      218
Model:      MarkovAutoregression  Log Likelihood      -264.575
Date:      Thu, 16 Dec 2021  AIC      559.149
Time:      22:42:03  BIC      609.917
Sample:      07-01-1955  HQIC      579.655
              - 10-01-2010
Covariance Type:      approx
                  Regime 0 parameters
=====
```

	coef	std err	z	P> z	[0.025	0.975]
const	0.5097	0.289	1.766	0.077	-0.056	1.075
ar.L1	0.1808	0.130	1.386	0.166	-0.075	0.437
ar.L2	0.2104	0.172	1.220	0.222	-0.128	0.548
ar.L3	-0.4561	0.206	-2.212	0.027	-0.860	-0.052
ar.L4	-0.5517	0.208	-2.654	0.008	-0.959	-0.144

```
                  Regime 1 parameters
=====
```

```
=====
              coef      std err          z      P>|z|      [0.025      0.975]
-----
const         0.1389         0.335         0.415         0.678         -0.518         0.796
ar.L1          1.1467         0.072        15.930         0.000          1.006         1.288
ar.L2         -0.4909         0.097         -5.079         0.000         -0.680        -0.301
ar.L3          0.4459         0.093          4.813         0.000          0.264         0.627
ar.L4         -0.1438         0.065         -2.218         0.027         -0.271        -0.017
```

Non-switching parameters

```
=====
              coef      std err          z      P>|z|      [0.025      0.975]
-----
x1             0.2657         0.042          6.372         0.000          0.184         0.347
x2             1.2091         0.032         38.107         0.000          1.147         1.271
sigma2         0.5332         0.056          9.502         0.000          0.423         0.643
```

Regime transition parameters

```
=====
              coef      std err          z      P>|z|      [0.025      0.975]
-----
p[0->0]        0.7189         0.133          5.400         0.000          0.458         0.980
p[1->0]        0.0261         0.014          1.841         0.066         -0.002         0.054
```

Warnings:

```
[1] Covariance matrix calculated using numerical
    (complex-step) differentiation.
>>> myresid=myres_fedfunds.resid
>>> acf_pacf_fig(myresid, both=True, lag=25); plt.show()
>>> plot_LB_pvalue(myresid, noestimatedcoef=0, nologs=25)
>>> plt.show()
>>> fig = plt.figure()
>>> ax1=fig.add_subplot(211)
>>> myres_fedfunds.smoothed_marginal_probabilities[0].plot(ax=ax1)
>>> plt.title('Smoothed probabilities of
              being in the high federal funds rate regime')
>>> ax2=fig.add_subplot(212)
>>> myres_fedfunds.smoothed_marginal_probabilities[1].plot(ax=ax2)
>>> plt.title('Smoothed probabilities of
              being in the low federal funds rate regime')
>>> plt.show()
>>> print(myres_fedfunds.expected_durations)
[ 3.55737378 38.37500431]
```

Example 8.8 (Absolute Returns of S&P 500) The dataset to be analyzed is also built in the Python package `statsmodels` and denoted as `areturns`. It is the absolute returns of S&P 500 and a weekly time series from May 4, 2004 to May 3, 2014. Its time series plot is shown in Fig. 8.18 and plainly exhibits heteroscedasticity. We select the Markov switching autoregressive model with order 4 to be fitted to the dataset. The estimated model is shown in the first Markov Switching Model Results table in the following Python code. We observe that the estimate for the variance of error ε_t is significantly 0.6316

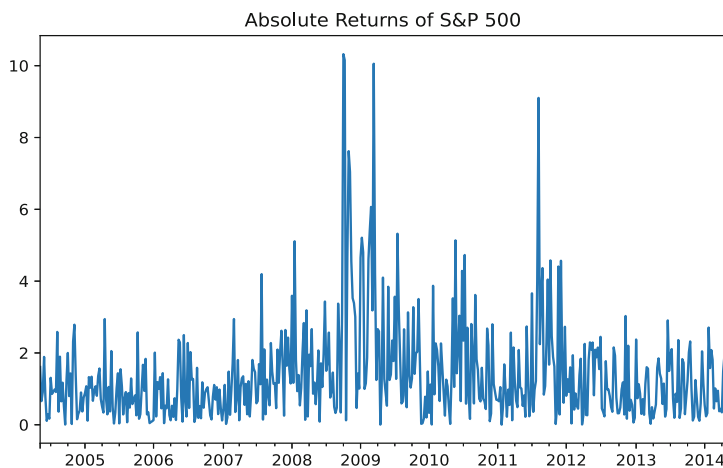


Fig. 8.18 Time series plot of the absolute returns of S&P 500 in Example 8.8

 [pyTSA_MarkovReturnsSP500](#)

(small) when the regime is 0, whereas significantly 2.4475 (big) when the regime is 1. This result illustrates that the switching is necessary. The ACF and PACF plots of the model residuals shown in Fig. 8.19 appear like ones of a white noise. However, the pity is that the p -value plot for the Ljung-Box test of the model residuals shown in Fig. 8.20 is undesirable. Now we take a logarithm of the time series data areturns and are to build a Markov switching autoregressive model for the log areturns. The estimated model is shown in the second Markov Switching Model Results table in the following Python code. In this case, the ACF and PACF plots of the model residuals shown in Fig. 8.21 are like ones of a white noise, and further the p -value plot for the Ljung-Box test of the model residuals shown in Fig. 8.22 is desirable (viz., all the p -values are greater or equal to 0.05). Thus the second estimated model is more acceptable.

```
>>> import numpy as np
>>> import pandas as pd
>>> import statsmodels.api as sm
>>> import matplotlib.pyplot as plt
>>> from PythonTsa.plot_acf_pacf import acf_pacf_fig
>>> from PythonTsa.LjungBoxtest import plot_LB_pvalue
>>> from statsmodels.tsa.regime_switching.tests.test
    _markov_regression import areturns
>>> index=pd.date_range("2004-05-04", "2014-5-03", freq="W")
>>> dta_areturns = pd.Series(areturns, index=index)
>>> dta_areturns.plot(title="Absolute Returns of S&P 500")
>>> plt.show()
>>> myMautol= sm.tsa.MarkovAutoregression(dta_areturns,
    k_regimes=2, order=4, switching_variance=True)
>>> myfitl=myMautol.fit()
>>> print(myfitl.summary())
```

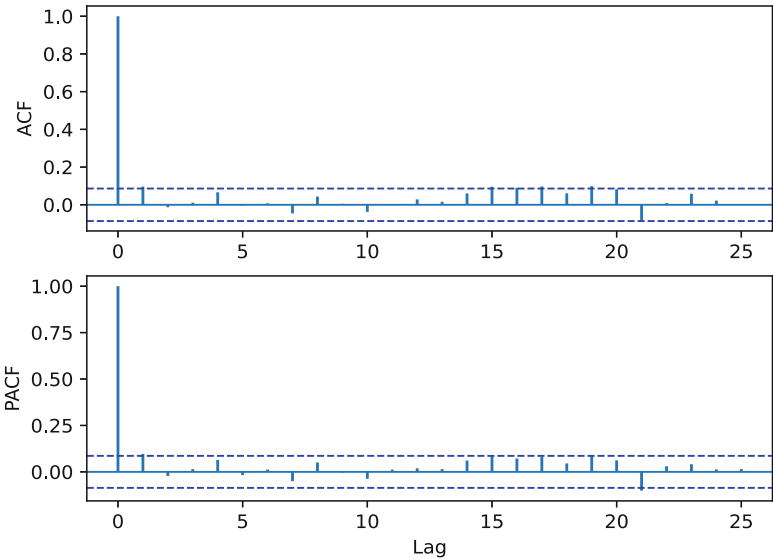


Fig. 8.19 ACF and PACF plots of the model residuals of the absolute returns in Example 8.8

 [pyTSA_MarkovReturnsSP500](#)

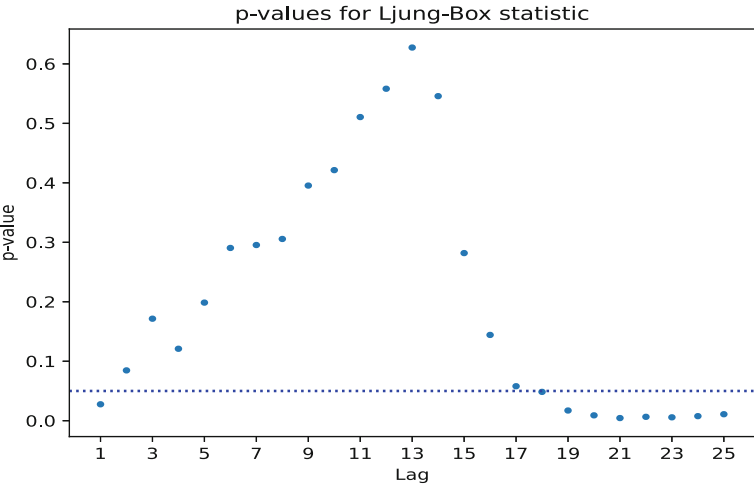


Fig. 8.20 *p*-Values for the Ljung-Box test of the model residuals of areturns in Example 8.8

 [pyTSA_MarkovReturnsSP500](#)

```

Markov Switching Model Results
=====
Dep. Variable:      y      No. Observations:      517
Model:      MarkovAutoregression      Log Likelihood      -740.582
Date:      Mon, 20 Dec 2021      AIC      1509.164
```

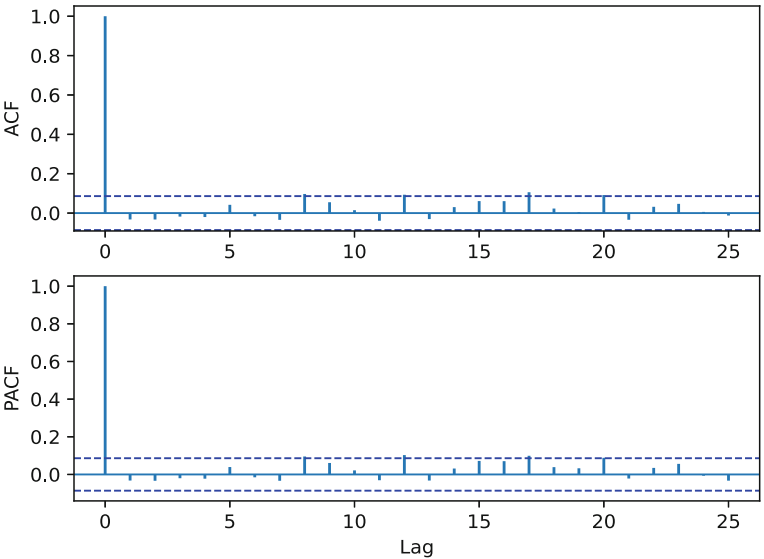


Fig. 8.21 ACF and PACF plots of the model residuals of the log areturns in Example 8.8

 [pyTSA_MarkovReturnsSP500](#)

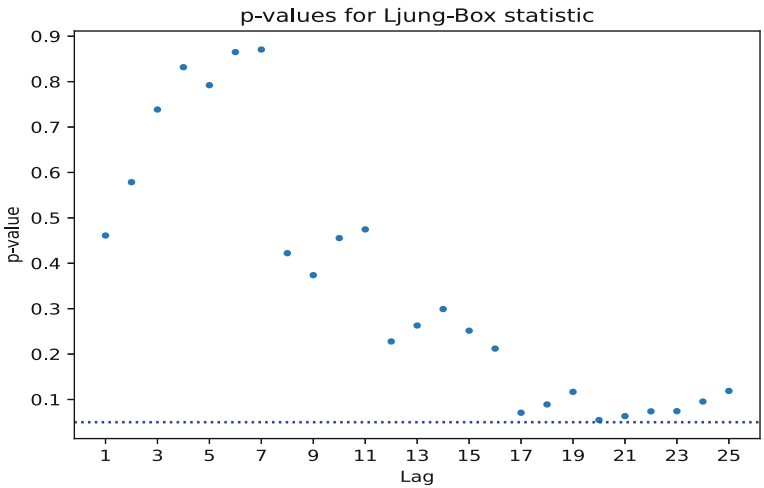


Fig. 8.22 *p*-Values for the Ljung-Box test of the model residuals of the log areturns in Example 8.8

 [pyTSA_MarkovReturnsSP500](#)

Time:	04:01:16	BIC	1568.636
Sample:	05-09-2004	HQIC	1532.467
	- 04-27-2014		
Covariance Type:	approx		

```

Regime 0 parameters
=====
      coef      std err          z      P>|z|      [0.025      0.975]
-----
const      1.0694        0.062     17.153     0.000        0.947        1.192
sigma2      0.6316        0.062     10.133     0.000        0.509        0.754
ar.L1       0.0538        0.048      1.127     0.260       -0.040        0.147
ar.L2       0.0142        0.047      0.306     0.760       -0.077        0.105
ar.L3       0.0564        0.055      1.026     0.305       -0.051        0.164
ar.L4       0.1808        0.053      3.387     0.001        0.076        0.285

Regime 1 parameters
=====
      coef      std err          z      P>|z|      [0.025      0.975]
-----
const      4.4808        0.362     12.363     0.000        3.770        5.191
sigma2      2.4475        0.686      3.568     0.000        1.103        3.792
ar.L1       1.1134        0.130      8.586     0.000        0.859        1.368
ar.L2      -0.2053        0.170     -1.209     0.227       -0.538        0.128
ar.L3      -0.3090        0.169     -1.828     0.068       -0.640        0.022
ar.L4      -0.0948        0.110     -0.858     0.391       -0.311        0.122

Regime transition parameters
=====
      coef      std err          z      P>|z|      [0.025      0.975]
-----
p[0->0]     0.9461        0.017     57.150     0.000        0.914        0.979
p[1->0]     0.7261        0.092      7.888     0.000        0.546        0.907
=====

```

Warnings:

```

[1] Covariance matrix calculated using numerical
    (complex-step) differentiation.
>>> myresid1=myfit1.resid
>>> acf_pacf_fig(myresid1, both=True, lag=25); plt.show()
>>> plot_LB_pvalue(myresid1, noestimatedcoef=0, nolags=25)
>>> plt.show()
>>> Ldta=np.log(dta_areturns)
>>> myMauto= sm.tsa.MarkovAutoregression(Ldta,
    k_regimes=2, order=4, trend='n', switching_variance=True)
>>> myfit=myMauto.fit()
>>> print(myfit.summary())

```

Markov Switching Model Results

```

=====
Dep. Variable:          y      No. Observations:      517
Model:      MarkovAutoregression      Log Likelihood      -774.828
Date:      Tue, 21 Dec 2021      AIC      1573.655
Time:      06:15:31      BIC      1624.632
Sample:      05-09-2004      HQIC      1593.630
            - 04-27-2014

```

Covariance Type: approx

```

Regime 0 parameters
=====
      coef      std err          z      P>|z|      [0.025      0.975]
-----
sigma2      6.3165        1.774      3.560     0.000        2.839        9.794
ar.L1       0.5610        0.080      6.977     0.000        0.403        0.719

```

```
ar.L2      0.0465      0.154      0.301      0.763      -0.256      0.349
ar.L3      0.0727      0.188      0.387      0.699      -0.296      0.441
ar.L4     -0.3439      0.096     -3.590      0.000      -0.532     -0.156

Regime 1 parameters
=====
      coef      std err          z      P>|z|      [0.025      0.975]
-----
sigma2      0.7327      0.062     11.830      0.000      0.611      0.854
ar.L1      0.0732      0.040      1.839      0.066     -0.005      0.151
ar.L2      0.0719      0.041      1.761      0.078     -0.008      0.152
ar.L3      0.0718      0.038      1.886      0.059     -0.003      0.146
ar.L4      0.1677      0.041      4.138      0.000      0.088      0.247

Regime transition parameters
=====
      coef      std err          z      P>|z|      [0.025      0.975]
-----
p[0->0]  6.567e-07         nan         nan         nan         nan         nan
p[1->0]   0.1206      0.033      3.710      0.000      0.057      0.184
=====

Warnings:
[1] Covariance matrix calculated using numerical
    (complex-step) differentiation.
>>> myresid=myfit.resid
>>> acf_pacf_fig(myresid, both=True, lag=25); plt.show()
>>> plot_LB_pvalue(myresid, noestimatedcoef=0, nolags=25)
>>> plt.show()
```

In this chapter, we have addressed the linear state space model and its application as well as the Markov switching model. They are both useful in various domains. Of course, the handling here is introductory. For more information on state space methods including nonlinear and non-Gaussian state space models, see, for example, Tsay and Chen (2019); Douc et al. (2014); and Durbin and Koopman (2012). As to the Markov switching model, the reader can refer to Frühwirth-Schnatter (2006) and Kim and Nelson (1999), among others.

Problems

8.1 Verify that the state space models (8.4) and (8.5) specify the ARMA(1, 1) model (8.3) and so the former is equivalent to the latter.

8.2 Find a state space form for the causal AR(*p*) model as follows:

$$Y_t = \varphi_1 Y_{t-1} + \cdots + \varphi_p Y_{t-p} + \varepsilon_t.$$

8.3 If the state space model (8.7)–(8.9) is time-invariant, that is, the parameters **c**_{*t*}, **F**_{*t*}, **R**_{*t*}, **Q**_{*t*}, **d**_{*t*}, **Z**_{*t*}, and **H**_{*t*} do not change over time and derive the Kalman filtering (forecasting and smoothing) formulae.

8.4 The dataset “realGdpConsInv” in the folder Ptsadata is from the Federal Reserve Bank of St. Louis, USA. It consists of the three economic variables `realgdp`, `realcons`, and `realinv`. Let `realcons` and `realinv` be the exogenous regressors. Then fit the SARIMAX model to the dataset. Lastly conduct an analysis of the model residuals.

8.5 As for the variable `logret` in Example 8.6, let the lagged `logret` be an exogenous regressor. Then use the function `MarkovRegression` to fit the following Markov switching model

$$X_t = \mu_{S_t} + X_{t-1}\beta_{S_t} + \varepsilon_t, \quad \varepsilon_t \sim \text{WN}(0, \sigma_{S_t}^2)$$

to the dataset in Example 8.6. Lastly compare your results with the results of Example 8.6.

8.6 Fit the Markov switching model with the number of regimes 2

$$X_t = \mu_{S_t} + X_{t-1}\beta_{S_t} + \varepsilon_t, \quad \varepsilon_t \sim \text{WN}(0, \sigma^2)$$

to the time series `fedfunds` in Example 8.7, and then conduct an analysis of the model residuals. Besides, how about the case of the variance of the error ε_t having switching.

8.7 As for the three variables `fedfunds`, `ogap`, and `inf` in Example 8.7, let the lagged `fedfunds` be a part of the exogenous regressors, and then use the function `MarkovRegression` to build a Markov switching (auto)regression model like the following model

$$X_t = \mu_{S_t} + (X_{t-1}, Y_t, Z_t)\beta_{S_t} + \varepsilon_t, \quad \varepsilon_t \sim \text{WN}(0, \sigma^2)$$

with the number of regimes 2 where X_{t-1} is the lagged `fedfunds`, Y_t is `ogap`, and Z_t is `inf`. Finally make a diagnostic check of the estimated model. Is it appropriate?

8.8 Consider the Markov switching model

$$X_t = \mu_{S_t} + X_{t-1}\beta_{S_t} + \varepsilon_t, \quad \varepsilon_t \sim \text{WN}(0, \sigma_{S_t}^2).$$

Use the Python function `MarkovRegression` to fit the model to the dataset in Example 8.8. Are you satisfied with your results?